KIX Start

KIX 18 Admin Manual - EN

Issued on: 26.02.2024

Table of contents

1 Introduction

1.1 About KIX 18

KIX version 18 is a browser-based open source service management system for IT teams, technical service, maintenance, after sales services and facility management. As an open source ticket system, KIX offers the right service solution for every company size and requirement, from SMEs to public or state institutions and authorities to multinational corporations:

KIX Start

Software-based service management
ideal for beginners.

⭐ ⭐ ⭐ ⭐ ⭐

Basic solution
easy to integrate
free of charge & immediately ready for use
for small service organisations

KIX Pro

The comprehensive business package for professional
professional IT and technical service.

⭐ ⭐ ⭐ ⭐ ⭐

customised & cross-departmental
incl. Self Service Portal and
diverse additional modules
for medium-sized and large companies

KIX Cloud

Our cloud offer (SaaS) for KIX Pro.
Intuitive, flexible, including full managed services

★★★★★

immediate provision
high cost transparency
incl. Self Service Portal
optimal for SMEs

ITIL® 4 Certified

The IT service management system KIX Pro 18 has been awarded SERVIEW Certifiedtool for 15 out of 19 possible **ITIL® 4 practices.** This makes KIX Pro 18 the world's most functionally powerful certified open source tool for the ITSM sector.

In order to make the use of ITIL® 4 as easy as possible for users, **the add-on "ITIL Practices"** was developed. These are templates with which changes can be made, problem cases, services and management tasks can be processed and numerous reports can be produced.

Optional extensions for KIX:

Field Agent App

Mobile app for iOS and Android for on-site technical service.
Can be used for KIX Start and KIX Pro.

Add-on "ITIL Practices"

Add-on for process management and
for the automation of workflows.
Can be used for KIX Pro.

Maintenance Plan

Add-on to support
recurring maintenance tasks.
Can be used for KIX Pro.

1.1.1 Additional modules from the KIX Connect product line:

The KIX Connect product line can only be used with KIX Pro.

Connect Database

Use of external database
connections as a data source.

Connect Webservice

Integration of HTTP/s web services
in KIX (without direct database access)

Connect Opsi

Provision of asset data
in KIX on the basis of the
inventory solution opsi

Connect Baramundi

Provision of asset data
in KIX on the basis of the
inventory solution Baramundi

1.2 Community

KIX is a dynamically growing software that is subject to a constant process of further development. As an open source company, it is important to us that we do not do this alone. We cordially invite you to become part of our KIX community and to support us with helpful hints for the further development of KIX and the user information.

1.3   About this manual

This manual covers the administration of KIX version 18 via the GUI of the Admin module and is intended for all users who have been entrusted with the administration of KIX 18. It is intended to serve as a reference work in their daily work and to enable them to manage the administrative tasks of KIX 18 (hereafter KIX).

The manual is divided into the sections "KIX Start" and "KIX Pro". The section "KIX Start" deals with the administration of KIX Start and the basic functions. The section "KIX Pro" is intended as a supplement to the KIX Start manual and describes the administration of the additional functions contained in KIX Pro as well as the KIX Pro add-on modules.

For users of KIX 18 we have created a separate manual, which is also divided into the sections "KIX Start" and "KIX Pro". The user manual is intended for all agents and KIX users who are charged with service provision.

Note: This manual contains configuration **examples** to explain the respective topics. If you transfer these examples to your system, you do so at your on own responsibility. We do not assume any liability for this.

Further documentation on KIX versions 17 and 18 can be found at https://docs.kixdesk.com/.

> ⓘ   This edition of the manual refers to the release status: see Release Information

1.4   To comprehension

In these instructions, the masculine form according to the grammar is used in a neutral sense. It always appeals to all male, female and diverse readers. Gender variants are not used for reasons of legibility and understanding of the text. We ask all readers for their understanding for this simplification in the text.

2 Glossary

Term	Explanation	Usage
Action	KIX distinguishes between ticket actions and item actions. Ticket actions are displayed in the ticket detail view as buttons in the action line and are available in the agent portal and/or the self-service portal, depending on the configuration. Article actions are available as a lateral label on the right side margin of an article, depending on the user's authorisation.	E.g. in the ticket
Add-on	Independent software component which, once installed, extends the scope of KIX. Add-ons are not included in the scope of delivery as standard.	E. g. Add-on "ITIL Practices"
Admin	Person with advanced user rights who manages the system, assigns user rights, and enforces usage guidelines	Separate module within KIX
Agent	A user of the agent portal, e.g. a service employee or an employee who processes an inquiry.	E.g. when creating a ticket
Article	Describes a work step within a process/ticket. Is a documented communication or comment generated by an agent or received from a user.	In the ticket
Assets	Any operating resource that has a function within the business processes, e.g. computers, software, servers, telephones, machines, vehicles, rooms, buildings, and contracts. Assets can also be referred to as: configuration item, config item, and CI.	In the Assets module
Attribute	Property or characteristic that can be assigned to an object (e.g. asset).	In the Advanced Search, in the class definition of assets

Term	Explanation	Usage
Channel	Determines the method of communication (e.g. email, note, etc.)	E.g. in the ticket
CMDB	The Configuration Management Database is referred to as the Asset Database in KIX. It was developed for access to and management of assets.	In the Assets module
Configuration Item, Config Item, CI	Any operating resource that has a function within the business processes, e.g. computers, software, servers, telephones, machines, vehicles, rooms or buildings, and contracts. Configuration items are also referred to assets.	In the Assets module
Contact	A contact person within the customer's organization. The contact is allocated to a ticket as the contact person.	E.g. when creating a ticket
Customer	An organization/your customer.	E.g. in the Customer Dashboard
Dialog	Separate window (overlay) with form fields or additional information.	E.g. send email from within ticket, Form dialogs in Admin module
Drag & Drop	Computer function enabling you to click/select graphical elements (e.g. icons, passages in a text, etc.), and drag them across the screen with the mouse, and drop them somewhere else.	
Failure Time	Represents the period of time during which an asset or service is not available during the agreed service time.	

Term	Explanation	Usage
FAQ	Knowledge database for compiling information: • Particularly frequently asked questions • Treatment of frequently occurring problems • Manuals, product data sheets, documentation etc.	Separate menu item on the Home Dashboard
Field Agent App	Mobile application for fitters, service technicians, etc. Tickets can be processed by the technician on site.	
First Response Time	If a service fails within a company, an agent must respond to this failure within a specified period of time. The time begins with the notification of the incident and ends with first contact with the customer.	In the ticket
Icon	Symbol, graphical illustration of an element or menu item.	E.g. menu bar on the Home Dashboard
Impact	Describes the extent of the consequences that an incident or problem has on a work process.	
Incident	Describes an incident or other issue relating to an IT service or asset.	
ITIL	IT Infrastructure Library. Collection of "best practice" for the delivery of (IT) services. KIX is increasingly looking at the agile process models of ITILv4.	Add-on "ITIL Practices"
Lane	Blocks of information dealing with similar topics in the zoom view; can be expanded and collapsed.	Contain aspects such as master data information, version details, assignments, permissions, etc.

Term	Explanation	Usage
Object	Technical designation for items saved in the system such as tickets, assets, etc. and for technical system components e.g. dynamic fields.	E.g. in the Admin module
Organisation	Brings together all data for a company and can be allocated to individual contacts.	In the Customer module
Overlay	Separate window (dialog) with form fields or additional information.	E.g. send email from within ticket, form dialogs in Admin module
Owner	Person who owns the ticket and/or is actively processing it.	In the ticket
Pattern	Basic character string used for translation purposes.	In the Admin module
Permission	Permissions define which work steps and actions an agent may execute, and which objects and information a user may view.	In the Admin module
Person Responsible	Person who can define which employees should process the ticket. They are notified when the process is being worked on. The person responsible can also be the owner.	In the ticket
Priority	Priority levels denote different levels of urgency and can be used to categorize tickets. They are indicated by a traffic light color code.	In the ticket
Queue	Element used to classify requests. The use of multiple different queues (e.g. for specific departments) enables inbound tickets to be categorized and organized. In KIX, queues are referred to as teams.	In the ticket
Response Time	Another word for First Response Time.	

Term	Explanation	Usage
Role	This is where you assign the permissions that define what the agent is allowed to/not allowed to edit in the system. Agents can have several roles.	In the Admin module
Self Service Portal	Client portal, which is connected to KIX. Only usable with KIX Pro	
Service	Stands for a service/performance that a company offers. Services are part of Service Level Agreements (SLA).KIX Pro includes the asset class "Service" to manage services.	In the ticket, e.g. maintenance or repair services
Service Level Agreement (SLA)	Agreement between a service provider and its customer. An SLA regulates which services the provider provides and which service standards are defined that the provider undertakes to comply with (e.g. response time to enquiries, resolution time in the event of problems, etc.).	E.g. in the ticket
Sidebar	Additional information at the edge of the screen, which can be expanded and collapsed as required.	E.g. the digital notepad
Solution Time	Period of time during which a failed service must be restored. The time begins with the notification of the incident and ends once it has been resolved. To mark ticket as resolved, set the ticket state "closed".	E.g. in the Admin module
State	The state defines the processing state of a ticket.	In the ticket
SysConfig	Abbreviated term denoting the system configuration of KIX.	In the Admin module, menu System > SysConfig
Team	Element used to classify requests. The use of multiple different teams enables inbound tickets to be categorized and organized. Another word for team is queue.	In the ticket

Term	Explanation	Usage
Template	Individually configured input mask for the creation of application or incident-related tickets (malfunction, average, holiday request, etc.). Individual templates are only possible in KIX Pro. KIX Start contains a customisable default template for all tickets.	In the ticket
Ticket	A service operation including the totality of all its communication and documentation steps in the form of incoming and outgoing emails, documented calls, notes or customer feedback via the web front end. They • are requested by a contact (organisation). • are assigned to a team (queue) as well as a processor and responsible agent. • have a priority as well as a type and status.	In the Ticket module
Ticket Lock	Means that special actions for a ticket can only be performed by the owner. This is designed to prevent the ticket from being processed by several people at the same time, or make this less likely.	In the Ticket module
Type/Ticket Type	Classifies the nature of a request, e.g. incident, problem, service request, etc.	In the Ticket and Admin modules
Update Time	Describes the time during which communication (call, email) with the customer (organization) must take place. If there has been no communication with the customer upon reaching the update time on the ticket, the ticket is considered escalated.	In the ticket
User	A person who works with the system. For example, an agent, the admin, a contact with system rights, a user of the Self Service Portal.	
Widget	Term for small, separate windows in a program; a portmanteau of the words "window" and "gadget".	Diagram widgets, table widgets, sidebar widgets

Term	Explanation	Usage
Workflow	A workflow describes the concatenation of different, consecutive steps within a business process. In the Admin Module, the administration of templates, actions and rule sets can be found under Workflow.	E. g. Add-on "ITIL Practices"

3 Help and Contact

We hope that this manual provides you with sufficient support for the administration of KIX 18. If you still have questions about KIX 18, please ask them in our forum: https://forum.kixdesk.com. There you will also find a range of information that can help you answer your questions.

You can also find help, tips and tricks on the Welcome page, which you can access at any time. To do so, click on the question mark in the header area of KIX:

You can also find a German-language quick start guide as well as other German-language videos on the KIX on our YouTube channe[1]l .

Alternatively, you can use the KIX Service Software GmbH conclude a service contract and thus take advantage of our competent support service. Please contact us by email or telephone.

- E-Mail: info@kixdesk.com[2]
- Telefon: 03 71/2 70 95-620

---

1 https://www.youtube.com/watch?v=CqCNIvdhayo&list=PLyKz2PdXNlw3b13FZEPP9CcFMTFRvFQh-
2 mailto:info@kixdesk.com

4 Installation

You can choose between different ways of using KIX 18:

- **KIX 18 On Premises:** Local installation on your own server
- **KIX 18 Cloud:** Application in the KIX Cloud - without installation

Content on this page:

4.1 System requirements for a client

For the host environments of KIX 18 we recommend the following operating systems / environments:

- CentOS/RHEL
- Debian
- Ubuntu
- MacOS

4.1.1 Screen sizes

Monitor	Minimum requirement
Resolution	Desktop: 1280px x 720px
Format	• Desktop: 16:9 • Tablet: 10"

4.1.2 Supported browsers

4.1.2.1 Client KIX 18: Agent Portal

Supported Browsers (JavaScript enabled)	Versions
Mozilla Firefox	103+

Supported Browsers (JavaScript enabled)	Versions
Chrom/Chromium	103+
MS Edge (Chromium based)	101+

4.1.2.2   Client KIX 18: Self Service Portal

Supported Browsers (JavaScript enabled)	Versions
Mozilla Firefox	103+
Chrom/Chromium	103+
MS Edge (Chromium based)	101+
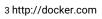
4.1.2.3   Field Agent App

Supported operating systems (smartphone)	Versions
iOS	15+
Android	10+

4.2   KIX On Premises Installation

KIX is installed via already pre-configured Docker containers and brings its own operating environment and database. This facilitates the use of the software on the various operating systems. Only a compatible Docker version (correctly and completely installed) must be present and running on the host environment. Therefore, we recommend checking whether the planned host environment supports the use of Docker before installing KIX 18. The use of KIX 18 requires Docker version 20.x and Docker Compose 2.1. More information at docker.com[3] or https://docs.docker.com/engine/install/linux-postinstall/.

For the installation of KIX 18 on your own server, use the files deposited in GitHub. For productive use, we recommend an installation under Linux. Below you will find a very compact quick guide for on-premises

---

3 http://docker.com

installation. You can find detailed installation instructions at https://github.com/cape-it/kix-on-premise/blob/master/deploy/linux/README.md.

Docker is operated via the console (terminal). Use the following console commands.

ℹ️The user on the command line must have execution rights to the Docker service.

How to install KIX (short version):

1. Install Docker and Docker Compose on the host system (usually "localhost" / hereinafter "your.docker.host"):
    - Docker: https://docs.docker.com/engine/install/
    - Compose: https://docs.docker.com/compose/install/

        ℹ️ *The scripts require Docker-Compose as a standalone installation, although newer Docker versions already include Compose as a plugin.*

2. Retrieve the initial Docker environment:
    - `cd /opt`
    - `git clone https://github.com/cape-it/kix-on-premise.git`

3. Change to the unpacked Linux or Windows directory.
    - `cd kix-on-premise/deploy/linux`
    - `cd kix-on-premise/deploy/windows`

4. **Recommendation**: Make a backup copy of the following files:
    - `environment`
    - `proxy/non-ssl.conf`
    - `proxy/ssl.conf`

5. SSL Setup
    - Add your SSL certificates to the Docker environment in the following files:
        - Certificate: `proxy/ssl/certs/server.crt`
        - Key: `proxy/ssl/certs/server.key`

    - Using SSL **instead** of non-SSL
        - Disable all server entries: in file `proxy/non-ssl.conf` (via comment #)
        - Enable all server entries: in file `proxy/ssl.conf` (by removing #)
        - **Note:** If this setup is selected, application ports defined in `BACKEND_PORT, FRONTEND_PORT, SSP_PORT` are used as drop-in replacements.

    - Using SSL in **addition** to non-SSL

- Activate all server entries: in file proxy/ssl.conf (by removing #)
- Change port setting: in file proxy/ssl.conf
  - from `80` to `443`
  - from `8080` to `8443`
  - from `9080` to `9443`

6. Start or stop KIX
   - Change to the unpacked Linux or Windows directory
     - c`d /kix-on-premise/linux`
     - `cd /kix-on-premise/window`
   - Execute the start script:
     - Windows: start.ps1
     - Linux: `./start.sh`
   - Run stop script:
     - Windows: `stop.ps1`
     - Linux: ./ `stop.sh`

7. After starting KIX18, log into the system. With unchanged configuration of the `environment` file with the following data:
   - Portal access
     - Agentenportal: http://your.docker.host:20001
     - Self Service Portal (nur KIX Pro): http://your.docker.host:20001
     - REST-API Backend: http://your.docker.host:20000/
   - Registration data for initial registration:
     - User: "admin"
     - Password: "Passw0rd"

8. Then use the Setup Assistant to set a new admin password and a "Super User" and to carry out the basic configuration of KIX.

ⓘ **Info**

Information on the installation of KIX Pro to KIX.Cloud as well as on the additional modules can be found under: Installing KIX Pro

---

4.3 KIX On Premises Update

To update an existing KIX environment, please update the Docker configuration. The instructions can also be found under: https://github.com/cape-it/kix-on-premise/blob/master/deploy/linux/README.md.

How to update KIX (short version):

1. Update Docker Configuration
   - `user@DockerHost:/opt/kix-on-premise/# git pull`
2. Update KIX
   - `user@DockerHost:/opt/kix-on-premise/deploy/linux#` **`./stop.sh`**
   - `user@DockerHost:/opt/kix-on-premise/deploy/linux#` **`./start.sh`**

> ⚠️ **Important!**
>
> Update KIX with **every** update! The releases interlock and build on each other. Skipping versions can lead to functional gaps and malfunctions.

> ⓘ **Please Note**
>
> Note that Docker takes a conservative approach to cleaning up unused objects. The objects are generally not removed, which can lead to memory overflow. A cleanup can be done using the `prune` command. You can find more information on the page: https://docs.docker.com/config/pruning/.

4.4 GitHub

On GitHub we provide you with a range of information and scripts for KIX 18 and also for KIX 17: https://github.com/kix-service-software

4.5   Migration KIX 17 > KIX 18

You can transfer the data from KIX 17 to KIX 18. In KIX Start this is done by executing a console command in the System> Console menu. KIX Pro users can start the migration automatically via the GUI.

You can perform the migration completely or limit it to individual objects by specifying appropriate parameters in the console command.

The migration is carried out by KIX 18. KIX 18 (active system) fetches the data from the source system KIX 17 (passive system). You must be logged into KIX 18 as a user with admin rights in order to be able to carry out the migration (roles: system admin, superuser).

Content on this page:

You are welcome to contact our support if you have any questions about migration or need help.

Fig .: The migration via the console

It is possible to continue working during the migration. However, keep the following note in mind:

> ⚠ **Hint**
>
> The migration will take some time, depending on the volume of data, and leads to an increased system load on both KIX systems as well as on the hardware and network. You should therefore schedule the migration at a quiet time (weekends, public holidays).

4.5.1 Notes on Migration

KIX 18 answers various scenarios differently than KIX 17 and discards some of the historically grown structures. Please note that not all KIX17 features and behaviors have a 1:1 equivalent in KIX 18. KIX18 is largely a standalone product. Below you will find some information on relevant aspects.

- **Permissions Structure:** The permissions system in KIX 18 is rebuilt from the ground up (see also Permission concept KIX 18 (see page 645)). For necessary support during the migration, please contact our support.

- **Customer and Agent Users**: In KIX 18, "Contacts" form the basis for contact persons and agents. A contact can be assigned a user account. This makes the contact a user and he can log on to the Self Service Portal or as an agent to the system. The migration creates
    - from each KIX 17 customer contact a KIX 18 contact with the usage context "customer" (access to the Self Service Portal)
    - from each KIX 17 agent user, a KIX 18 contact with the usage context "Agent" (access to the Agent Portal).
    - If a KIX 17 login identifier exists as both a "customer user" and an "agent user", the resulting KIX 18 user receives both usage contexts.
    The basis for the migration is the existing customer and agent users in the KIX 17 database (see Preparations below).

- **Contacts and Organisations:** In KIX 18, contacts do not necessarily have to be assigned an organisation. The tickets for these contacts do not need an organisation either. Therefore, when migrating contacts and tickets, no organisation will be created if the contact or ticket does not have an assigned organisation.
    If no customer company is created for the CustomerID of a contact, the assignments are not transferred and the contacts exist without any assignment.

    > ⓘ **Until further notice, the following applies:** For the use of the SSP, the assignment of an organisation is required.

- **Article types:** KIX 18 no longer uses item types. Instead, communication channels are offered. The item type "Telephone" is no longer available in the standard and is replaced by the item type "Note". As a result, conversations are generally documented as conversation notes - regardless of whether they were conducted by telephone or face-to-face.
⚠️Specific item types are also replaced by communication channel "Note".

- **Config Item Attachments:** KIX 17 offers two options for CI attachments. The recommended and still existing one is the use of CI attributes of the type Attachment. Changing these attachments will continue to create an asset version in KIX 18. The non-versioning attachments known from KIX 17 will not be continued in assets. Any attachments are adopted and converted into versioning attributes.
⚠️ Please note that the operation will change as a result.

- **Link Types:** KIX 17 uses some link types and relationships that do not seem plausible. In KIX 18, therefore, all link types between any objects are no longer permitted. The migration creates such links as "Relevant To" links.

- **Text modules:** Text modules no longer contain subject lines in KIX 18.

- **Configuration settings:** Configuration settings are not transferred and require specific consideration (see below: Advanced configuration transmission (see page 51)).

- **Ticket status types:** If your KIX installation uses its own ticket status types, the ticket statuses must first be changed to the default ticket status types.
⚠️Specific custom ticket status types can block the migration and must be removed before the migration.

4.5.2 Migration requirements

- Current KIX 17 system (version 17.14.0 - or higher)
- defined and activated pre-shared key (see page 44) mode in KIX 17
- KIX 17 from version 22 is required as the source system for the correct migration of SLA times (Target time and Fulfillment time). Otherwise, only the SLA master data will be transferred, but not the times.

4.5.3 Preparations

4.5.3.1 Contact data sources

The migration uses the entries in the customer_user table as the basis for transferring the contact records. If specific customer data source tables have been created in the KIX-DB, their contents must be transferred to the table "customer_user". Only data that corresponds to the predefined structure of the table will be transferred. I.e. if extended attributes (additional columns in the DB table customer_user) have not already been created as dynamic fields, they must first be converted into dynamic fields. Please contact our support if you need assistance.

If the source system uses an LDAP/Active Directory connection as a source for contact data, its contents must first be synchronised into the database table "customer_user" of KIX17. A script included in KIX17 (https://github.com/kix-service-software/kix17-start/blob/rel-17/scripts/syncLDAP2DB.pl) can be used for this purpose.

4.5.3.2 Organisation/customer numbers

KIX18 requires that organisations and thus customer numbers are explicitly stored in the system. In KIX17, this corresponds to the "Customer Companies". However, KIX18 no longer allows contacts without an organisation assignment. For this purpose, the migration automatically creates an organisation in KIX18 for each customer number from KIX17. For the correct transfer of ticket customer numbers and asset properties with attribute type "CustomerUserCompany", it is also mandatory to observe the notes of the previous section.

4.5.3.3 Other notes

Ticket statuses will only be migrated successfully if they correspond to the predefined status types. KIX18 does not support custom status types without in-depth configuration. If the source system uses specific ticket status types, therefore, change the statuses in the source system to usually existing status types before the migration starts.

KIX17 allows many configurations, some of them quite specific, which were sometimes also created with direct interventions in the database. There is a realistic chance that these configurations will cause problems in the migration of the data. If you are unsure, please contact our support.

> ⚠ **Hint**
>
> Follow the instructions for preparing the migration to avoid incorrect or incomplete data.

4.5.4 Migration flow chart

- Activate PSK mode in KIX 17 and define pre-shared key.
- Open console in KIX 18, select console command and specify transfer parameters, start execution
- Deactivate PSK mode in KIX 17 by deleting the pre-shared key
- Check and edit the transferred data.

4.5.5 Which data are transferred?

KIX 18 actively queries the content to be migrated from the KIX 17 instance. The retrieved content is adapted to the structures (e.g. new IDs) of KIX 18. Below you will find a list of which data is transferred. Individual attributes and system additions used in KIX 17 are not transferred. To do this, please contact our support.

These data are transmitted:

- Configuration data
    - Ticket status
    - Ticket types
    - Queues / Teams (the signatures are included)
    - Dynamic Fields Configurations
    - General Catalog classes
    - Asset class definitions
- Business data
    - Organization
    - Contacts (Customer User)
    - Contacts (Agent User)
    - Assets (Config Items) including versions
    - FAQs including attachments
    - SLAs
    - Tickets
        - Article including attachments
        - Ticket history
        - Dynamic fields
    - Replacement of IDs
        - The IDs from KIX 17 are replaced by new IDs in KIX 18
    - Connections
        - Assets for tickets
        - Assets to assets
    - Services (KIX Pro)
        - Assets of class services

4.5.6 Notes on the transferred data

Ticket data

- After the migration, all tickets of the source environment are present in the target environment, including their properties:
    - Articles: all articles incl. attachments and respective channel.
    - History: Entries in the ticket history refer to the migrated objects.
    - Time entries: The times recorded on the ticket are migrated.
    - Lock status: Tickets that a user has locked are also locked for him in KIX 18.
    - Ticket statuses and types: Newly created statuses and types in the target environment will be invalidated after the migration.
    - The deactivation of statuses and types occurs because ticket status workflows no longer exist and the old values are initially only retrieved for reasons of backward/history compatibility.
    - Priority: Mapping is applied for the following source priorities (source => target).
        - "1 very low" => "5 very low"
        - "2 low" => "4 low"
        - "4 high" => "2 high"
        - "5 very high" => "1 very high"
- An interrupted migration can be resumed. Only the tickets that have not been migrated or that have been partially migrated are considered.
- **Note:** The "Watch" flag on the ticket will not be migrated, i.e. a watched ticket has to be set to "Watch" again after migration.

Text modules

- After the migration, the text modules of the source environment are available in the target environment.
    - The categories of the KIX 17 text modules are stored in the keywords of the KIX 18 text modules.
        - The names of the category levels are recorded as individual keywords.
        - Spaces are replaced by an underscore ("_").
    - The placeholder substrings "<ORTS_" are replaced by "<KIX_".
- **Note:** The validity of placeholders is not verified.

Connections

- After the migration, all links from the source environment are available in the target environment.
    - Config items for tickets
    - Config items to config items
    - Config items for FAQs

- Tickets to tickets
- Tickets to FAQs
- Link types that do not exist or are not permitted are replaced by "RelevantTo".

Queues / teams

- After the migration, all "queues" of the source environment are available as "teams" in the target environment.
- The structure of the queues can be found in teams (upper / lower queues).
- The signatures of the source environment are deposited with the teams.
    - older placeholders of the type <OTRS_xyz ...> have been replaced by <KIX_xyz ...
      It may be that not all placeholders are compatible and require manual follow-up checks and adjustments.

CI class definitions

- After the migration, all "CI class definitions" (including all versions) of the source environment are available as "asset class definitions" of the target environment.
    - If a class with the same name already exists in KIX 18, the migration class is given the prefix "Migration-".
- CI attribute types that are no longer supported are displayed as attributes of the "Text" type.
- After the migration, the previously existing CI attributes of the type "EncryptedText" are available in the KIX as "Text". This means that the passwords stored in it are visible to users with rights to the assets.
- To accommodate the non-versioned CI attachments from KIX17, each asset class is additionally given the attribute "CIAttachments" (last attribute) of the type "Attachment".
- For fields whose type attribute is not identical to that in KIX17, there is another attribute "MigratedType" in KIX18 with the respective value of the type attribute from KIX17.
- Perl comments in the string of the class definition are lost because the string is converted into a Perl structure and back to a string after the adjustment. The comments are then lost.
- **Note** on corrections for migrations starting from KIX17.19 (or lower) **or** to KIX18 v24 (or lower):
  If no archive versions of the assets are displayed after the migration, this can be corrected by applying the following DB statement to the KIX18 database.

```
UPDATE xml_storage SET xml_type = REPLACE( xml_type,
'ITSM::ConfigItem::Archive:', 'ITSM::ConfigItem::Archive:') WHERE xml_type LIKE
'ITSM::ConfigItem::Archive:%';
```

Config items / assets including versions

- After the migration, all "config items" including versions of the source environment are available as "assets" with versions in the target environment.

- Attribute types that are no longer supported are displayed as text attributes
- Unversioned CI attachments are displayed as the version attribute "CIAttachments" in the latest asset version.
- CI images are not accepted.

CustomerCompany / Organizations

- The "CustomerCompany" object in KIX 17 becomes "Organization" in KIX 18.
- The attributes of the "CustomerCompany" in KIX 17 become the following attributes in KIX 18:

KIX 17 CustomerCompany Attributes	KIX 18 Organization Attributes
CustomerCompanyCity	City
CustomerCompanyComment	Comment
CustomerCompanyCountry	Country
-	ID (will be created upon transfer)
CustomerCompanyName	Name
CustomerID	Number
CustomerCompanyStreet	Street
CustomerCompanyURL	Url
ValidID	ValidID
CustomerCompanyZIP	Zip

Contact person / user

- After the migration, all "CustomerUser" in the source environment are available as "Contacts" ("IsCustomer") in the target environment.
    - Use of SSP is activated,
    - "Customer" role assignment exists
    - PW hash takeover

- After the migration, all "Agent Users" in the source environment are available as "Contacts" and "Users" ("IsCustomer" && "IsAgent") in the target environment.
    - "Customer" role assignment exists
    - "Agent User" role assignment exists
    - Password hash has been adopted
    - If the login name of a KIX 17 user (agent) already exists in KIX 18, he is created as a new user with the suffix "-migrate" (similar to asset classes).
- When migrating a KIX17 environment to KIX18, the system user from KIX17 (UserID 1, usually "root@localhost") is not migrated.
  Instead, all its assignments are transferred to the system user of the KIX18 environment (log-in: "admin"), i.e:
    - Tickets that are assigned to the responsible person or processor "root@localhost" in KIX 17 are assigned to the user "admin" in KIX 18.
    - In the history of tickets or assets, the user "admin" is also referenced instead of "root@localhost".
- The "CustomerUser" object in KIX 17 becomes "User" or "Contact" in KIX 18.
- The attributes of "CustomerUser" in KIX 17 become the following attributes in KIX 18:

KIX 17 CustomerUser Attribute	KIX 18 User/Contact Attribute
-	Contact.AssignedUserID (Will be created during transfer (User.ID))
UserCity	Contact.City
UserComment	Contact.Comment
UserCountry	Contact.Country
UserEmail	Contact.Email
UserFax	Contact.Fax
UserFirstname	Contact.Firstname
-	Contact.Fullname
UserLastname	Contact.Lastname

KIX 17 CustomerUser Attribute	KIX 18 User/Contact Attribute
UserMobile	Contact.Mobile
UserCustomerIDs	Contact.OrganisationIDs (Requires lookups on Organization.ID via Organization.Shortname)
UserPhone	Contact.Phone
UserCustomerID	Contact.PrimaryOrganisationID (Requires lookup on Organization.ID via Organization.Shortname)
UserStreet	Contact.Street
UserTitle	Contact.Title
-	Contact.UserID
ValidID	Contact.ValidID
UserZip	Contact.Zip
-	Contact.ID (Will be created upon transfer)

FAQ including attachments

- After the migration, all FAQ categories of the source environment are available in the target environment.
- If a language is referenced in the source system that does not exist in the target environment, this is set up in the target system.
- The contents of FAQ articles, which are used as a symptom, cause, solution and comment by default in the source system, are legible in the target system.
- Remain:
    - Richtext formatting
    - Inline images

- Attachments to the FAQ article
- The identification as "public" or "external" in the source system leads to the setting of the flag "Show in SSP".
- The FAQ ratings can be omitted.

Dynamic fields

- After migration, the following dynamic field data is available in the target environment:
    - Configurations
    - The values contained in the dynamic fields.
    - Referral data

- Checklists:
    - Dynamic fields of type "Checklist" are mapped in "MobileProcessingChecklist[n]" after migration.
    - Mapping of the values (KIX17 => KIX18):
        - closed => OK
        - notok => NOK
        - pending => pending
        - open => -
    - Only the title and status of the checklist entries are transferred; the description is not transferred.
      The transmission of the statuses is limited to the checklist statuses of KIX 18.

- Dynamic fields of unsupported object types
    - are deactivated in the target system (are set to "invalid")
    - receive the suffix "- DF Type not yet supported!" in the label.
    - The edit dialog opens and displays a message.

- Dynamic fields are mapped as follows (KIX17 => KIX 18):

Field type KIX17	Object in KIX18	Field type in KIX18	Validity after migration
Attachment	Ticket	is not supported	invalid
Captcha	Ticket	will not be migrated!	valid
Checkbox	Ticket	Selection	valid

Field type KIX17	Object in KIX18	Field type in KIX18	Validity after migration
Date	Ticket	Date	valid
DateTime	Ticket	DateTime	valid
Dropdown	Ticket	Selection	valid
DropdownGeneralCatalog	Ticket	Selection	invalid
ITSMConfigItemSelection	Ticket	AssetReference	valid
Multiselect	Ticket	Selection	valid
MultiselectGeneralCatalog	Ticket	Selection	invalid
ObjectReference/ CustomerCompany	Ticket	OrganisationReference	valid
ObjectReference/ CustomerUser	Ticket	ContactReference	valid
ObjectReference/User	Ticket	ContactReference	valid
RemoteDB	Ticket	Selection	invalid (Manual change to DataSource (if necessary also setup DataSource andConnection) required).
RichText	Ticket	is not supported	invalid
SingleSelection	Ticket	Selection	valid

Field type KIX17	Object in KIX18	Field type in KIX18	Validity after migration
Text	Ticket	Text	valid
TextArea	Ticket	TextArea	valid
TextArea/FAQ	FAQ	TextArea	valid
Token	Ticket	Text	invalid

General Catalog:

- After the migration, all values of all General Catalog classes in the source environment are available in the target environment.
- General catalog classes that already exist in the target environment are supplemented by additional selection values from the source environment.
- If entries are received from the source system that already exist, they are left, but the references in the incoming data are "reassigned" accordingly.

Service data

- After the migration, all service entries are available as assets of the "Service" class in the target environment
- The entry "Assigned team/queue" is not accepted.
- Criticality will not be accepted.
- There is no migration of the existing data in the "Service" class.

SLA (KIX Pro):

- After migration, all SLAs of the source environment are present in the target environment.
- The SLA fulfilment times are transferred from the target environment to the source environment. The SLA violation (yes/no) and SLA deviation (duration) information is based on KIX18 calculations, starting from the creation/fulfilment time.
- SLA start time calculations based on a dynamic field (DateTime) are **not considered**, the creation time of the ticket applies.
- The SLA properties "Type" and "Update Time" are omitted.
- If a calendar is referenced in the source system that does not exist in the target environment, the target system uses the default calendar.

4.5.7 Perform the Migration

4.5.7.1 1. Activate PSK mode in KIX 17

KIX 17 and KIX 18 support a PSK mode (PSK = pre-shared key). The PSK is a verification key to secure data transmission. It prevents unauthorized access to the database. Migration only takes place if both systems know this previously agreed key. KIX17 rejects requests if no or an incorrect PSK is transmitted.

The PSK is stored in the KIX 17 configuration file (Config.pm). In KIX 18, the PSK is given to the console command as a parameter (e.g. "PSK = my_PSK_Password").

> 🔴 **Attention**
>
> If the PSK mode is activated, the system is open for external access. Unauthorized third parties can gain access to the database if they know the key. The conventions for assigning a secure password are therefore binding for the naming of the key!
> After completing the migration, delete the key in both systems and deactivate the PSK mode to prevent external access!
> Use a PSK that does not contain any characters to be URL-encoded (https://www.w3schools.com/tags/ref_urlencode.asp) or quote yourself in step 2 if such characters are used.

To enable PSK mode:

1. Open the configuration file in the KIX 17 file system. You can find it under: /opt/kix/Kernel/Config.pm.
2. Find the "insert your own config settings" here "section.
3. Add the following lines of code within the section. Remove the comments ("// PSK [...]"). Assign your own secure password instead of "my_PSK_Password" and save your changes.

```
# Activate PSK-Modus
$Self->{'Migration::Active'} = 1;

# Name PSK
$Self->{'Migration::PSK'} = 'my_PSK_Password';
```

```
# The database DSN
$Self->{DatabaseDSN} = "DBI:Pg:dbname=$Self->{Database};host=$Self->{DatabaseHost};";

# ----------------------------------------------------- #
# insert your own config settings "here"                #
# config settings taken from Kernel/Config/Defaults.pm  #
# ----------------------------------------------------- #
$Self->{CheckMXRecord} = 0;
$Self->{FQDN} = 'kalliope.';
$Self->{'Migration::Active'} = 1;
$Self->{'Migration::PSK'} = 'my_PSK_Password';

# ----------------------------------------------------- #
# data inserted by installer                            #
# ----------------------------------------------------- #
# $DIBIS
```

4.5.7.2   2. Start the migration via the KIX 18 console

1. In the admin module of KIX 18, navigate to *System> Console.*
2. In the selection field select the command
    `Console::Command::Admin::Installation::Migrate::Run`
3. Enter the transmission parameters (see table below) in the command line.
    Use the following scheme:

Command line scheme

```
--source ... --source-id ... [--options ...] [--type ...] [--mapping-file ...]
[--filter ...(+)] [--workers ...]
```

Examples

```
1. --source KIX17 --source-id mykix17 --options URL=http://myKIX17domain:Port/
kix/migration.pl,PSK=my_PSK_Password --type ticket
2. --source KIX17 --source-id mykix17 --options URL=http://172.17.0.1:9990/kix/
migration.pl,AutoDeps=1,PSK=test123
```


Fig.: Example migration command

4. Start the migration by clicking on "Execute". The migration can take some time.
5. The console will inform you of the migrated data when it is complete.

> ✅ **Tip**
>
> You can get help on the console commands by entering `--help` or clicking on ⑦

Possible parameters:

Use the following parameters with 2 leading hyphens (e.g. `--source`). Parameters marked with a * are mandatory.

Parameter	Description
source*	The type of source system from which the data is to be migrated (e.g. `--source KIX17`). This must be a source supported by KIX 18.
source-id*	Identifier for the concrete source system (e.g. `--source-id myKIX17`). In KIX18 it is possible to migrate from several sources. KIX18 uses the source-id to distinguish from which system it has migrated which data. ⓘ **Please note:** If you want to perform a migration in the menu *System > Migration KIX 17* (KIX Pro) and then clean up any errors via the console, please note: • When migrating via the console, the source-id is freely selectable. However, it must be identical for further partial migrations. • When migrating via the GUI (KIX Pro), the source-id is the same as the URL entered in the GUI, e.g. http: `//myKIX17domain.de:PORT/kix/migration.p`l

Parameter	Description

options*	Options for migration, e.g. server/host of the source system and PSK:
	
	`.--options URL=`http://172.17.0.1:9990/kix/
	migration.pl,AutoDeps=1,PSK=test123
	
	• Separate multiple options with commas.
	• Calling the console command "`Admin::Installation::Migrate::ListSources`" shows possible options.
	• 🛈 The details of the source (URL) and the PSK are mandatory.
	
	Possible Options:
	
	• **URL:** URL of the file "migration.pl" (e.g. `http://192.168.160.200:4711/kix/index.pl` or `http://myKIX17domain.de/kix/index.pl`).
	    • You can find the URL in the address line in the browser of your open KIX17 system
	    • Replace "`index.pl`" with "`migration.pl`" in the command line.
	• **PSK**: Enter the key specified in KIX17 (e.g. `PSK = my_PSK_Password`)
	• **AutoDeps:** Enables/disables the automatic migration of dependent data.
	    • `AutoDeps=1`: Dependent data is also migrated.
	    If, for example, you only want to migrate queues/teams, the corresponding user and system address entries are also migrated. ⚠️ Use "AutoDeps=1" only with caution, as duplicate entries will result if the "source-id" is incorrect.
	    • `AutoDeps=0`: Dependent data will not be migrated.
	    If, for example, you only want to migrate queues/teams, the corresponding entries of user and system address will not be obtained (useful if users and system addresses have already been obtained in a previous run).

Parameter	Description

type	By specifying this parameter, you can restrict the migration to selected objects (tickets, organisations, assets, etc.). Separate multiple types with commas (e.g. `--type ticket,customer_company,configitem`).
	
	If no type is specified, all supported objects are migrated.
	
	Usable types are (note dependencies among each other!):
	
	  • configitem
	  • configitem_definition
	  • configitem_history_type
	  • customer_company
	  • customer_user
	  • dynamic_field
	  • dynamic_field_value
	  • faq_category
	  • faq_item
	  • general_catalog
	  • general_catalog_preferences
	  • kix_text_module
	  • link_object
	  • link_relation
	  • link_type
	  • queue
	  • service*
	  • sla*
	  • sla_preferences*
	  • system_address
	  • ticket
	  • ticket_history_type
	  • ticket_priority
	  • ticket_state
	  • ticket_type
	  • users
	
	*The types marked with an asterisk are only available in KIX Pro.
	
	Tip: The console (see page 271) command "
	
	`Console::Command::Admin::Installation::Migrate::ListTypes --source KIX17` " lists all available types.

Parameter	Description
mapping-file	The JSON file to use for mappings (use an existing object instead of creating a new one). The most important mappings (e.g. priority, status, etc.) are already available internally.
filter	One or more filter expressions. Format and content depend on the source selected. An SQL WHERE clause can be specified for each object type. Format: `--filter "ticket{user_id = 123 AND responsible_id in [1,2,3]}"`
workers	Number of parallel processes to be used for the bulk objects such as tickets etc. Standard: 1 With the number of workers you determine the available system performance. This has an impact on the performance of your server. The higher the number of workers, the faster the migration process if the system is heavily loaded. **Avoid overloading your system!** ℹ Already 4 workers can utilise a host to 100% capacity. Therefore, KIX.Cloud environments are limited to a maximum of 2 workers. If this parameter is not specified, the default value 1 is used.
help	Show help for this command.
quiet	Suppresses the display of general information. Only error messages are displayed in the console.

4.5.7.3   3. Deactivate pre-shared mode

When the PSK mode is activated, **unauthorized third parties** can gain access to the database as soon as they gain possession of the key. It is therefore imperative to prevent access to the database again after the migration has been completed (or canceled)!

To do this, delete the key and deactivate the pre-shared mode as follows:

1. Open the configuration file in the KIX 17 file system. You can find it under: `/opt/kix/Kernel/Config.pm.`
2. Find the "`insert your own config settings" here `" section.
3. Within the section, delete the lines of code you inserted and save this change.

4.5.7.4   4. Check and edit the transferred data

The migrated data are marked with a migration prefix if an object with the same name already exists in the target system, e. g. Class "**Migration** Computer, **Migration** User Name".

Check the transferred data, placeholders, signatures etc. and adjust them if necessary. You can edit these individually manually or automatically by using a job.

> ⚠ **Note**
>
> After the data has been transferred, it is available. However, building the indices for full text searches or number of tickets per team, assets per class or similar can take some time. Depending on the volume of data (several 100,000 tickets), this can also take several hours.

4.5.7.5   5. Post-processing and cache cleaning

> ⚠ **Note**
>
> After completing the migration, a cache clean-up must be performed on the KIX18 environment to avoid inconsistent data representation. To do this, execute the command "`Maint::Cache::Delete `" in the *System > Console* menu, log out of the GUI and log back in.

4.5.7.6   6. Advanced configuration transmission

A tool is provided for the transfer of further configurations such as ticket or response templates, "generic agents", etc. It is continuously being developed and can be obtained from the forum: . It is being continuously developed and can be obtained from the forum: https://forum.kixdesk.com/index.php?topic=11882.0.

4.6   Backup and restore

4.6.1   Securing on-premises installation

To secure an on-premises installation, we recommend taking advantage of containerization and keeping it simple. Accordingly, the backup of the contents of the Docker volumes represents the backup of your KIX.

1. Stop Docker services: see https://github.com/kix-service-software/kix-on-premise/tree/master/deploy/linux#readme

2. Back up Docker volumes: on the Docker host, under Ubuntu (volumes in / var / lib / docker / volumes /)

Backup of KIX Docker volumes

```
shell> tar cfvz /tmp/KIXDockerVolumes.$(date "+%Y-%m-%dT:%H:%M:%S").tar.gz /var/lib/
docker/volumes/kix_backend/ /var/lib/docker/volumes/kix_frontend/ /var/lib/docker/
volumes/kix_db/ /var/lib/docker/volumes/kix_shared/
```

---

3. Restart Docker services: see https://github.com/kix-service-software/kix-on-premise/tree/master/deploy/linux#readme

4.6.2   Backup of the database

In addition to the Docker volumes, you can also back up the KIX database. You can use "pg_dump" of the DB container for this. Attachments and content of articles are not saved!

Backup of the KIX database

```
shell> docker exec kix_db_1 pg_dump -U kix kix >> /tmp/KIXDB.$(date "+%Y-%m-%dT:%H:
%M:%S").sql
```

4.6.3   Recovery on-premises installation

Installation according to https://github.com/kix-service-software/kix-on-premise/tree/master/deploy/linux#readme

1. Stop Docker services

2. Replace Docker volumes with backup (on the Docker host under Ubuntu e.g. in: / var / lib / docker / volumes /)
3. Restart Docker services

4.6.4 Backup KIX Cloud

Backups for KIX Cloud take place regularly and automatically.

A backup of the KIX Cloud can be made available for download on request. As part of their support contract, users of the KIX Cloud can contact support@kixdesk.com[4] if a backup is required or to have the system reset.

---

4 mailto:support@kixdesk.com

4.7   Setting up a reference environment (on premises)

For an on-premises installation, it can be helpful to set up a reference environment, for example to prepare or test configurations or to check updates before going live.

The recommended procedure is based on backing up the existing KIX installation and restoring it on another Docker host.

The procedure is described step-by-step below. "Source" is the original environment. The "target" is the additional Docker host or the copy of the original environment.

Step	Environment	Description
1	Source	Deactivation of the Email collection (important!)
2	Source	Create the backup according to the instructions in Backup and Restore (see page 52)
3	Source	Reactivate the Email collection
4	Aim	Transfer of the backup to the host for the reference environment, e.g. using rsync[5] or scp[6]
5	Aim	Basic installation of a KIX18 with your registry ID (Installation - KIX On Premises) (see page 26)
6	Aim	Restore Backup according to the instructions in Backup and Restore (see page 52)

---

5 http://manpages.ubuntu.com/manpages/jammy/en/man1/rsync.1.html

6 http://manpages.ubuntu.com/manpages/trusty/man1/scp.1.html

Step	Environment	Description
7	Aim	Configuration of the reference environment: • Deactivation of the Email dispatch (Outbox (see page 219) , recommended for test environments) or adaptation of the sender address (see page 228) • Setup of reference/test Email accounts (Inbox (see page 201)) • Adjusting the front-end, the back-end such as the SSP host names (Setup assistant - step 1 (see page 58))

5 Initial Setup

When you log in for the first time, KIX 18 opens with the Setup Assistant , which guides you through the most important steps of the initial setup. The initial setup includes creating a Super User, entering information about your company and configuring the incoming and outgoing mail. You can skip individual installation steps and continue later. Until the initial setup is complete, the Setup Assistant will open every time you start KIX 18.
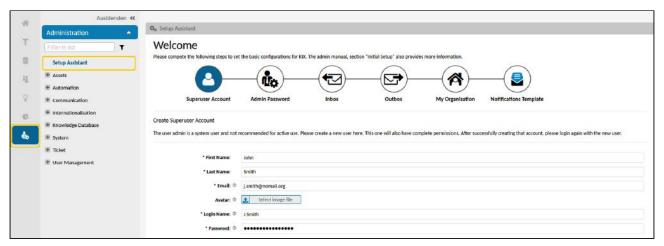
Fig.: The Setup Assistent in Admin Module

> ⬥ **Attention!**
>
> - The initial user for the first login works in the system as a root user with complete authorization to all system areas. In order not to endanger the security of the system, you **must** create a super user in the setup assistant and change the password for the initial user (steps 1 and 2).
> - Do not delete the initial user, basic system functions are bound to it! Otherwise you would lock yourself out of the system.

After the initial setup has been completed, KIX starts with the welcome page. You can open this page at any time by clicking on the question mark icon in the header.

The welcome page contains current information on KIX 18 and links to the KIX manuals and the release information in the forum.
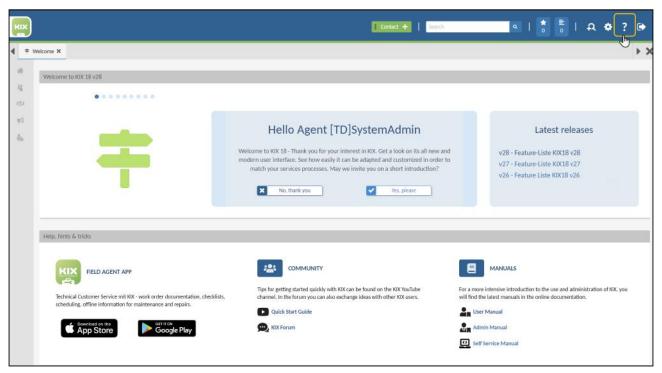
Fig.: The welcome page of KIX 18

The further setup of the system is done directly in the Admin Module, which is only available to users with admin rights. You will find the Admin Module in the module menu on the side .
You can navigate to the individual administration areas via the explorer in the Admin Module.
There you will also find the Setup Assistant.

5.1   Setup Assistant

When you log into KIX for the first time, KIX opens with the setup assitant. This will guide you through the initial setup in just a few steps.

You can switch between the individual steps, skip individual steps and interrupt the initial setup at any time to continue at a later point in time. Your previous entries will be retained. KIX therefore always starts with the setup assitant until the initial setup is complete if the user logs in as the "admin" user or with the "superuser" role.

Content on this page:

Depending on your user rights, you can also open the setup assitant at any time via the Explorer in the admin dashboard.
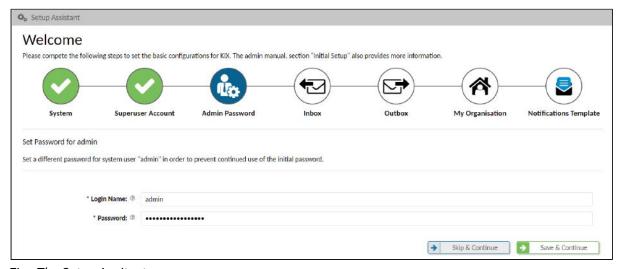
Fig.: The Setup Assitant

5.1.1   Overview

The setup assitant guides you step-by-step through the initial setup.

Step	Icon	Description
1		**System:** Deposite of the FQDN
2		**Superuser Account:** Create a superuser
3		**Admin Password:** Change the access data for the initial user
4		**Inbox:** Set up an email account for the email inbox
5		**Outbox:** Setting up an outgoing mail server for sending emails
6		**My Organisation:** Store company reference data
7		**Notifications Template:** Customization of the HTML template for notifications

To switch between the individual steps, click on the icons in the assitant. The color of the icons indicates the progress in the assitant:

Icon	Description
	White Icon: Step is still unprocessed
	Blue Icon: currently open step
	Yellow check mark: step is incomplete
	Green check mark: step is complete

There are the following buttons in the setup assitant:

Button	Description
Skip & Continue	Skip this step and skip to the next open step.
Save & Logout	Saves the user as a new contact and logs the user out after completion (only step "Superuser Account").
Save & Retrieve	Saves the configuration of the email account and retrieves the inbox from the server ("Inbox" step only).
Save & Continue	Saves the user as a new contact and jumps to the next open step.
Preview	Shows a preview of the notification template (only KIX Pro).

5.1.2 Preparation

When setting up for the first time, you create a superuser, change the access data for the initial user, configure incoming and outgoing mail and store the data on your organization (company). Have the following information ready:

- First name, last name, email address of the user "Superuser"
- The URLs on which your KIX system is located and via which it can be reached.
- Credentials for the incoming mail server
    - Login name of the email account
    - Password of the email account
    - Pick-up point of the email server (host: port)
    - Pick-up method with or without encryption (e.g. "IMAP" or "IMAPTLS")
    - with IMAP: Directory in which the incoming emails are located (usually "INBOX")
- Credentials for the outgoing mail server
    - Address to the SMTP interface of the email server (e.g. "smtp.mailbox.org")
    - Number of the port over which the emails are sent
    - Username (login of the email account)
    - Password of the email account
- Reference data of your company, such as address and contact details.

If you use the mail server of a service provider (GMail, IONOS, Strato etc.), you will receive the required login information from them. If you use your own mail server, you should know the login information.
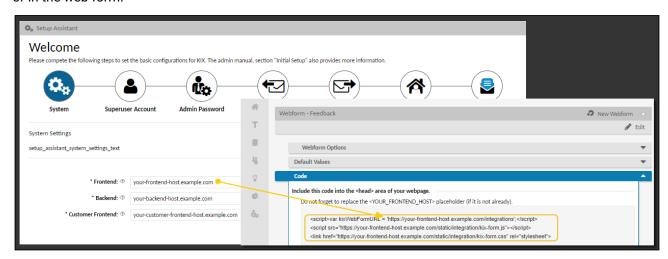
5.1.3 Use the setup assistant
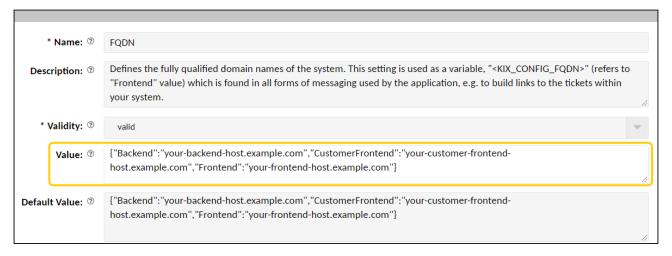
5.1.3.1 Step1: System

In this step you store the FQDN (Full Qualified Domain Name) for your KIX system. These details are set in the SysConfig key "*FQDN*" and thus made available to the various mechanisms in KIX, e.g. in OAuth2 profiles or in the web form.

Replace the initial example URLs for the KIX Front- and Backend with the URLs via which your KIX system can be reached. If you are working with KIX Pro, also enter the URL for the Self Service Portal under "Customer Frontend".

Alternatively, you can set the FQDN in the SysConfig key "*FQDN*":
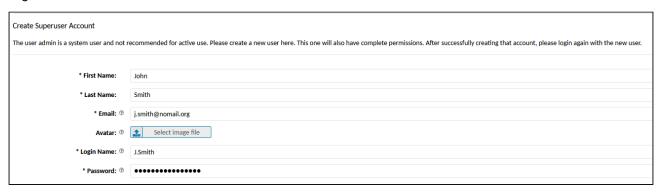
5.1.3.2 Step 2: Superuser Account

In this step you create a user with the role "Super User". The superuser is a system user at user level with rights to the entire system. His rights are more extensive than those of the administrator ("System Admin" role). This superuser is created in the User administration> User menu as a user with the data stored here. It is automatically assigned to the "My Organization (MY_ORGA)" organization. You can change this data at any time in the user administration. Then log in again as a user with the superuser. To do this, click the "Save & Logout" button.

Create Superuser Account

The user admin is a system user and not recommended for active use. Please create a new user here. This one will also have complete permissions. After successfully creating that account, please login again with the new user.

* First Name:	John
* Last Name:	Smith
* Email: ⓘ	j.smith@nomail.org
Avatar: ⓘ	⬆ Select image file
* Login Name: ⓘ	J.Smith
* Password: ⓘ	●●●●●●●●●●●●●●
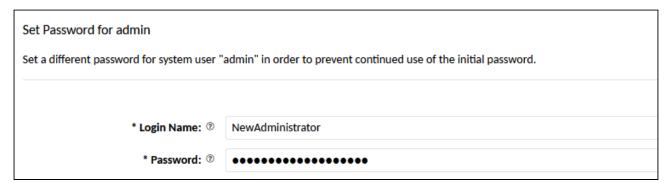
> ⚠ **Hint**
>
> The initial user "admin" works in the system as a root user with full authorization down to the lowest system level. It is not intended to be used in the system! For the security of the system it is therefore necessary that you create a user with the role "Super User".

5.1.3.3 Step 3: Admin password

In this step you define a new password for the initial user.

With the provision of KIX you will also receive the password for the initial user "admin". To protect the system from unauthorized access by third parties, you must replace the username and password. The admin password can only be changed using the setup assistant, as the initial user is below the user level.

Set Password for admin

Set a different password for system user "admin" in order to prevent continued use of the initial password.

| * Login Name: ⓘ | NewAdministrator |
| * Password: ⓘ | ●●●●●●●●●●●●●●●●●●●● |

> ⬣ **Important!**
>
> Use a secure password with upper and lower case letters, special characters and numbers. Keep the password safe and secure to prevent unauthorised access by third parties. If the password is forgotten, you can **lock** yourself out of the system.

5.1.3.4   Step 4: Inbox

In this step you will set up an email account for the inbox.

KIX polls the inbox of your email server at regular intervals and generates new tickets from the incoming emails. To do this, you need to specify an email account from which incoming emails will be retrieved. The information required for this can be obtained from your web hoster or is available to you if you operate your own mail server.

After you have entered the information, you can test the email reception. Click Save & Retrieve. If the delivery fails, you will receive a notification message.
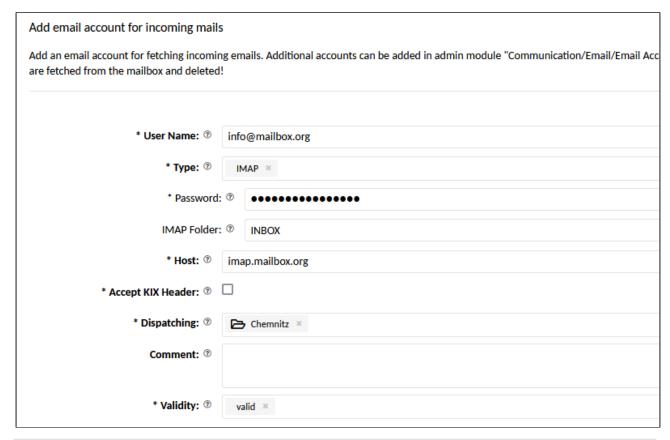
You can set up and edit additional email accounts in the *Communication> Email> Email Accounts* menu.

> ✅ **Tip**
>
> You can skip this step if you, as a prospect, just want to get an overview of KIX.

Add email account for incoming mails

Add an email account for fetching incoming emails. Additional accounts can be added in admin module "Communication/Email/Email Acc
are fetched from the mailbox and deleted!

* User Name: ⑦	info@mailbox.org
* Type: ⑦	IMAP ✕
* Password: ⑦	●●●●●●●●●●●●●●●●●
IMAP Folder: ⑦	INBOX
* Host: ⑦	imap.mailbox.org
* Accept KIX Header: ⑦	☐
* Dispatching: ⑦	📂 Chemnitz ✕
Comment: ⑦	
* Validity: ⑦	valid ✕

Field	Description
Username	Login of the email account whose inbox you want to access. For example: info@mailbox.org
Password	Password of the email account
Host	Address for the collection interface of the incoming mail server Usually there is a separate host URL for each pickup method: e.g.: pop.mailbox.org or imap.mailbox.org A special port can also be specified (host: port)

Field	Description
Type	Transmission protocol for email retrieval. The pick-up method is specified with the selected protocol. If necessary, find out about the differences between POP3 and IMAP beforehand. Supported protocols: • without encryption: IMAP or POP3 • with SSL encryption: IMAPS or POP3S • with TLS encryption: IMAPTLS or POP3TLS Recommendation: First test IMAP or POP3 without encryption. If the wrong option is chosen, the authentication will fail. A corresponding notification is issued.
IMAP-Folder	Directory in which the incoming emails are on the server. KIX picks up the inbox from this directory (usually the "INBOX" or "Inbox" folder). This option is only available if the retrieval is from an IMAP account.
Allow KIX Headers	**When activated,** information from the KIX email headers is applied (e.g. ticket type, team, priority, etc.). The email header is accepted, evaluated and the information it contains processed, regardless of what is configured in the email account. **If not activated**: The email header is discarded when the emails are received and is ignored. ⚠ **Please note:** The emails sent via KIX can be provided with a KIX header. You can read these out if necessary and use them for actions based on them. But: Under certain circumstances, the header information supplied by KIX may contradict the email handling you have specified and lead to undesired actions or system errors! **Therefore, only activate this option if you trust the KIX header (e.g. from other KIX systems) and know from which senders the retrieved messages originate.**

Field	Description

Dispatching	Select the principle according to which the retrieved emails are to be distributed:
	
	• **Default Queue (SysConfig)**
	    • The retrieved emails are distributed to the default team. The default team is the team that is stored in the SysConfig key "PostmasterDefaultQueue". In the delivery state, this is the "Service Desk" team. You can specify a different team in the SysConfig key.
	    • If emails cannot be assigned to a team, the default team serves as a fallback.
	• **Recipient addresses (To, CC, etc.)**
	    • The distribution of retrieved emails is based on the email recipient.
	        • If the recipient address is created as an email address in the system, the ticket will be created in the team that is stored at this system address.
	        • If the address is unknown or the email cannot be assigned to a team, the ticket will be created in the standard team (see above).
	• **Team**
	    • If this option is selected, you can specify a team in which the collected emails will be created as new tickets.
	    • All teams created in the system are available for selection. Teams in light grey are (temporarily) invalid. They are listed but cannot be selected.
	
	For further information on distribution, see Notes on email distribution. (see page 207)

Comment	You can save your own notes.
Validity	**valid:** The emails are retrieved and distributed to the assigned team. **Invalid / temporarily invalid:** The inbox of the email account is not (no longer) retrieved, so that no (further) emails are received via it. Emails that have already been accessed are stored in the system as tickets and can be processed.

5.1.3.5 Step 5: Outbox
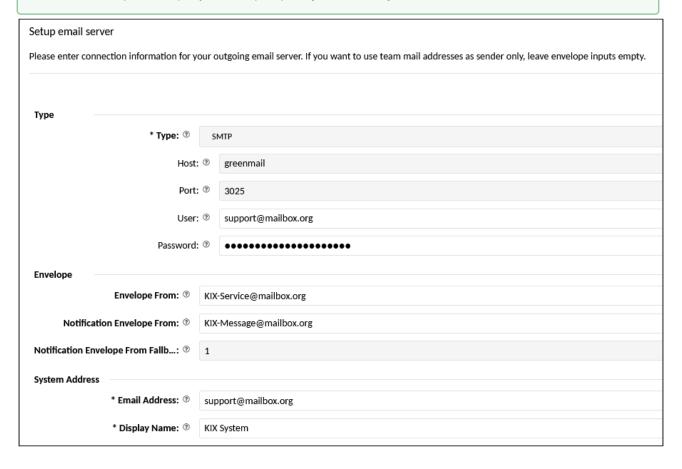
In this step you set up an outgoing mail server (Sendmail) and the standard email address for the outgoing mail.

In order to be able to send emails from KIX, you must specify a mail server and a standard sender address for the outgoing mail. You can obtain the required server information from your web hoster or are available to you if you operate your own mail server.

Depending on your system rights, you can change this information at any time in the *Communication> Email> Outbox* menu and in the *Communication> Email> Email Addresses* menu.

> ✅ **Tip**
>
> You can skip this step if you, as a prospect, just want to get an overview of KIX.

Setup email server

Please enter connection information for your outgoing email server. If you want to use team mail addresses as sender only, leave envelope inputs empty.

Type

* Type: ⑦	SMTP
Host: ⑦	greenmail
Port: ⑦	3025
User: ⑦	support@mailbox.org
Password: ⑦	●●●●●●●●●●●●●●●●●●●

Envelope

Envelope From: ⑦	KIX-Service@mailbox.org
Notification Envelope From: ⑦	KIX-Message@mailbox.org
Notification Envelope From Fallb...: ⑦	1

System Address

* Email Address: ⑦	support@mailbox.org
* Display Name: ⑦	KIX System

Field	Description
Type	Transmission protocol for sending emails (SMTP). With the selected protocol you determine the transmission method. If necessary, you can obtain detailed information from your web host. Supported protocols: • without encryption: SMTP • with SSL encryption: SMTPS • with TLS encryption: SMTPTLS **Recommendation:** First test SMTP without encryption. **Note:** The "DoNotSendEmail" option does not send emails, it can be used for test systems.
Host	Address to the SMTP interface of the outgoing mail server, e.g. smtp.mailbox.org
Port	Port via which the emails with the previously selected "Type" are sent. The port depends on the selected transmission protocol: • SMTP: 25 • SMTPS: 465 • SMTPTLS: 587
User	Username for logging into the outgoing mail server. This often corresponds to the name of the email account. For example: johndoe@example.de This value can also be cleared via port 25.
Password	The password of the email account. This value can also be cleared via port 25.

Field	Description
Envelope	You can put your outgoing emails in an "envelope" (optional). Only what is written on the "envelope" can then be recognized by the mail transport.

Envelope from: You can specify a sender that should appear on the "envelope", e.g. sender@example.com. If no sender is specified, the team's email address is used as the sender.

Notification Envelope from: You can enter an email address that will be used as the sender header in outgoing notifications. If no address is given, the header of the "envelope" is empty.
(Corresponds to the SysConfig key "NotificationSenderName (see page 193) ")

Notification Envelope from Fallback: If no "*SendmailNotificationEnvelopeFrom*" is specified, this setting allows you to use the sender address instead of a blank envelope sender (required in certain mail server configurations). |
| System address | KIX requires at least one valid email address for sending and is therefore delivered with an initial standard address (kix@localhost). Replace this with your standard email address for the outgoing mail (e.g. mail out@my-company.com) and enter a sender ID for this email address.

KIX uses this address to send all outgoing emails if no further email addresses are created or if sending via another address fails. Alternatively, the sender address can also be changed in the *Communication> Email> Email Addresses* menu.

The sender address must match an email account configured in KIX (see above step 3 "Inbox" or menu *Communication > Email > Email accounts*). |
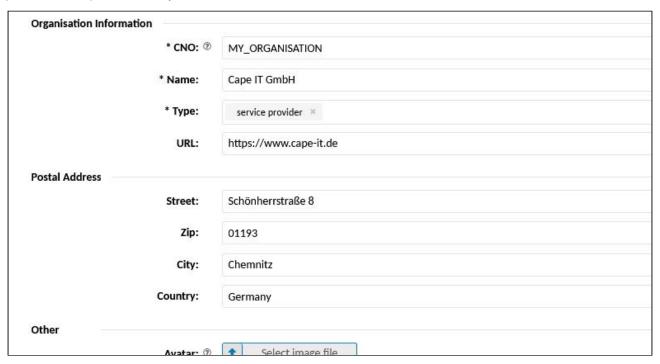
5.1.3.6   Step 6: Own Company

In this step you enter the reference data of your company.

For internal and external communication, KIX needs the master data for your company. This data is stored in the system under the initial organization "MY_ORGA" or under the customer number you specified here. You can change this company data at any time in the "Organizations" module.

You can optionally store additional company data in the configuration keys (SysConfig keys). These can be used as a reference in various places. For example in text modules and email signatures or for dynamic placeholder replacement in jobs and notifications.

Field	Description
Customer Informations	**KNR:** The customer number or name under which KIX should manage your company in the system. Enter a unique identifier. Numbers and letters are allowed.
	Name: The name of your company.
	Type: The company type under which your company is to be managed in the system (default: Service Provider).
	URL: The URL of your company website.

Field	Description
Adress	The postal address of your company headquarters.
Avatar	The avatar is an image file for symbolically identifying your company, e.g. in the sidebar. You can optionally upload your own image file as an avatar.
Comment	You can save individual notes.
Validity	When you save your company data, the setup assistant creates a new organization with the ID specified under KNR. You can set this organization to either valid or (temporarily) invalid.

valid: The organization can only be used in the system if it is set to "valid".

Invalid / temporarily invalid: The organization cannot be used by the system, e.g. for test purposes. |

Field	Description
SysConfig Keys	The information stored here is used in the team signature that is initially set. This information can also be used in KIX placeholders. The setup assistant saves this information in the SysConfig keys of the system configuration. Alternatively, you can configure the individual keys in the *System> Sysconfig* menu.
	Organization: The name of your company for use in the KIX mail header.
	OrganizationAddress: Street and house number of your company for use in placeholders.
	OrganizationDirectors: The registered director of your company. For use in placeholders, e.g. in the email imprint
	OrganizationHotline1: Your company's phone number. For use in placeholders, e.g. in email signatures
	OrganizationHotline2: Your company's fax number. For use in placeholders, e.g. in email signatures
	OrganizationLong: The name of your company for use in wildcards.
	OrganizationRegistrationLocation: The place where your company is registered according to the legal requirements. For use in placeholders.
	OrganizationRegistrationNumber: Your company's registration number, e.g. Commercial register number. For use in placeholders.

5.1.3.7   Step 7: Notification Layout

KIX can send automated notifications for certain ticket events so that agents are informed about these ticket events (see chapter Notifications (see page 184)).

In this step you can individually configure the standard layout provided by KIX for these notifications and adapt it to your corporate identity. To do this, you need knowledge of HTML and CSS. You will find good help at https://www.selfhtml.org (in German). Our support team will be happy to help you design the notification layout.

You can also change the template at any time in the *Automation> Notification Layout* menu.

Fig .: Example of an HTML email

The HTML source code displayed in the setup assitant shows the layout of the standard template. You can change the text and style information to customize the layout. For easier editing, you can copy the source code into an HTML editor (e.g. Notepad ++).

- Contents between `<style>` and `</style>` define the formatting of the individual elements on the page, such as page background, header and footer of the mail, display of headings and links, etc.
- Contents between `<body>` and `</body>` form the page content. The page content is grouped in individual containers. These containers are formed with `<div> [...] </div>`.
- Contents between `<!` - and `->` are comments. These notes are for your information and are not taken into account in the email display. If you want, you can safely remove them.
- You should not change the lines `[% Data.Subject | html%]` and `[% Data.Body [....] %]`. These are the HTML placeholders for the email subject and the content of the email.

Click on "Preview" to check the configuration you have changed before saving.

5.1.4 Reset the setup assistant or change the status

If necessary, you can reset or change the status of the setup assitant to the initial state. This is done in the SysConfig key "*SetupAssistantState*".

> ⬣ **Warning**
>
> Changes to the system configuration can lead to serious damage to the system and should only be made by experienced KIX users or our support team!

The status can be set separately for each individual step in the setup wizard. You can thus explicitly reset each step to its

- reset to its delivery status, e.g. to enter the information again
- set to "completed", e.g. if the information required in the step has already been stored elsewhere in the system (e.g. if the FQDN to be entered in step 1 has already been stored in the SysConfig key "*FQDN*")
- set to "skipped", e.g. if the information required in the step has not yet been entered completely or the initial data does not need to be changed (e.g. if "admin" is sufficient as super user and no other super user is to be set).

Parameters and states of steps

Parameter	Description	Possible values
stepID	ID of the step (unique identifier)	• System: setup-system-settings • Superuser Account: SuperUserAccount • Admin Password: Mail • Inbox: MailAccount • Outbox: setup-sending-email • My Organisation: SetupMyOrganisation • Notifications Template: setup-notification-template

Parameter	Description	Possible values
completed	Corresponds to a click on the button "Save & Continue". If true, this step is considered completed.	`true` | `false`
skipped	Corresponds to a click on the button "Skip & Continue". If true, this step is considered "skipped/not completed". It can be edited again at a later time.	`true` | `false`
result	Result of the setup step Used for loading the values when the step is opened again (e.g. "My Organisation" step). The extent to which the specified values are evaluated depends on the respective steps (code).	`null` or individual data structure, e.g. `{"OrganisationID":1}`

Basic configuration of the Setup Assistant

The following sample configuration contains all configuration parameters of the Setup Assistant and sets all steps in the Setup Assistant to "unedited".

Example 1 - Basic configuration

```
[
  {
    "stepId":"setup-system-settings",
    "completed":false,
    "skipped":false,
    "result": null
  },
  {
    "stepId": "SuperUserAccount",
    "completed": false,
    "skipped": false,
    "result": null
  },
  {
```

```json
            "stepId": "Mail",
            "completed": false,
            "skipped": false,
            "result": null
        },
        {
            "stepId": "MailAccount",
            "completed": false,
            "skipped": false,
            "result": null
        },
        {
            "stepId": "setup-sending-email",
            "completed": false,
            "skipped": false,
            "result": null
        },
        {
            "stepId": "SetupMyOrganisation",
            "completed": false,
            "skipped": false,
            "result": null
        },
        {
            "stepId":"setup-notification-template",
            "completed":false,
            "skipped":false,
            "result":null
        }
    ]
```

Reset the status of the assistant:

1. Navigate to the *System> SysConfig* menu.
2. Find the key "*SetupAssistantState*" and open it.
3. Click "Reset to Defaults," then click Save.

The status of the Setup Assistant has now been reset to the delivery status. The delivery status is an empty value.

Change the status of the assistant:

1. Navigate to the *System> SysConfig* menu.
2. Find the key "*SetupAssistantState*" and open it.
3. Copy either the value of the SysConfig key or the code example 1 (partially or completely) into a JSON editor (e.g. jsonformatter.io[7]).
4. Set the parameters `completed` and/or `skipped` to `true` or `false` as required.

---

7 http://jsonformatter.io

5. If necessary, set the parameter `result` to `null` or store a data structure for values there (see code example 2).
6. Minimize the JSON string by removing double spaces and line breaks.
7. Copy the minimized JSON string to the clipboard.
8. Switch back to the configuration key and replace the value with the JSON string from the clipboard.
9. Save your change.

The status of the Setup Assistant now corresponds to your manual configuration.

Deactivate Setup Assistant:

If KIX has already been set up, the Setup Assistant will still be displayed when a user with superuser role logs in until it has been "clicked through" once. To disable this, proceed as follows:

1. Navigate to the *System > SysConfig* menu.
2. Find the key "*SetupAssistantState*" and open it.
3. Copy the code example 1 or 2 into a JSON editor (e.g. jsonformatter.io[8]).
4. Set the parameter `completed` to `true` for all steps.
5. If necessary, set the parameter `result` to `null` or store a data structure for values there (see code example 2).
6. Minimise the JSON string to remove double spaces and line breaks.
7. Copy the minimised JSON string to the clipboard.
8. Switch back to the configuration key and replace the value with the JSON string from the clipboard.
9. Save your change.

The individual steps in the Setup Assistant are now considered to have been completed, so the Setup Assistant will no longer be displayed from now on. It can be called up again at any time via the Admin menu.

Note: The Setup Assistant will be displayed again after an update if additional steps have been added. This ensures that all required information can be entered in the Setup Assistant. You can deactivate the Setup Assistant again if you also set the completed parameter to true for the new steps.

Example 2 - Configuration with values

```
[
  {
    "stepId": "setup-system-settings",
    "completed": true,
    "skipped": false,
    "result": null
  },
```

---

8 http://jsonformatter.io

```json
    {
        "stepId": "SuperUserAccount",
        "completed": true,
        "skipped": false,
        "result": {
            "contactId": 999,
            "userId": 999
        }
    },
    {
        "stepId": "Mail",
        "completed": false,
        "skipped": true,
        "result": null
    },
    {
        "stepId": "MailAccount",
        "completed": true,
        "skipped": false,
        "result": {
            "accountId": 2
        }
    },
    {
        "stepId": "setup-sending-email",
        "completed": true,
        "skipped": false,
        "result": {
            "systemAddressId": 1
        }
    },
    {
        "stepId": "SetupMyOrganisation",
        "completed": true,
        "skipped": false,
        "result": {
            "organisationId": 1
        }
    },
    {
        "stepId": "setup-notification-template",
        "completed": false,
        "skipped": true,
        "result": null
    }
]
```

5.2 Guide

After the initial setup using the Setup Assistant, you can further customize KIX to your specific requirements in the Admin module. Due to dependencies in the system, we recommend that you do this in the following order:

Required initial setup (alternative to Setup assistant):

Sequence	What do I have to do?	Where do I configure this?	Which chapter in the manual?
1a	Set up the user "Superuser"	Admin module > User Management > Users Required role: Super User	• Creating and Editing a User and Assigning Roles (see page 664)
1b	Change the Admin Password	in the Setup assistant	• Setup Assistant (see page 58)
1c	Set up Inbox	Admin module > Communication > Email > Inbox	• Inbox (see page 201)
1d	Set up the Outbox	Admin module > Communication > Email > Outbox alternatively: Admin module > System > SysConfig	• Outbox (see page 219) • Configuring the Email Dispatch (SysConfig) (see page 222)
1e	Set up the own organization	Organisation module > Customer "My_Orga"	In the user manual

Further system setup in recommended order:

Sequence	What do I need to do? (mandatory)	Where do I do that?	Which chapter in the manual?
1	Set up agents and assign roles (see note below)	Admin module > User Management > Users	• Users (see page 661) et seqq. • Creating and Editing a User and (see page 664) Assigning Roles (see page 0) • Authentication/Authorisation and Connection to Active Directory (see page 671) • Roles/Permissions (see page 632) et seqq.
2	Set up additional Email accounts if necessary (Inbox)	Admin module > Communication > Email > Email Accounts	• Inbox (see page 201)
3	Set up Email addresses (outbox)	Admin module > Communication > Email > Email Addresses	• Email Addresses (see page 228)
4	Set up Teams (Queues)	• Admin module > Ticket > Teams	• Teams (see page 617)
5	Set up Default Team (Default Queue)	• Admin module > System > SysConfig > key "PostmasterDefaultQueue"	• Setting Up the Default Team (see page 625) (Default Queue) (see page 0)

Sequen ce	What do I need to do? (mandatory)	Where do I do that?	Which chapter in the manual?
6	Create/import organization data	• "+ New" button > "New Organisation" tab • Customer module > Overview Organisations > button: "New Organisation" / "Import"	In the user manual
7	Create/import contact data	• "+ New" button > "New Contact" tab • Customer module > Overview Contacts > button: "New Contact" / "Import"	In the user manual
8	Set up FAQ categories	Admin module > Knowledge Database > FAQ	• Knowledge Database (see page 267)
9	Update content of FAQ	FAQ module	In the user manual

Optional and in any order:

What do I can I do? (optional)	Where do I do that?	Which chapter in the manual?
Configure asset classes	Module Admin > Assets > Asset Classes	• Asset Classes (see page 120) • Creating or Editing an Asset Class (see page 123) • Class Definition (see page 126) et seqq.

What do I can I do? (optional)	Where do I do that?	Which chapter in the manual?
Edit ticket settings	Module Admin > Ticket > Priority | States | Types	• States (see page 614) et seqq. • Priorit (see page 611) ies (see page 611) et seqq. • Types (see page 629) et seqq.
Create text modules	Module Admin > Ticket > Text Modules	• Text Modules (see page 626)
Set ticket notifications	Module Admin > Automation > Notifications	• Notifications (see page 184) • Creating or Editing a Ticket Notification (see page 186)
Configure email filters	Module Admin > Communication > Email > Email Filter	• Email Filters (see page 231) • Configuring an Email Filter (see page 233)
Maintain multiple languages	Module Admin > Internationalisation > Translations	• Translations (see page 264)
Create automated jobs	Module Admin > Automation > Jobs	• Jobs (see page 170) • Creating or Editing a Job (see page 175)

5.3   Language settings

The initial system language of KIX is English. The German language translation of the system is included. You can change the system language in the "Preferences".

To do so, click on the gear icon in the header area. In the dialog that opens, select the desired language under "*Localization > Language*" and save your selection.

Fig.: The gear icon for language selection
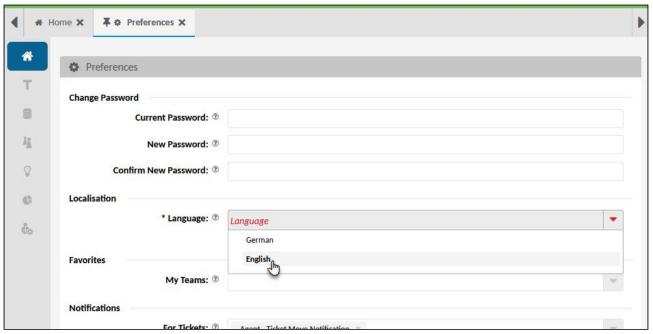
Fig.: Selecting the language in the personal settings

Further information on language settings: Translations

6 Basic Information

The chapter "Basic Information" provides you with an overview of the structure of KIX and is intended to familiarise you with the user interface in the Admin module.

6.1   Overview of the Admin Module

The Admin module is part of the agent portal and is opened via the "Admin" button ![icon] in the module menu on the left. It is only available to users with the roles "Superuser" and "System Admin".

In the Admin module, you make the basic settings for KIX, administer the user rights and configure the views and functions for the agent portal and for the Self Service Portal[1] according to your company-specific requirements. The Admin module supports you in:

- Administration of the company's internal infrastructure (machines, devices, buildings, vehicles, contracts, etc.)
- User and rights management
- Setting up e-mail communication
- Administration of ticket properties
- Control and automation of tickets
- Modification of the user interface (GUI)
- Creation and integration of additional input and selection fields
- Configuration of the additional modules of KIX Pro
- and much more,

---

1) The Agent Portal is the application area that agents work with. The Self Service Portal is the customer portal and can only be used with KIX Pro.

6.1.1 The Admin Dashboard

The Admin Dashboard opens after clicking on the Admin module in the left module bar �-.

The left sidebar (see page 97) contains the module explorer 7 . It contains

- the Setup Assistant (see page 58) , which guides you through the initial setup of KIX. It is displayed after the first login and can be opened again at any time.
- the **KIX** administration menu in an alphabetically ordered menu tree. This is used for the administration and individual configuration of the agent and self-service portal.
  Select a menu item in the tree to load the content into the content area (see page 102) 8 . The table displayed there contains the individual configuration objects. Clicking on a table entry takes you to the zoom view (see page 105) of the selected object or to its editing mode (see page 107) .
- the configuration options for the additional modules and extensions you have purchased.

If you are using a small monitor, the sidebar is displayed collapsed. Click on the double arrow at the top of the sidebar to expand and collapse the sidebar.

Due to authorisation checks, loading the Admin module may be delayed by a few seconds in low-performance environments.

Content on this page:

- Overview (see page 87)
- Determine IDs (see page 92)
- Mobile View of KIX 18 (see page 94)

ⓘ **Note to the admin manual**

Both the structure of the table of contents and the navigation links in the chapters of the admin manual are based on the structure of the KIX menu tree. In the manual we refrain from the explicit reference that a menu item is located below KIX and use, for example, "Navigate to *System > SysConfig*" instead of "Navigate to *KIX > System > SysConfig*".

6.1.1.1 Overview
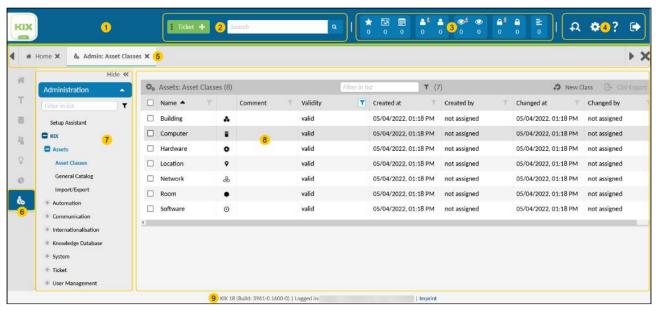
Fig.: Structure of Admin Dashboard

No	Area	Description
1	Header	KIX header with logo and toolbar for frequently used functions
2	green Plus-Button and Fulltextsearch	Create new objects (tickets, assets, organisation, contact, FAQ). The selected object also defines the context for further actions, e.g. the fulltext search.
3	Quick access to important functions and views of the logged-in user	Quick access to frequently used agent functions (see user manual "Home Dashboard"). • Personal views (favourites, kanban, calendar) • Tickets • Watched tickets • Locked tickets • News and Error notes

No	Area	Description
4	Tools	• Complex search for selected object properties • Personal settings of the user • Welcome page with help, tips & tricks and link to manuals. • Logout **Tip:** The links to the manuals support voice control. This means that the URL to the manuals can vary according to the language used. This may be necessary if, for example, you have created additional languages in the SysConfig key "DefaultUsedLanguages". In the following SysConfig keys, you can specify one URL per language: • KIX::SelfServiceManual • KIX::UserManual • KIX::AdminManual

No	Area	Description
5	Tab bar	Contains the open dialogues in one tab each.
		You can move the tabs in their order by dragging and dropping.
		To close the tab, click on the cross in the tab or click on the tab with the middle mouse button.
		📌A pin marks the tab as pinned. Pinned tabs are retained after logout. Double-click on a tab to pin or unpin it.
		Note: In order to ensure that configurations are completed, subsequent dialogues cannot be pinned:
		• Create/Edit Job • Create/edit actions (KIX Pro) • Create/edit templates (KIX Pro) • Create/edit report definition • Import organisations • Import of contacts • Collective actions
		🔄 The reload icon in the tab indicates a change in the tab content. Clicking the tab again reloads the tab content and removes the icon.
		❎ Click on the X on the right side of the tab bar to close all tabs.

No	Area	Description
6	Module menu	Switching between the KIX modules • **Home Dashboard:** Dashboard of the logged in user • **Tickets:** Creation and editing of tickets • **Assets** (CMDB): Management of the company's own infrastructure and resources (assets) • **Organisations:** Create and edit customers and contacts • **Service contracts:** Create service contracts (KIX Pro only) • **FAQ:** Administration of FAQ entries (knowledge base) • **Reports:** Creation of reports • **News:** Create and manage messages to agents and customers (KIX Pro only) • **Admin:** Administration of the agent and self-service portal as well as possible KIX Pro add-on modules The available modules depend on the rights of the user who is logged in.
7	Left Sidebar (see page 97)	Explorer with administration menu in alphabetical order. Clicking on the double arrow opens or closes the explorer.
8	Content Area (see page 102) (Dashboard)	Display of the page content, according to the selected menu: • Dashboards • Overview tables with search and filter functions • Detailed views • Creation and editing dialogues The display of the page content can be configured for the frontend (see configuration of the user interface (see page 327) (GUI))

No	Area	Description
(without)	Right Sidebar (see page 97)	Contains widgets to provide various information. A click on the double arrow shows or hides the sidebar. In the admin module, the sidebar is only available in selected dialogues (e.g. "New e-mail filter"). The widgets to be displayed in the frontend and their contents can be configured. (see Configuration of Sidebar (see page 412))
9	Footer	Program version (see page 116) , user who is currently logged in, and imprint. You can store the link to the imprint in the SysConfig key "*ImprintLink*" (menu *System > SysConfig*)

6.1.1.2 Determine IDs

When configuring KIX, you will need the ID of objects (ticket, asset, organisation, contact, etc.). For example, in placeholders, when configuring roles or to reference a specific object directly. You can determine the IDs in different ways:

Determine ID via the URL of the address bar

Every open tab, every view, every search etc. has a unique URL. Each object can be called up directly using the URL. For example, calling up the organisation with the ID "1": `http://`
`your.agentenportal.kix.cloud/organisations/1.`

The URL is displayed in the address bar of the browser. It contains both the object type of the opened object (e.g. organisations) and its ID (e.g. 1).

You can also use the URL to call up the current view in a different context or via other services (see also Using the browser URL).
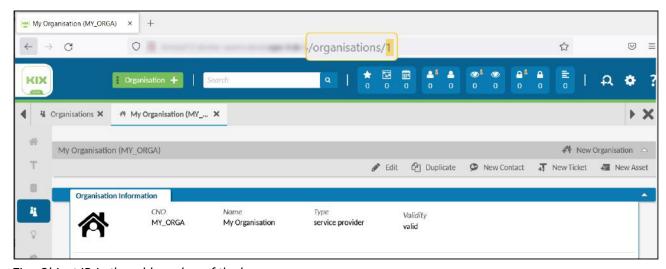
Fig.: Object ID in the address bar of the browser

Reading the ID in the browser footer

You can also read the ID of an object in the footer of the browser, e.g. to determine the ID of the priority.

- Navigate to the overview of the desired object (e.g. *Ticket > Priorities* or *System > Dynamic fields* or *Ticket > Status* etc.).
- Move the mouse pointer over the name of the desired entry.
- Read the ID in the footer of the browser. The ID is the last parameter (last number).
- This works with every object and in all overviews.
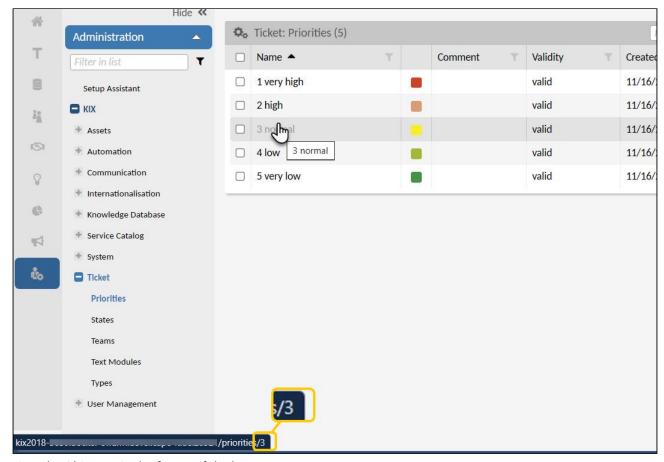
Fig.: The Object-ID in the footer of the browser

6.1.1.3 Mobile View of KIX 18

The appearance of KIX depends on the screen size. On smaller screens, the content area takes up the entire width of the screen and the menu and header lines are minimised:

Button		Description
☰	Hamburger Button	Displays the module menu. Displays the open tabs below the module menu.
»	left double arrow	Opens the left sidebar with the Explorer.
▼	Dropdown-Button	Opens the toolbar for fast access.
+	New-Button	Opens the dialogue for creating new objects.
«	right double arrow	Opens the right sidebar with widgets (frontend only).

6.1.2 Elements of programme interface

The programme interface contains various, partly configurable, elements such as

- Sidebars (see page 97)
- Widgets (see page 101)
- Lanes (see page 97)
- Detailed views (see page 105)
- etc.

The following chapters provide an overview.

Elements of programme interface	Description
	Personal Home Dashboard On the personal home dashboard, agents can find a quick overview of all events.
	Module menu All available modules (e.g. tickets, assets, FAQ) can be easily accessed from any page via the module menu with just one click.
	Explorer On dashboards, the explorer in the left sidebar helps you to keep an overview. This has the familiar tree structure and guarantees easy navigation even in extensive structures.
	Tables Clear tables, which you can sort and filter according to your needs, help you optimally with your daily work processes.

Elements of programme interface	Description
	Lanes On the detail pages of the individual objects (e.g. ticket, asset) you will find a lot of information, which is neatly arranged in several lanes to save space.
	Sidebars Various additional information (e.g. contact details) are discreetly located in the right sidebar, which you can easily show and hide. Various additional information (e.g. contact details) are discreetly located in the right sidebar, which you can easily show and hide.

6.1.2.1   Sidebars

The sidebars are located to the left and right of the content area (see page 102) . They can be shown or hidden by clicking on the double arrow.

You can configure (see page 412) the widgets (see page 101) contained in the sidebars and their contents in the menu *System > SysConfig* according to your own needs.

Left sidebar

The left sidebar contains the module explorer. The module explorer in the admin module includes the administration menu in an alphabetical tree structure. Here, all configuration options of KIX are grouped thematically. Click on a menu item to load the page content into the content area.
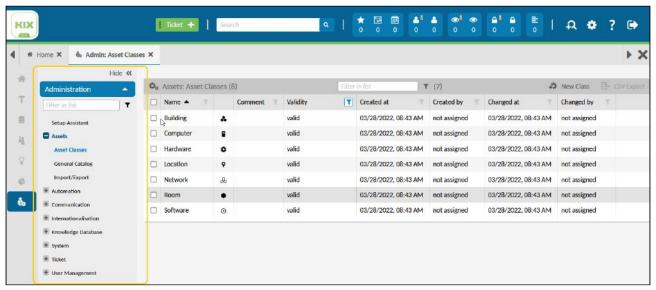
Fig.: The left sidebar with the "Administration" module explorer.

Menü	Aufgabenbereiche
Setup Assistant	Starts the Setup Assistant for the initial setup of KIX
Assets	Device database (formerly: CMDB); 1. Asset classes (create and maintain equipment) 2. General Catalog (create selection values) 3. Import/Export (CSV import and export of assets)

Menü	Aufgabenbereiche
Automation	Automate KIX functions: 1. Set up system notifications 2. Set layout of system notifications 3. Set up automated jobs
Communication	Set up communication channels: 1. E-mail (set up inbox, outbox and distribution, set up team e-mail addresses). 2. Web form (create contact form)
Internationalization	Maintain multilingualism of the system
Knowledge Database	Create and manage FAQ categories
Service Catalog (KIX Pro)	Establishment of Service Level Agreements (SLA)
System	Configure system settings 1. Create dynamic fields 2. Manage icons (KIX Pro) 3. Execute console commands 4. Viewing log files 5. Migration from KIX 17 (KIX Pro) 6. Configure the system and customise the user interface
Ticket	Make basic settings for tickets 1. Make basic ticket settings (priorities, status) 2. Create and manage teams (queues) 3. Manage text modules 4. Create and manage ticket types 5. Set up ticket actions and ticket templates (KIX Pro)
User Management	1. Create and manage users 2. Assign roles and permissions

Right sidebar

The right sidebar contains various widgets (see page 101) with additional information. In the Admin module, the right-hand sidebar is initially only available in selected dialogues (e.g. the "New E-mail filter" dialogue).
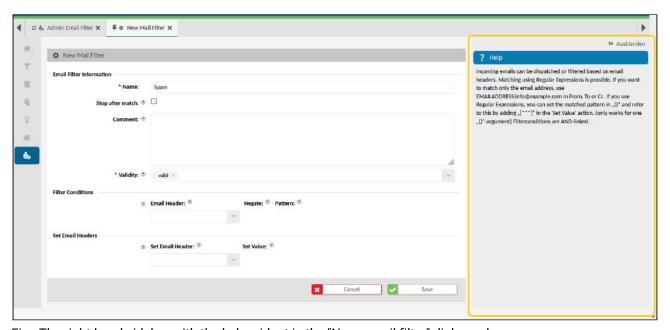
Fig.: The right-hand sidebar with the help widget in the "New e-mail filter" dialogue box

In the modules, the sidebar contains several folding widgets with various information for agents:

- In the home dashboard: the Notes widget, where agents can leave personal notes.
- In the ticket views: initially 4 configurable widgets with information about the Contact, Affected assets, Suggested FAQ entries and the tickets of the contact.
- In form dialogues: texts for help and - if configured - further widgets with information.
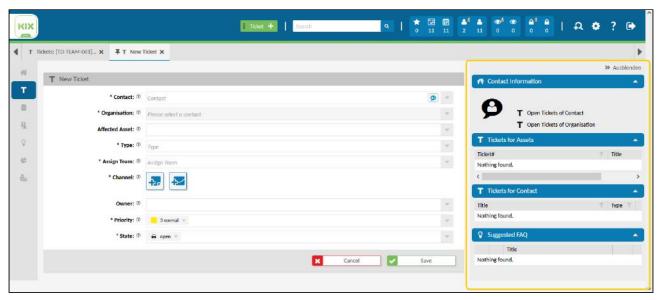
Fig.: The right sidebar in the ticket new dialog with 4 widgets

6.1.2.2   Widgets

Widgets are containers with configurable content. A distinction is made between

- Card widget (object information card widget)
  - Contain text or image elements that are displayed column- and row-wise.
  - e.g. widget "Contact Information" in the sidebar or "Ticket Information" in the upper lane of the detail view.

- Table widget (table widget)
  - Contain object tables
  - e.g. widget "Suggested FAQ" or the tables in the content area

The widgets are configured by adjusting the corresponding SysConfig keys in the *System > SysConfig* menu (see Practical section > The configuration of widgets (see page 336)). (see page 336)

Fig.: The "Contact Information" card widget
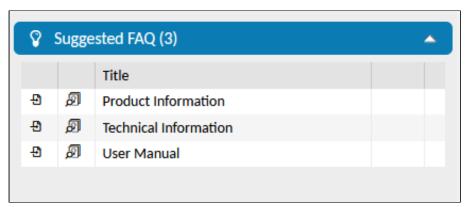
Fig.: The "Suggested FAQ" table widget

6.1.2.3   Content Area

The content area is the content and work area. In the admin module, it opens only after selecting a menu in the module explorer of the left sidebar.

The contents of the selected menu are displayed in tables. The table entries are displayed in alphabetical order by name. You can reverse the sort order by clicking on a column header.

You can limit the table entries displayed by entering filter terms:

- Filter by table content: Enter the filter term in the search field in the table header and click on the filter symbol next to it.
- Filter by column content: Click on the filter symbol in the column header and enter the filter term in the text field that opens.

For performance reasons, large tables are not loaded automatically (e.g. SysConfig). A full text search (see page 103) is available to limit the table to the relevant table entries.

Clicking on a table entry takes you to the zoom view (see page 105) or to the editing mode. (see page 107)
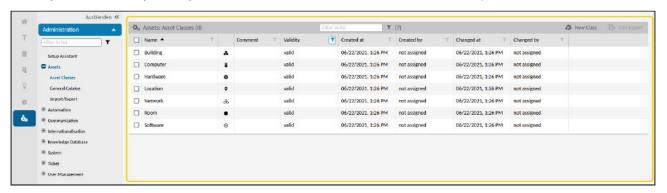
Fig.: Content Area
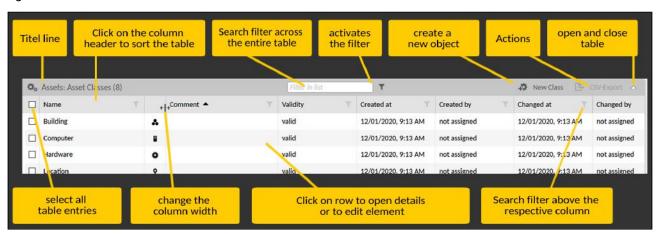
Fig.: Structure and Functions of Content Area

ⓘ **Please note**

The table in the content area lists the name of an object exactly as it was created. However, the zoom view is translated into the language used by the user who is logged in. If German users use English designations, for example, it is therefore possible that the table in the content area will list the English designations, that these designations are translated into German in the zoom view, but that the form dialog displays the name in English again (exactly as it was created). This is not an error in the program, but is due to the way in which the languages are managed.

Full text search

A search function is available in very extensive overviews. The search function allows you to search for a specific term or to limit the table entries to be displayed from the outset. This is the case, for example, in the user or SysConfig tables. Enter the search term in the search field and click on "Search" or press ENTER.

Unknown parts of the search term can be replaced with wildcards (*) anywhere in the search phrase (e.g. "ticket-new*", "mu*man", "*stern"). If you only enter the asterisk in the search field, all table entries will be listed. Loading all table entries may take a moment.

System: Sysconfig (18)						Edit	Reset	Reload Fronten...

ticket-details*

Name ▲	Modified	Context	Metadata	Access Le...	Validity	Value	Changed at	Changed b
ticket-details		kix18-web-frontend	Context	internal	valid	{"id":"ticket-details","name":"Ticket Details","ty...	06/22/2021, 1:33 PM	not assigne
ticket-details-affected-asset-tickets		kix18-web-frontend	Widget	internal	valid	{"id":"ticket-details-affected-asset-tickets","nam...	06/22/2021, 1:33 PM	not assigne
ticket-details-article-attachments-overlay		kix18-web-frontend	Widget	internal	valid	{"id":"ticket-details-article-attachments-overlay"...	06/22/2021, 1:33 PM	not assigne
ticket-details-article-list-table-config		kix18-web-frontend	TableWidget	internal	valid	{"id":"ticket-details-article-list-table-config","na...	06/22/2021, 1:33 PM	not assigne

Fig.: Search function for very extensive tables

> ⓘ **Note on wildcard search in organizations and contacts**
>
> For the search for users as well as for customers and contacts in the agent portal, no asterisks have to be set as wildcards. KIX automatically sets an asterisk internally after the search term (wildcard suffix). The same applies to the autocomplete fields.
> The asterisk before the search term (wildcard prefix) is not initially set automatically. It can be set manually when entering the search term.
> For contacts and organisations, you can set whether the wildcard prefix should be set automatically (see page 588) . This is done in the keys "*ContactSearch::UseWildcardPrefix*" and "*OrganisationSearch::UseWildcardPrefix*" of the system configuration.

CSV Export

You are able to export the table or selected table entries to other programs in the form of a CSV file. To do so, select required table entries and click "CSV-Export" in table header. In browser dialog that opens, select how you would like to use CSV file.

	Name		Comment ▲		Validity		Created at		Created by		Changed at		Changed by
☐	Building	♣			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned
☑	Computer	▮			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned
☑	Hardware	⚙			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned
☐	Location	♀			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned
☐	Network	⅋			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned
☐	Room	●			valid		12/01/2020, 9:13 AM		not assigned		12/01/2020, 9:13 AM		not assigned

Fig.: Exporting Selected Assets to a CSV File

6.1.2.4   Zoom Views

The zoom views display additional information on an object (asset, agent, team, etc.) and enable you to edit the corresponding master data. Object zoom views open when you click the object in the table in the content area.
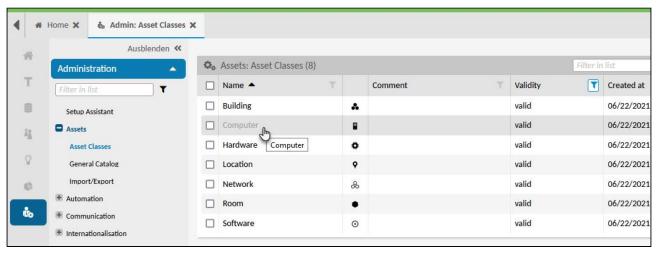
Fig.: Opening Zoom View of "Computer" Asset

Structure of Zoom View

The information listed in the detail view includes reference data, authorisations, assignments, versions, execution plans, etc.. These are displayed grouped in so-called **lanes**.

The first lane always contains the reference data of the selected object. It is always labelled "[object designation] information". It can contain additional tabs, such as the "History" in the detailed view of a job.

All other lanes contain variable contents depending on the selected object. Each lane contains a small white arrow at the top right edge with which you can expand and fold the lane.

Fig.: Structure of Zoom View

No	Area	Description
1	Title bar	Displays the object that has been opened and the buttons • for creating a **new** object • for **editing** the object that has been opened • for **printing** or **duplicate** the current screen view
2	Lane "[Object] Information"	The first lane of the detail page always contains the reference data of the opened object. Further tabs are possible, e.g. "History" in the menu "*Automation > Jobs*".
3	Lane "Version Details"	This lane is shown for objects that have been assigned a version, e.g. asset classes. Expanding a version lets you view the class definition for an asset class, for example. If required, you can copy it to the clipboard for use as a template when creating a new class.
4	Arrow buttons	For expanding and folding the lane.

6.1.2.5   Form Dialogs

Form dialogs always open when you create something new or edit something. They open in what is known as an overlay and contain form fields for entering and selecting the data.
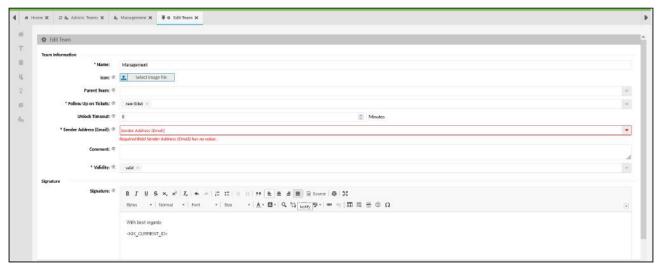
Fig.: Form Dialog for Editing a Team

The fields shown in the form vary depending on the context from which the form was opened. However, some form elements are included in almost every form:

Form element	Description
	Fields with a red border indicate that information is missing here.

Form element	Description
	By selecting the validity, you define whether or not an element can be used by the system.

Data cannot be deleted once it has been saved in the system, as doing so would eliminate the underlying basis for tickets, assets, or teams. Instead, you can set the validity to "invalid" or "invalid-temporarily". If you select either of these options, the element will no longer be used. The two are differentiated in terms of their context only: Whereas selecting "invalid" means the system permanently refrains from using the element, selecting "invalid-temporarily" means the system only
temporarily refrains from using it. This may be the case for a guest user, for example, who is only permitted to work with the system for a limited period of time.

Please note: (Temporarily) invalid elements are not displayed or used in the agent portal or self-service portal. In the Admin module, invalid elements are grayed out but can still be selected. |
| | The comment fields are always optional. They enable you to save explanatory notes for the element that you are currently editing or creating. These comments are only visible in the Admin module. The comments are not shown in the agent portal. You can scale comment fields at the bottom right corner. |
| | Forms often contain blocks of content where the form fields are grouped by topic. There are small white arrows at the edge of these information blocks for expanding and folding the blocks of content as required. |
| | You can enter formatted text in the editor. The use of KIX placeholders is possible.
The editor can be scaled at the bottom right corner (triangle symbol). |
| | Clicking the question mark displays information about a form field. |

Form element	Description
	Selection fields on the ticket (e.g. contact, responsible person, affected assets, related tickets, etc.) expect at least 3 characters to be entered in order to make a preselection for display in the selection field. You can use the asterisk (*) as a wildcard for unknown text parts. Possible combinations for a search for "Anderson" would be, for example: *An, wil*an*, An*. You can enter 3 asterisks (***) if no part of the text is known or if all elements are to be displayed. This may take a moment. You can also find information on General Function Descriptions in the KIX Start Users Manual.

Ability to Select Invalid Elements

In the Admin module and in the Advanced Search, you are also able to select and edit invalid elements. This means you can set up jobs or notifications as part of preparatory work, for example.

Example: You select a (temporarily) invalid guest agent who should receive several ticket notifications. At the relevant point in time, all you need to do is set the guest agent to "valid" for them to receive all ticket notifications – as permitted by their roles.

Invalid objects can also be selected in jobs – for example, a job can be run using an invalid team. Likewise, in the Advanced Search it is possible to find tickets with services that are no longer being offered. Furthermore, you can edit invalid elements such as deactivated SysConfig keys or users and set them to "valid" again. Invalid elements are grayed out in tables and selection fields.
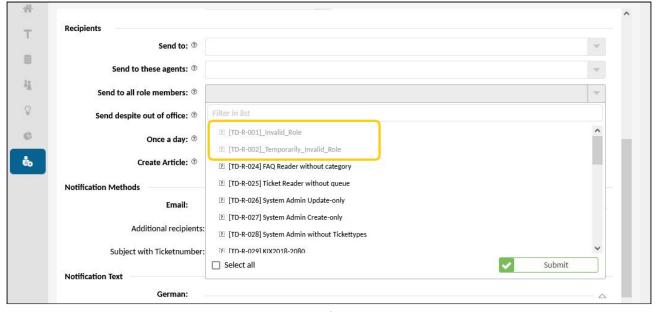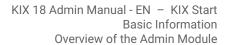
Fig.: Selecting Invalid Agents When Creating a New Notification

ⓘ **Please note**

If, however, invalid dynamic fields are specified in notifications, for the search function, or to filter tickets in jobs, they will <u>not</u> be used when these options are executed.

6.1.3 Keyboard Navigation

Below you will find an overview of the keyboard commands and shortcuts that you can use in KIX:

Key/Shortcut	Function
[Esc]	Closes the dialog or selection area without saving (the same as "Cancel"). The focus moves off the field when doing so.
[Enter]	Selects the value that is currently highlighted; confirms and closes the field. In single selection fields: Switches to the next field.
[Space]	In single selection fields: Selects the entry and closes the field. In multiple selection fields: Selects or deselects the value that is in the focus, leaves the field open.
[Tab]	In single selection fields: Selects the entry, closes the field, and switches to the next field. In multiple selection fields: Switches to the next functional element (field or button); sets the focus in the list filter.
[Shift]+[Tab]	Jumps to the previous field.
[Shift]+[Navigation]	Highlights all values that are in the focus
[Ctrl]	Shifts the focus of the keyboard control to the dialog
[Ctrl]+[Tab]	Switches tabs in the browser (if several tabs have been opened)
[Ctrl]+[Enter]	Save/start search
[Ctrl]+[right arrow key]	Next page
[Ctrl]+[left arrow key]	Previous page
[End]	Jumps to the last value in the selection field.

Key/Shortcut	Function
[Home]	Jumps to the first value in the selection field.
[Page Up] or [PGUP]	Jumps up 10 values in the selection field.
[Page Down] or [PGDN]	Jumps down 10 values in the selection field.

ⓘ **Notes**

- The keyboard control only works if the focus is also in the dialog/form or in the relevant element. If the focus is elsewhere, the control will not respond either.
- To edit a form with multiple pages, the navigation changes as follows:
    - [Ctrl]+[Left arrow] - jumps to the beginning of the word
    - [Ctrl]+[Right arrow] - jumps to end of word
    - With [Ctrl]+[Arrow left] and [Ctrl]+[Arrow right] you can NOT change the page of the form.
- Switching between the individual tabs via keyboard control is not possible.

6.2 KIX Architecture

The development of KIX 18 (hereafter KIX) is based on Docker with a consistent focus on container deployment (analogy: like an .exe file for a Docker Swarm Runtime environment). This allows us to efficiently further develop KIX and thus expand the range of functions with each new release. In addition, container deployment allows us to provide customised KIX for on-premises solutions as well as the simple inclusion of KIX extensions such as KIX Connect.

You can use KIX as a cloud solution or host it yourself (on-premises). KIX is available in different versions: KIX 18 Start, KIX 18 Pro, KIX 18 Cloud (see introduction (see page 13)). KIX Start is also available free of charge for self-operation with a reduced range of functions. Regardless of the operating model and range of functions, all KIX versions are subject to open source software licences, which are based on the same technology and are maintained in the same way.

System requirements and information on the on-premises installation of KIX Start/Pro can be found at: Installation (see page 26) . KIX Pro customers can also request access to release candidates (RC) before the final release versions are published. Please note that the use of release candidates is **not recommended** for productive environments.

Container images are provided for use in in-house operations. The required Docker Compose Environment can be obtained from https://github.com/kix-service-software/kix-on-premise (see also Installation (see page 26)). Further instructions for use in Kubernetes or OpenShift environments can be found at:

- https://kubernetes.io/docs/tasks/configure-pod-container/translate-compose-kubernetes/
- https://kompose.io/getting-started/

KIX includes various services as delivered:

- Frontend: Graphical user interface (GUI) for the agent portal [1]
- Backend: The heart of KIX, accessible via the REST API interface [2]
- Database: KIX's own PostgreSQL database
  Alternative database: You can specify a different DBMS in the environment file during the 1st start-up.
  - only for on-premises installations
  - only PostgreSQL version 12 or higher recommended
  - Long-term support for other DBMSs is not guaranteed.
- Cache
- Self Service Portal (customer portal, KIX Pro required)

KIX can access various external services:

- LDAP-based directory services such as Active Directory (several possible)
- external web services (KIX Pro extension)
- various external database management systems (KIX Pro extension KIX Connect Database)
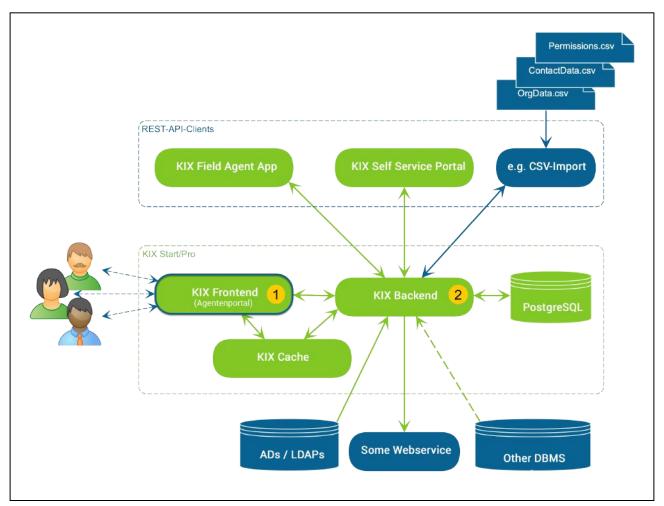
Fig.: KIX Architecture

6.2.1 Scheme KIX Database

For the SQL statement in report definitions or for the configuration of individual components, it is helpful to know the structure and naming within the KIX database. In the following, we provide you with information on this.

Information on the structure of the KIX database:

- as SVG graphic: https://github.com/kix-service-software/kix-backend/blob/master/doc/database/KIX.svg
  (Full screen: click on "Raw"; then use zoom and scroll functions of the browser)
- as XML schema: https://github.com/kix-service-software/kix-backend/blob/master/scripts/database/kix-schema.xml

Users of on-premises installations can also create the SQL database schema using various tools such as

- DBSchema: https://dbschema.com/?AFFILIATE=144826&__c=1
- Documentation DBSchema: https://dbschema.com/documentation/
- SchemaSpy: https://dev.to/mostalive/how-to-visualize-a-postgresql-schema-as-svg-with-schemaspy-516g

In Ubuntu Linux environments, the database schema can be generated with the following script:

```
# extract SQL schema from backend container...
dockerhost$ docker exec -it kixpro-backend-1 /opt/kix/bin/kix.Console.pl
Dev::Tools::Database::XML2SQL --database-type postgresql --source-path /opt/kix/
scripts/database/kix-schema.xml --target-filename /tmp/kixdb.sql.postgresql.sql
dockerhost$ docker cp kixpro-backend-1:/tmp/kixdb.sql.postgresql.sql.postgresql.sql /
tmp/

# install SQL translator on docker host...
dockerhost$ apt install libsql-translator-perl

# prepare SQL schema...
dockerhost$ sed -i.orig '/--\|SET/d' /tmp/kixdb.sql.postgresql.sql

# apply SQL translator on docker host to generate SVG (filter non-relevant tables)...
dockerhost$ sqlt-graph -o /workspace/KIX.svg -t svg -f PostgreSQL --show-datatypes --
skip-tables-like "(auto_response.*|pm_.+|process_id|package_repository|
system_maintenance|kix_ticket.*|kix_dep_dynamic_field_prefs|kix_dep_dynamic_field|
addressbook|customer_portal_group|acl.*|attachment_storage|virtual_fs_db|
smime_signer_cert_relations)" /tmp/kixdb.sql --layout sfdp --fontname courier --show-
sizes --show-datatypes --show-constraints --node-shape "record" --edgeattr
"color=#a9a9a9" --edgeattr "arrowhead=crow" --edgeattr "style=dashed" --graphattr
"overlap=false" --graphattr "rankdir=LR" --graphattr "ratio=0.5" --graphattr
"splines=true"
```

6.3   Version numbers

KIX pursues a rolling release strategy and consists of several components such as the

- Backend / core
- Agent portal
- Self-Service Portal (KIX Pro).

The versions (Build) of the agent portal and the kernel (Backend) are displayed in each installation after logging into the agent portal in the lower screen area. Both together clearly identify the software version. We dispense with additional coding of this information in the form of a release number.

The build numbers are supplemented by the patch release numbers. The patch release numbers are separated from the build numbers by a hyphen. They are incremented by 1 with each published patch release and reset to 0 with each new build.

Example: KIX 18 (Build: 3888-0.1565-0)

- 3888-0: The first number is the build number of the agent portal (3888), followed by the patch number of the agent portal (-0).
- 1565-0: The second number is the build number of the backend (1565), followed by the patch number of the backend (-0).
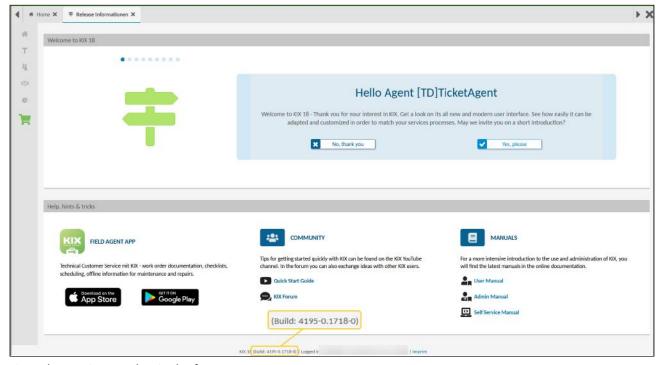
Fig .: The version number in the footer

The description of the content or the changes contained in the respective build can be found in our forum, eg: https://forum.kixdesk.com/index.php?board=22.0 A Google search for, for example, "3663.1415" usually

also leads for the information you require. The build numbers correspond to the announced releases and allow a very precise identification of the installed software version.

With this approach it is also possible to provide smaller patch or interim releases.

The build and patch numbers can also be output via console command. Navigate to *System > Console* and start the command `Console::Command::Admin::Installation::ListPlugins` .

Administrators of an on-premises installation can also determine the software version directly from the front-end or back-end containers.

Determine the version number of the front end from the Docker host:

```
root@ubuntu20:/opt/kix-on-premise/deploy/linux# docker exec -it kix_frontend_1 cat /
opt/kix/RELEASE
PRODUCT = KIX
VERSION = 18
BUILDDATE = Mon, 21 Feb 2022 09:48:51 +0200
BUILDHOST = git.intra.cape-it.de
BUILDNUMBER = 3888
PATCHNUMBER = 0
```

Determine the version number of the backend from the Docker host:

```
root@ubuntu20:/opt/kix-on-premise/deploy/linux# docker exec -it kix_backend_1 cat /
opt/kix/RELEASE
PRODUCT = KIX
VERSION = 18
BUILDDATE = Mon, 21 Feb 2022 09:50:08 +0200
BUILDHOST = git.intra.cape-it.de
BUILDNUMBER = 1565
PATCHNUMBER = 0
```

7 Assets - The Asset Device Database

Under the menu item *Assets* you will find the Configuration Management Database (CMDB), i.e. the asset database of KIX. Here, all resources and resource information are managed that are required for service provision. These can be, for example, locations, buildings, rooms, machines, hardware and software, but also telephones, small devices or contracts.

All these resources are created as so-called **assets** in the system. In order to map these assets in a structured way, they are assigned to different asset classes. The characteristics of the asset classes are flexible and depend on the respective application. They are distinguished by their properties (attributes). KIX offers a number of existing asset classes as delivered. Asset classes are also referred to as Config Item classes, CI classes, in ITIL also as "CI types".

Changes to asset classes and their attributes are recorded in class definitions. These definitions are subject to versioning. Changing the attributes of an asset class does not have a destructive effect on inventory data. These remain in the previous form and content.

Changes to asset information are historicised and also versioned. It is possible to look back to a previous version of the asset information at any time, regardless of whether the referenced asset class definition is out of date or not.

The authorisation check when accessing assets is carried out according to the authorisation concept. Access to assets requires access rights to the resource "configitems" and to the asset itself.

Asset Classes

> The sub-menu *Asset Classes* is where all asset classes are created and configured. The individual assets (operating resources) are assigned to these classes. Agents see the asset classes and corresponding assets in the "Assets" module menu in the agent portal.

General Catalog

> The *General Catalog* is a directory that can hold all kinds of values. For instance, it is where you can save individual selection values for asset classes. For example, you can save the vehicle types "Truck", "Car", and "Van" for the asset class "Vehicles". When creating or editing asset classes, the values in the General Catalog can be referenced in the class definition. This means that when agents create an asset of the "Vehicles" class, they can select from these values and thus narrow down the type of vehicle.

Import/Export

> The sub-menu *Import/Export* enables you to import multiple assets in one go and create them under the relevant asset classes. You are also able to export all assets of a class, in order to edit them in a table and subsequently import them back in as a mass update, or in order to import them into other systems.

> ✅ **Tip**
>
> - If you prefer to use the term "Config Items" or "CI" instead of "assets", you can save a pattern for doing this under *Internationalisation > Translations* and thus amend the term in the system.
> - KIX Pro users can visualise the relationships between assets in a graph and start a failure simulation if required (see user manual).

7.1 Asset Classes

All operating resources used by the company and the corresponding information, such as buildings, machines, devices, or even contracts, are referred to as assets and created as such in the system. To provide structure when they are mapped in KIX, they are assigned to asset classes. For example, all buildings in the company are combined under the asset class "Building", all computers in the company are combined under the asset class "Computer", and so on. Synonyms for assets are also: config item, (technical) resource, device, configuration object.
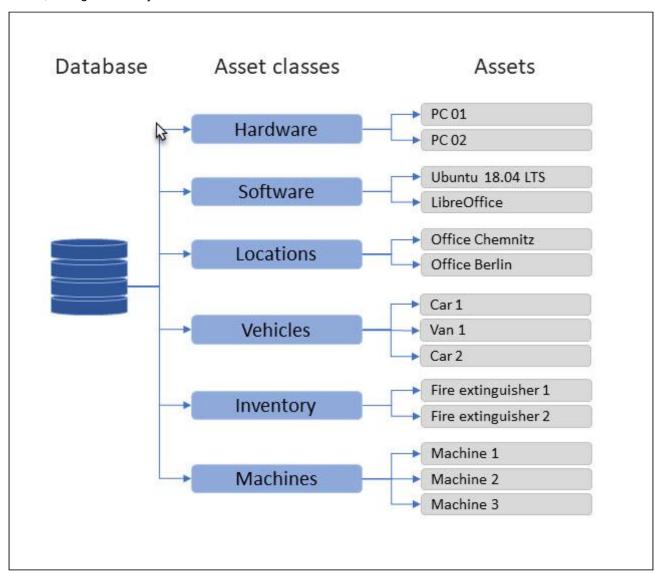
Fig.: Scheme Asset-Database > Asset classes > Assets

To manage these asset classes, go to menu *Assets > Asset Classes*. This is where you can create new asset classes, edit them, and see the corresponding details.

KIX comes supplied with a small selection of asset classes, which enables you to create your rooms, buildings, and some of your devices as operating resources in the system. You can create additional asset classes to also record all other operating resources that your company uses. The table in the Dashboard lists the asset classes created in the system.
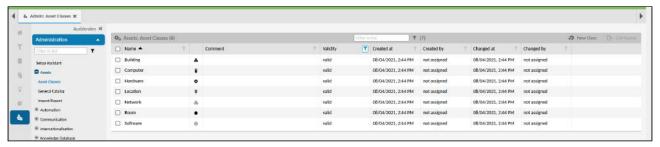
Fig.: Overview of Assets (Example)

Agents can find the asset classes in the Asset Explorer. The names of the asset classes are translated into the language of the user, provided that a pattern for this is stored in the system (menu *Internationalisation > Translations*).
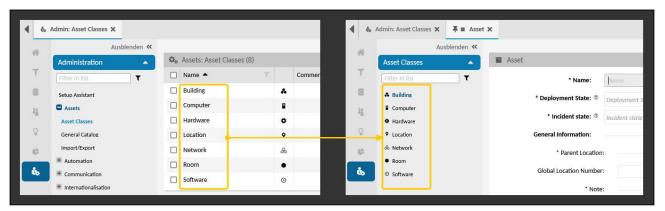
Fig.: The asset classes maintained in the Admin module are available in the asset explorer.

> ⓘ **Notes**
>
> When creating a new asset class, KIX automatically creates the corresponding new CSV mapping. This mapping enables you to automatically import the assets belonging to the class into KIX (also see the chapter Import/Export (see page 152))

7.1.1 Zoom View of Asset Classes

You are able to view the details of an asset class (master data, class definitions, versions). Navigate to *Assets > Asset Classes*. In the content area, a table listing all asset classes saved in the system opens. In table, click an asset class to open associated zoom view.

Alongside the lane containing the general asset class information, the zoom view also contains the "Versionsdetails" (Version Details) lane. It contains a list of all versions of the asset class, as well as the respective class definition (see page 126) for each version. If required, you can copy the class definition of a

version and use it as a reference class definition when creating new asset classes. You can use the small arrow buttons on the right-hand side of a version to expand and collapse the information displayed for the class definition.

The buttons in the title bar of the detail view allow you to edit and create new asset classes.
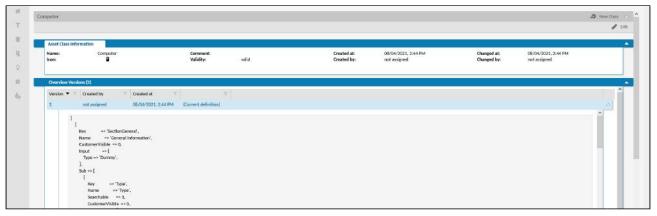
Fig.: Details of "Computer" Asset Class

7.1.2 Creating or Editing an Asset Class

KIX comes supplied with a small selection of asset classes that you can use to create some of your buildings, rooms, and devices as operating resources in the system. You can create other asset classes to record all other operating resources that your company uses.

An asset class is defined by the source code in your class definition. The properties of a class are specified there using attributes. These attributes generate the form fields for creating/editing an asset. When you create a new asset class, you need to save a class definition. To do so, you can either write the source code of the class definition yourself, or use the class definition of an existing asset class as a reference. Additional information on the class definition and source code is provided in chapter "Class Definition ".
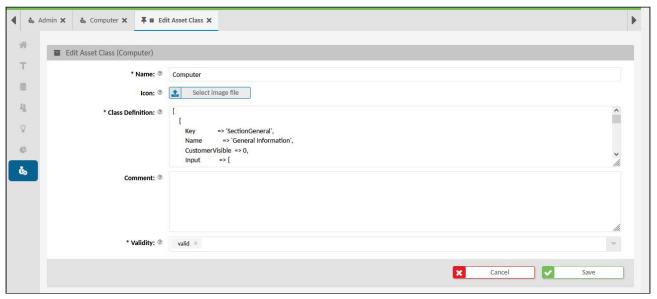
Fig.: Editing the "Computer" asset class.

> ⓘ **Notes**
>
> Each change made to a class definition generates a new version. Old versions cannot be reactivated. If you need to access an old version, you need to copy the old class definition into the current class definition, thus generating a new version.
>
> It is not possible to delete an asset class, because doing so would eliminate the underlying basis for the assigned assets. However, you can set the asset class concerned to "invalid". It will then not be possible to create any more assets for this class.
>
> When creating a new asset class, a new mapping with the same name will be generated for automated asset imports and exports (menu Import/Export).

To create a new asset class based on a reference class, proceed as follows:

1. Navigate to *Assets > Asset Classes*. In the content area, a table listing all asset classes saved in the system opens.
2. In table, click asset class to be used as reference for new asset class. The zoom view opens.
3. The lane "Overview Versions" in the zoom view contains one or more versions of the class definition.
4. Open version to be used as reference. To do so, click small arrow button on its edge.
5. Copy source code of class definition to clipboard.
6. Switch back to the zoom view and click on "New class" there. A form dialog for creating the new asset class opens. Under "Name", enter meaningful name for new asset class and optionally select icon for identifying asset class in asset explorer.
7. Paste source code from clipboard into "Class Definition" field.
8. Optionally edit source code of class definition (also see "Class Definition (see page 126) " et seqq.).
9. Optionally add comment, and then set new class to "valid".
10. Click "Save" to save new class.

The new asset class has now been created, meaning assets can be created for this class or imported via the menu *Assets > Import/Export*.

To create a new asset class with its own source code, proceed as follows:

1. Navigate to A*ssets > Asset Classes.* In the content area, a table listing all asset classes saved in the system opens.
2. In table, click "New Class". A form dialog for creating the new asset class opens.
3. Under "Name", enter meaningful name for new asset class and optionally select icon for identifying class in asset explorer.
4. Write source code in "Class Definition" field (also see "Class Definition (see page 126) " et seqq. and "Source Code of Class Definition (see page 128) ").
5. Optionally add comment, and then set new class to "valid".
6. Click "Save" to save new class.

The new asset class has now been created, meaning assets can be created for this class or imported via the menu *Assets > Import/Export.*

To edit an existing asset class, proceed as follows:

1. Navigate to *Assets > Asset Classes.* In the content area, a table listing all asset classes saved in the system opens.
2. In table, click asset class to be edited to open associated zoom view.
3. In opened zoom view, click "Edit". A form dialog opens displaying the data that has already been saved as well as the current class definition.
4. Amend the data and/or class definition as required (also see "Class Definition (see page 126) " et seqq. and "Source Code of Class Definition (see page 128) ").
5. Click "Save" to apply changes.

Your changes now take effect for this asset class and, thus, in all new and existing assets of this class.

> ⓘ **Notes**
>
> Making changes to the class definition automatically updates the corresponding mapping. For this reason, you may need to export the CSV file again if you are maintaining the assets manually (menu Import/Export (see page 152)).
>
> After changing a class definition, the corresponding CSV mapping is only updated automatically if the SysConfig key "ImportExport::CSVMappingAutoCreate###ForceCSVMappingRecreation" has been set accordingly.

The form dialog contains the following input fields, among others:

Field (selection)	Description	
Name	Name of class	
Icon	The selected icon is used to graphically identify the class in the asset explorer. You can choose an icon from one of the provided libraries or upload a graphic file.	
Class Definition	Enter source code for class definition or change it if required (also see "Source Code of Class Definition (see page 128) "). Each class is defined by its source code. It sets out the structure and make-up of the form elements when creating an asset.	
Validity	valid:	The class is permanently available. Other assets can be created for this class. Tickets can be linked to assets of this class.
	invalid/ invalid-temporarily:	The class is (temporarily) unavailable. No more assets of this class can be created. No tickets can be linked to assets of this class. (Temporary) Invalid asset classes are not available for the search for "class" in the complex search.

7.1.3 Class Definition

The class definition is the basic structure of an asset class. It contains attributes which map the properties of an asset class. Each asset class has its own class-specific attributes. For example, the "Computer" class contains attributes for aspects such as the manufacturer, model, and serial number, whereas the "Room" class contains attributes for aspects such as the building, story, and room type.

You can change these attributes (properties) by adding or removing attributes in the source code of the class definition or by changing their properties. In the agent portal, these attributes then make up the form fields available when creating or editing an asset.

All changes made to the class definitions are logged in the history and assigned a version. The zoom view of an asset class enables you to view a previous version at any time.
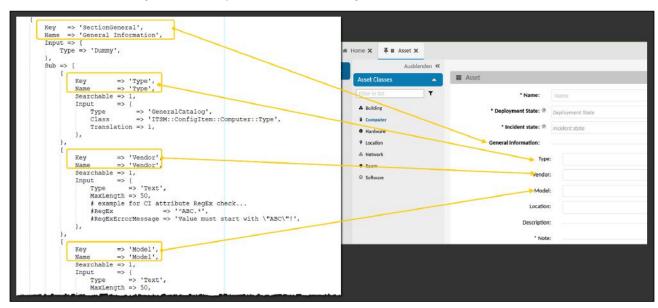
Fig.: Class Definition Generates Form Fields

ⓘ **Migration notes KIX 17 > KIX 18**

- The CI attributes of type "Encrypted Text" existing in KIX 17 are available as asset attributes of type "Text" after migration in KIX 18. This makes passwords visible to users with permissions on assets.
- Perl comments (#) in the class definition code are removed during migration.
- After migration, all CI class definitions including all versions of the source environment are present in the target environment as asset class definitions. If there is already a class of the same name, the migration class gets the prefix "Migration-".
- CI attribute types that are no longer supported are represented as "Text" attributes.
- Each asset class additionally contains an attribute "CIAttachments" (last attribute) of the type "Attachment" to hold the unversioned CI attachments from KIX 17.
- Eeach asset class additionally contains an attribute "CIAttachments" (last attribute) of the type "Attachment" to include the non-versioned CI attachments from KIX17.
- On fields whose type attribute is not identical to the one in KIX17, there is an additional attribute "MigratedType" in KIX18 with the respective value from the type attribute from KIX17.

7.1.3.1 Source Code of Class Definition

The class definition of an asset class consists of a Perl source code. It specifies which form fields are available when creating or editing an asset. If you need other form fields or if the form fields need to be rearranged, for example, you can change the source code accordingly. However, you cannot change the basic elements of an asset, such as the name, deployment state, incident state, or asset link.

> 🛑 **Attention!**
>
> Changing the class definition may cause system errors. You should be familiar with how to handle source code and take great care when making your changes. Please do not hesitate to contact us if you require any assistance.

The following example of a class definition comes from the "Computer" asset class.

No.	Description	
1	Start of a form group. In this example, it concerns the group "General Information". A group contains additional form elements (attributes) such as text fields, drop-down lists, buttons, etc.	
2	This block of code generates a drop-down element with the corresponding text label. In this example, the drop-down contains all the computer types created in the General Catalog. You can copy such a block of code, paste it at the relevant point, and amend it as required.	
3	This block of code generates a text field with the corresponding text label. In this example, it concerns the "Vendor" field with a width of 50 px. You can copy such a block of code, paste it at the relevant point, and amend it as required.	
4	Lines of code preceded by a hash (#) indicate comments. They are not taken into account by the system. Lines initially commented out are notes and examples or attribute properties already prepared for you. If necessary, remove a hash mark to activate the corresponding line of code. You can write your own comments or notes in the code, e.g. to mark individual code blocks or to note notes on the code block. Place a hash mark in front of each comment line. The example opposite shows prepared regular expressions (RegEx).	

7.1.3.2 Editing a Class Definition

You can change the class definition of an asset class by changing the corresponding source code. In doing so, you specify which form fields (attributes) are available when creating/editing an asset, what they look like, and how they are grouped. However, you cannot change the basic elements of an asset, such as the name, deployment state, incident state, or asset link.
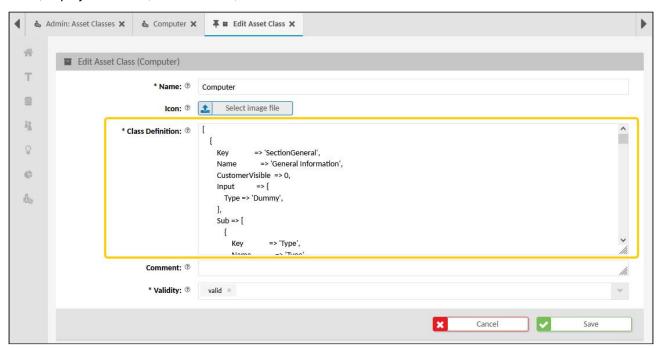
Fig.: Class Definition of "Computer" Asset Class

> ⬥ **Attention!**
>
> Changes made to the class definition constitute a direct intervention in the KIX system! Therefore, you should preferably be familiar with how to handle source code, and you should take great care when making your changes

To change the source code of a class definition you can, for example:

- change the existing attributes and values directly in the source code
- copy parts of the source code, paste them at a different point in the source code, and then amend them if required
- write your own lines in the source code
- delete or replace parts of the source code
- delete or replace the entire source code.

To change the class definition, proceed as follows:

1. Navigate to *Assets > Asset Classes*. In the content area, a table containing all asset classes created in the system opens.

2. In table, select asset class with class definition to be changed. The zoom view of the selected asset class opens.

3. 3. In title bar of zoom view, click "Edit" to open form dialog for editing asset class. In the form dialog, you will find the "Class Definition" field containing the source code of the class definition.

4. To make it easier to edit the class definition, click and hold bottom right-hand corner of text field and drag it to desired size.
   Alternatively, copy source code to external text editor (e.g. Notepad++ or https://jasonformatter.io) , edit code there, and copy code back into form.

5. Change source code by adding other lines of code or changing existing data. To do so, also read chapters "Source Code of Class Definition (see page 128) ", "Attributes of a Class Definition (see page 132) ", "Attribute Types in Class Definition (see page 136) ".

6. Then click "Save" to save your changes.

Your changes take effect immediately in all new and existing assets of this class.

> ⚠ **Important**
>
> When making changes, ensure that the syntax in the source code is correct! For example, place character strings in inverted commas, end each line with a comma, and close all opened brackets. If you copy parts of the source code, ensure that you paste them back in at the right place in the syntax. Observe the hierarchy of any opened brackets when doing so. We would be glad to assist you should you require any support.

> ⓘ **Notes**
>
> - If you use your own attribute designations, you can maintain the translations for them yourself. To do so, navigate to the menu *Internationalisation > Translations* and enter the corresponding patterns there.
> - To display form fields in the Self Service Portal (KIX Pro), set the attribute `'CustomerVisible' = 1` (see also Controlling visibility in the SSP)

7.1.4 Attributes of a Class Definition

The attributes of a class definition generate the form fields for creating assets. The properties and values of the attributes and how they are arranged and grouped define how the form fields are arranged and what they look like – which, in turn, defines the structure of the form. Below you will find a selection of attributes that you can use in the source code of a class definition:

Fig.: Attributes in Class Definition

Attribute	Description	Values	Examples
CountDefault	Default number of entries (if array attribute)	0... *n*	`CountDefault => 0,`
CountMax	Maximum of entries (if array attribute)	0... *n*	`CountMax => 1,`
CountMin	Minimum of entries (if array attribute)	0... *n*	`CountMin => 0,`
CustomerVisible	Controls the visibility in the Self Service Portal 0 = invisible (false) | 1 = visible (true)	0 | 1	`CustomerVisible => 1,`

Attribute	Description	Values	Examples
Input	Describes the data structure of the object. Can be included other attributes such as Type, Size, MaxLength, RegEx, etc.	Hash	```Input => { Type => 'Text', Size => 50, MaxLength => 55, Translation => 1, Required => 0 }```
Key	Internal identifier of the attribute. The form field is saved under this designation in the database. For this reason, the keyword must be unique and may only contain letters and numbers. If it is changed retrospectively, it will no longer be possible to read the data from old definitions.	String alphanumeric	```Key => 'Model',```
MaxLength	Maximum length of the entry in the field	1... *n*	```MaxLength => 55,```

Attribute	Description	Values	Examples
Name	Labeling of the field in the form. It generates the text label shown in front of a form field. The name may comprise letters, numbers, blanks, hyphens, and underscores. Please do not use any control characters such as single or double inverted commas or brackets as this would cause discrepancies with the source code. For German users, for example, we recommend that you use English words so that you can manage the translations using patterns (*Internationalisation* menu).	String	`Name => 'Model',`
Required	Defines required fields 0 = no required (false) | 1 = required (true)	0|1	`Required => 1,`
Searchable	Defines whether the field is searchable. 0 = not searchable; 1 = searchable 🛈 In jobs or actions, only attributes that are searchable are available for selection (`'Searchable' => 1`).	0|1	`Searchable => 1,`
Translation	Value translation yes/no? 0 = The text in the field is not translated automatically. 1 = The text in the field is translated. Only possible for fields with predefined values.	0|1	`Translation => 1,`
Size	Size of the input field	1... *n*	`Size => 50,`

Attribute	Description	Values	Examples
Sub => [...]	Creates a subordinate structure. As a result, the form elements are nested like in a tree structure. The sub-element can, in turn, contain further sub-elements.		```Sub => [{ Key => 'Type', Name => 'Type', Searchable => 1, Input => {...} }]```
Type	Defines the attribute type (also read "Attribut Types in the Class Definition (see page 136) "). For example: • Text = single-line text field • General Catalog = selection field • TextArea = multi-line text field • CustomerCompany = selection field/selection of customer user • Date = date field • DateTime = selection of date and time	String without space/ special character	```Type => 'Text',```
UniqueForCI	The value must be unique within several selection values (CountMax).		
UniqueForCIClass	The value must be unique within the asset class.		

7.1.5 Attribut Types in the Class Definition

Attribute types and their values specify the attributes in the class definition. For example, they define the type of a text field (input), how many characters can be entered, or where the data in a selection field is referenced from. Below you will find a selection of attribute types for the "Type" attribute.

Attribute types:

- Attachment (see page 136)
- CIClassReference (see page 137)
- Contact (see page 139)
- Date (see page 139)
- DateTime (see page 140)
- Dummy (see page 140)
- General Catalog (see page 142)
- (Integer) (see page 143)
- Organisation (see page 143)
- SLAReference (see page 143)
- Text (see page 144)
- TextArea (see page 145)

```
Name        => Model ,
Searchable => 1,
Input       => {
    Type      => 'Text',
    MaxLength => 50,
```

Fig.: The "Text" attribute type generates a text field, in this case with a maximum length of 50 px.

7.1.5.1 Attachment

Enables file attachments to be saved for asset versions.

Example configuration	Notes
```{ 'Key' => 'Attachment', 'Name' => 'Attachment', 'Input' => { 'Type' => 'Attachment', }, 'CountMin' => 0, 'CountMax' => 5, },```	• A maximum of 1 file per attribute can be attached. If several files are to be attached, this must be mapped via the array structure of CI attributes. • If no specific attribute type is used for attachments, they are not versioned. They cannot be restored in earlier versions.

7.1.5.2 CIClassReference

Reference to other asset classes. Enables the selection of valid asset classes.

Example configuration	Notes

``
{
  'CountDefault' => 0,
  'CountMax' => 1,
  'CountMin' => 0,
  'CustomerVisible' => 0,
  'Input' => {
    'ReferencedCIClassLinkDirection'
=> 'Reverse',
    'ReferencedCIClassLinkType' =>
'Includes',
    'ReferencedCIClassName' => [
      'Location',
      'Building',
      'Room'
    ],
 	
'ReferencedCIClassReferenceAttributeKey
' => 'Name',
    'Required' => 1,
    'Type' => 'CIClassReference'
  },
  'Key' => 'ParentLocation',
  'Name' => 'Parent Location',
  'Searchable' => 1
},
``
	• **ReferencedCIClassName:** *n* asset classes can be specified as a range of values in the form of an array.
	If only one asset class is to be used, it can be specified as a single string.
	• **ReferencedCIClassReferenceAttributeKey**: Optional specification.
	Controls the resolution of values for the import/export of asset classes.
	    • If specified:
	        • Export: Names/values of the referenced objects are output instead of their IDs.
	        • Import: The imported values are resolved based on the specified asset attribute ("Name" for asset name). Class-specific attribute keys can also be specified, e.g. "SerialNumber" for referenced class "Computer".
	    • If not specified: The number of the referenced object is output.
	    • Parameter:
	        • Name: Outputs the name of the referenced object instead of its ID.
	            • In the example opposite, the name of the asset referenced in the "Parent location" field is exported and not its asset number (ID).
	        • Key of a class-specific attribute: The value of the object referenced in the asset is exported.

Example configuration	Notes
	• Example: 'ReferencedCIClassReferenceAttributeKey' => 'GLN', The value specified in the "Global location number" field is output (for reference to class "Building"). • If several entries exist, the first entry of the attribute is used. • If the attribute is not set on the asset, the asset number is used.

7.1.5.3   Contact

Allows you to store a valid contact stored in the system (filterable selection list) and is the basis for assigning personal devices to contacts. Selection via filterable selection list.

Example configuration	Notes
```{ 'Key' => 'OwnerUser', 'Name' => 'Assigned User', 'Input' => { 'Type' => 'Contact', }, },```	Assigned user

7.1.5.4   Date

Enables a date value to be saved (calendar selection).

In the backend, the date is saved in the format YYYY-MM-DD hh:mm:ss, whereby the time value is fixed at `00:00:00` .

Example configuration	Notes

``
{
  'Key' => 'FirstUsageDate',
  'Name' => 'First Usage Date',
  'Searchable' => 1,
  'Input' => {
    'Type' => 'Date',
    'Required' => 1,
    'YearPeriodPast' => 20,
    'YearPeriodFuture' => 10,
  },
},
``
	• **YearPeriodPast**: selectable values in past
	• **YearPeriodFuture**: selectable values in future

7.1.5.5 DateTime

Enables a time stamp to be saved (calendar/time selection). The entry is made via a calendar time selection. In the backend, the date is saved in the format `YYYY-MM-DD hh:mm:ss` .

Example configuration	Notes

``
{
  'Key' => 'SeomDateTime',
  'Name' => 'Some Date with Time',
  'Searchable' => 1,
  'Input' => {
    'Type' => 'DateTime',
    'Required' => 1,
    'YearPeriodPast' => 20,
    'YearPeriodFuture' => 10,
  },
},
``
	• **YearPeriodPast**: selectable values in past
	• **YearPeriodFuture**: selectable values in future

7.1.5.6 Dummy

Does not contain a value; is used to structure subordinate attributes.

Example configuration
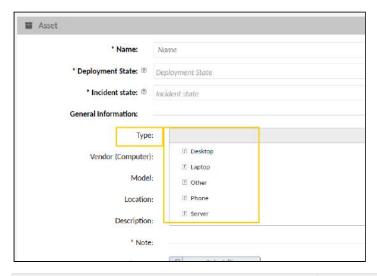
```
{
    'Key' => 'SectionOwner',
    'Name' => 'Owner Information',
    'CustomerVisible' => 1,
    'Input' => {
        'Type' => 'Dummy'
    },
    'Sub' => [
        {
            'Key' => 'OwnerOrganisation',
            'Name' => 'Assigned Organisation',
            'Searchable' => 1,
            'CustomerVisible' => 0,
            'Input' => {
                'Type' => 'Organisation'
            },
        },
        {
            'Key' => 'OwnerContact',
            'Name' => 'Assigned Contact',
            'Searchable' => 1,
            'CountDefault' => 1,
            'CountMin' => 0,
            'CountMax' => 25,
            'CustomerVisible' => 1,
            'Input' => {
                'Type' => 'Contact'
            },
        },
    ],
},
```

7.1.5.7 General Catalog

Allows you to store values from the General Catalogue (see page 146) . These are displayed as a filterable selection list.

Example configuration	Notes
| ```
{
  'CountDefault' => 0,
  'CountMax' => 1,
  'CountMin' => 0,
  'CustomerVisible' => 0,
  'Input' => {
    'Class' =>
'ITSM::ConfigItem::Computer::Type',
    'Translation' => 1,
    'Type' => 'GeneralCatalog',
  },
  'Key' => 'Type',
  'Name' => 'Type',
  'Searchable' => 1
},
``` | • **Class**: Reference to the General Catalog class, which defines the value range.<br>• **Translation**: Selection values are translated<br>• **ValueDefault**: Preselect selection value from list |

7.1.5.8   (Integer)

⚠️ The "Integer" type has been dropped and is replaced by the "**Text**" type with the corresponding RegEx configuration. Integer enabled the selection of integers (without comma) in a single selection list.

7.1.5.9   Organisation

Enables the storage of a valid customer (organisation) stored in the system and is the basis for assigning customers to devices. Selection via filterable selection list.

Example configuration	Notes
```{ 'Key' =>'CustomerCompany', 'Name' => 'CustomerCompany', 'Input' => { 'Type' => 'Organisation', }, },```	Organisation (company) of a customer

7.1.5.10   SLAReference

Allows the entry of valid SLAs on asset versions. (KIX Pro only). The entry is made via a filterable selection list. The behavior in input, search and display corresponds to that of the CIClassReference.

In the specified asset classes, the SLA stored on an asset is used when entering the SLA "SLA by Affected Asset" on a ticket. For this purpose, only one SLA may be entered.

Example configuration

```
{
  'CountDefault' => 0,
  'CountMax' => 1,
  'CountMin' => 0,
  'CustomerVisible' => 0,
  'Input' => {
    'Type' => 'SLAReference'
  },
  'Key' => 'AssignedSLA',
  'Name' => 'Assigned SLA',
  'Searchable' => 1
},
```

7.1.5.11   Text

Enables the entry of single-line, short, unformatted texts. Youcan specify a regular expression and/or a maximum length that the text must correspond to.

Example configuration	Notes
``{
'Key' => 'IPAddress',
'Name' => 'IP Address',
'Searchable' => 1,
'Input' => {
'Type' => 'Text',
'RegEx' => '^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$',
'RegExErrorMessage' => 'Please enter your full IP address: 123.123.123.123.',
'MaxLength' => 40,
},
},```	• **Size**: Length input field • **MaxLength**: Max. input length • **RegEx**: Regular expression on which input is checked • **RegExErrorMessage**: Display if reg. expression does not apply

7.1.5.12   TextArea

Enables the entry of multi-line, relatively long, unformatted texts. You can specify a regular expression
that the text must correspond to.

Example configuration

```
{
  'Key' => 'Description',
  'Name' => 'Description',
  'Searchable' => 1,
  'Input' => {
    'Type' => 'TextArea',
    'RegEx' => '\.$',
    'RegExErrorMessage' => 'Please enter full stop.',
  },
},
```

7.2   General Catalog

The General Catalog is a values catalog in which you can save a wide range of values for reuse elsewhere. It is often also referred to as an overall catalog or universal catalog. The General Catalog can be found in the *Assets > General Catalog* menu.

It enables you to save selection values for drop-down lists, such as attributes of asset classes or event states for assets (e.g. deployment state or incident state). It is also possible to save assembly series, model types, device types, building types, etc. You can reference these selection values in the class definition of an asset class so that they are available to agents when creating new assets.

Fig.: Overview of General Catalog

Example: By navigating to the menu *Assets > Asset Classes,* you create a separate asset class called "Vehicles" and save the selection values for the "Vehicle type" (Truck, Car, Van) in the General Catalog.

Fig.: Separate "Vehicles" Class with Different Values in General Catalog

In the class definition for the asset class "Vehicles", you then integrate a drop-down which references the vehicle types saved in the General Catalog.

Fig.: In Class Definition, Reference Values of "Vehicles" Class

Afterward, agents are able to create new assets of the class "Vehicles" and select from the vehicle types in the drop-down menu.

Fig.: When creating a new asset of the class "Vehicles", agents are able to select from the vehicle types saved in the General Catalog.

7.2.1 Adding an New Value to the General Catalog

The General Catalog comes supplied with a selection of values which are used by the system/the initial data. You can add other values or set superfluous values to "invalid". It is not possible to delete values from the General Catalog. The table in the Dashboard lists all the values contained in the General Catalog.

Fig.: New class "ITSM::ConfigItem::Vehicles::Type" with values ("Car", "Truck", "Van")

To create a new value, proceed as follows:

1. In explorer, navigate to Assets > General Catalog. In the content area, a table listing all the values created in the General Catalog opens.
2. In table, click "New Value". A form dialog for creating the new value opens.
3. Complete form (see table below) and set validity to "valid".
4. Click "Save" to save new value. The new value has now been created.

You can then create a new asset class in the menu Assets > Asset Classes, for example, and access the values saved here in the class definition.

Fig.: Providing values of the General Catalogue via the class definition

> ℹ **Note**
>
> If newly created values are to be translated by the system, the patterns for this must be stored in the *Internationalisation > Translations* menu.

To edit a value, proceed as follows:

1. In explorer, navigate to Assets > General Catalog. In the content area, a table listing all the values created in the General Catalog opens.
2. In table, click value to be changed. A form dialog for editing the value opens.
3. Amend information as required and click "Save" to apply changes.

Your changes take effect immediately. The agents may need to refresh their browser page to see the changes.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Class	Enter text of your choosing or select value from drop-down list. You can reference this designation in the source code of the asset class definition. If you enter a free text (e.g. "ITSM::ConfigItem::Vehicles::Type"), a new class is automatically created. You can create further values for this class after saving. It is available after saving in the "Class" selection field
Name	Enter selection value, e.g. "Truck" or "Car". Create several entries with same General Catalog class and different names to define a list of selection values.
Icon	Select icon for identifying the class. You can choose an icon from one of the provided libraries or upload a graphic file.
Validity	• valid: the class is valid and can be used. • invalid-temporarily: it is temporarily not possible to use the class. • invalid: the class is invalid and cannot be used.

7.3 Import/Export

The menu *Assets > Import/Export* enables you to simultaneously import multiple assets from a CSV file or export them to a CSV file. With this you can create or change all assets of a class in one go in an elegant way. This makes it easier and quicker to set up a new system and maintain assets that were created previously (mass update).

The imports and exports are performed using CSV mappings, which contain the assignment definitions for exchanging the data. KIX automatically creates a separate mapping for each asset class under the same name. Optionally, you can also create mappings using the console (see page 271) .

Fig.: KIX automatically creates a CSV mapping for all asset classes.

The table in the Dashboard lists all mappings created in the system. By checking off relevant option, select mapping with asset class into which you want to import assets or with assets you wish to export. Only then are the buttons "Import" and "Export" activated. You can only ever select one mapping.

	Name ▲	Comment	Validity	Changed at	Importstate
☐	[TD-ASC-001] Test class (auto-created map)	Automatically created (Definition create event)	valid	08/04/2021, 2:47 PM	
☐	Building (auto-created map)	Automatically created (console command)	valid	08/04/2021, 2:46 PM	
☑	Computer (auto-created map)	Automatically created (console command)	valid	08/04/2021, 2:46 PM	
☐	Hardware (auto-created map)	Automatically created (console command)	valid	08/04/2021, 2:46 PM	
☐	Location (auto-created map)	Automatically created (console command)	valid	08/04/2021, 2:46 PM	

Assets: Import/Export (9) — Filter in list — Import Export

Fig.: Overview of Mappings in Dashboard

> ⓘ **Notes**
>
> - In the SysConfig key "*ITSMCIAttributeCollection::Organisation::Content*" you can specify which attribute (ID, Number or Name) of an organisation is to be output during import/export. The default is "Number".
>   **Please note:** The attribute must be the same for export and subsequent import! This means that if the name of the organisation is imported, the SysConfig key should also be set to "Name". Otherwise you will receive an error message "Could not import Organisation: no Organisation ID found for [attribute (value)]". The value in the import file must then be changed accordingly.
> - If the name or a class-specific attribute is to be used instead of the ID for the objects referenced on the asset, you can specify this in the class definition (see Attribut Types in the Class Definition (see page 136) > Attribute type "CIClassReference")
> - KIX Pro users can have the asset numbers replaced with their own identifiers during import using a job (see Change Asset Numbers during Import)

7.3.1 Importing Asset Data

When importing data in the menu *Assets > Import/Export*, the data of several assets are read in from an existing CSV file. You can use an existing CSV file for this or create a CSV file beforehand via the export function, which you can then import back in an edited form.

When you create an asset class, KIX automatically creates a CSV mapping that contains the data structure of the asset class. The data structure in the mapping controls the assignment of source and target data. After changes to the class definition of an asset class, the associated CSV mapping is automatically updated if the associated SysConfig key

Content on this page:
- Conditions for the import (see page 154)
- Perform the asset import (see page 156)
- CSV import via script with extended options (see page 158)

"*ImportExport::CSVMappingAutoCreate###ForceCSVMappingRecreation*" is set to 1 (default: 0).

Fig.: Assets are read out from a CSV file and created under the asset class using mappings

When importing the data, existing asset data is overwritten or new asset data is added. The decisive element for determining which one applies is the asset ID in the CSV file.

- If the ID already exists in KIX, the corresponding asset data will be overwritten. This is advantageous if you want to make changes to multiple existing assets.
- If the ID does not exist yet in KIX, the asset will be created using the next sequential ID. This ID is generated internally by the system. This is advantageous if you want to create multiple new assets in KIX in one go.

7.3.1.1 Conditions for the import

For a successful import, it is imperative to comply with the conditions for the CSV file. Otherwise, the data cannot be correctly assigned and the import will fail.

- All texts must be set in " " (double apostrophe).
- The attribute "Name" must be specified.
- Multiple attributes must be separated by semicolons ";".
- The column names must be set in double apostrophe and separated by semicolons.
- Date fields expect the format: yyyy-mm-dd. If the format is, for example, dd.mm.yyyy, the asset is imported but the date is not processed.
- The data structures of the source (CSV file) and the target (KIX) must match exactly.

- If necessary, the MIME type of the file must be corrected in the browser. Permitted are: application/ vnd.ms-excel, text/plain, text/csv.


```
Service (auto-created map)_202306021147 - Editor

Datei  Bearbeiten  Format  Ansicht  Hilfe

"Number";"Name";"Deployment State";"Incident State";"SectionGeneral::1";"SectionGeneral::1::ParentService";"Sectio
"48116000004";"customerlogin1 service";"Production";"Operational";;;;"3-support process";;;;;;;;;;;;;;;;
"48116000003";"ServiceAssetHH";"Production";"Operational";;;;"3-support process";"1";;;;;;;;;;;;;;;;
"48116000002";"[TD-A-102] Service 2";"Production";"Operational";;;;"1-existential";;;;;;;;;;;;;;;;
"48116000001";"[TD-A-101] Service 1";"Production";"Operational";;;;"1-existential";;;;;;;;;;;;;;;;
```

Fig.: Example CSV file for import/export

Furthermore, please note:

Attribute	Conditions and notices
Number	Specification of the asset ID • The ID must be mandatory. • Import of new assets: The asset ID specified under "Number" must not yet exist in KIX and must not be "0". • Import of existing assets: The asset ID specified under "Number" must be identical to the asset ID in KIX. • If a "Number" in the CSV file is identical to an existing asset ID in KIX, the asset data will be overwritten (update of asset data). • If the "Number" and the Asset ID do not match, the asset data will not be updated, but a new import will take place.
Name	Name/designation of the asset. The name must be mandatory.
OwnerOrganisation	Name of the organisation. The organisation is only set if it already exists in KIX.
OwnerContact	KIX ID of the contact. The contact is only set if it already exists in KIX.

Attribute	Conditions and notices
General Catalog values like: SectionGeneral::1::Type	e.g. hardware • The specified values must already be created in the General Catalogue. • The designation must match the exact designation in the General Catalogue; translations are not possible.
Linking of assets like: SectionGeneral::1::ParentLocation	e.g. location, building, room • The asset number must be indicated without a preceding "A#". • If such dependencies are to be specified, the asset classes that do not • first import the asset classes that do not have dependencies, or • the dependent asset classes are not specified in the CSV file, in order to subsequently enter them manually.

7.3.1.2   Perform the asset import

For the asset import, the data structure of the imported CSV file must match the data structure in the mapping. This applies in particular to the column structure and column designations. Otherwise the import will fail. As an aid, you can export an asset class in which at least one asset is created.

1. Export CSV file (optional):
    1. In the Explorer, navigate to Assets > Import/Export. A table opens in the content area listing all CSV mappings created in the system.
    2. Select the desired asset class with a tick. Only now are the "Import" and "Export" buttons active.
    3. Click on "Export" and save the CSV file in your local file system.
        The exported CSV file has the necessary data structure for a correct allocation of source and target data.

2. Adding assets
    1. Open the exported CSV file and use it as a template for structure and layout.
        Manually add more assets to the CSV file (2) or make changes to the exported assets (1).

Please note the above conditions

> ✅ **Tip**
>
> If you would like to simultaneously import multiple assets, give each asset a numerical ID that is as high as possible, e.g. 9999999999. All assets can have the same ID. When importing the data, KIX will detect that this ID does not exist yet and will create the asset sequentially under the next available ID. As a result, you do not need to monitor the sequence of the IDs yourself.

2. Save your changes to the CSV file.

> ⚠️ **Notice**
>
> Microsoft Excel removes the inverted commas when saving the CSV file. We therefore recommend using Libre Office. Libre Office also offers the possibility of specifying filter options in "Save as...". to specify filter options.
> Before importing, check the CSV file for a correct structure (e.g. with a text editor).

3. Import CSV file:
    1. In the Explorer, navigate to Assets > Import/Export. A table opens in the content area listing all CSV mappings created in the system.
    2. Click on "Import" in the table header. A file selection dialogue opens.
    3. In this dialogue, navigate to the directory in which the CSV file with the asset data is located and select it with "Open". The data import starts.
    4. If the import was successful, a success message (green) is displayed.
    5. If the file type does not match the expected type or if the file contained errors, a corresponding error message (red) is displayed.

The imported assets are now available in the Assets menu of the agent portal.

> ✅ **Tip**
>
> KIX Pro offers the possibility to replace the asset numbers generated by KIX with own identifiers during import. This is done with a job of the type "Asset".

7.3.1.3   CSV import via script with extended options

The import of asset data is also possible via command line and the tool https://github.com/kix-service-software/kix18sync. Both CSV data without explicit attribute mapping in the header line and those with can be used. Please note that the header line must contain the technical identifiers and field indices as generated by the asset CSV export via GUI.

If you have a CSV file without suitable headers as attribute mapping, you can use the option " `--uam` " (use asset mapping) when calling the script (kix18.CSVSync.pl[9]). A column-to-attribute mapping is then assumed, which must be contained in the configuration file used (see example https://github.com/kix-service-software/kix18sync/blob/master/config/kix18.CSVSync.cfg). The file to be imported must not contain a header line. All lines are interpreted as asset data.

Example configuration for the "uam" option on the asset class computer

```
/opt/kix18sync$ ./bin/kix18.CSVSync.pl --config ./config/kix18.CSVSync.cfg --ot Asset
--if ./sample/AssetData04_Computer_Sample.csv --ac Computer --uam
```

For data imports from external sources and updates in the KIX CMDB, sometimes no KIX asset numbers are available. If you use unique asset names, you can then use the option " `--anl` " (asset name lookup) option. This option allows you to search for asset names if no or an invalid asset number is specified.

Example configuration for the option "anl" on the asset class computer

```
/opt/kix18sync$ ./bin/kix18.CSVSync.pl --config ./config/kix18.CSVSync.cfg --ot Asset
--if ./sample/AssetData04_Computer_Sample.csv --ac Computer --anl
```

---

[9] https://github.com/cape-it/kix18sync/blob/master/config/kix18.CSVSync.cfg

7.3.2 Exporting Asset Data

The data export function provided under the menu *Assets > Import/Export* generates a CSV file listing all assets of an asset class in the form of a table. You can save the CSV file in your file system and

- manually edit or add assets, in order to then import them using a mass update
- save it locally as a data backup
- use it as a template for importing data import the asset data into other systems
- read the asset data into other systems.

Fig.: All assets of the "Software" class were exported to a CSV file.

To export the assets of a particular class to a CSV file, proceed as follows:

1. In explorer, navigate to *Assets > Import/Export.* In the content area, a table listing all the CSV mappings created in the system opens.
2. By checking off relevant option, select mapping of asset class you wish to export. Only then are the buttons "Import" and "Export" activated.
3. In table, click "Export" to generate CSV file. Depending on which browser you are using, a dialog for opening and/or saving the CSV file opens.

The exported CSV file contains the data of all assets from the selected asset class in the form of a table. import the data into other systems.

> ✅ **Tips**
>
> - Use the export function to create local data backups, e.g. prior to a mass update. You will then always have a copy of the data available in case of emergencies.

- If you want to export the name or a class-specific attribute instead of the ID for the objects referenced on the asset, you can specify this in the class definition (see Attribut Types in the Class Definition (see page 136) > Attribute type "CIClassReference")

7.3.3 Using Mappings via the Console

You can have CSV import/export mappings created manually or automatically for all existing asset classes. You can then use these to import and export asset data via console commands.

Fig.: Using the console for import/export

Navigate to menu *System > Console.* Execute one or more of the following commands there as required:

Command	Description
`Admin::ImportExport::ListMappings`	Lists all mappings in the system.
`Admin::ImportExport::AutoCreateMapping`	Creates a new mapping for **all** asset classes, whereby the mapping ID is permitted to change. **Please note:** Mappings will only be newly created and updated if the key "ImportExport::CSVMappingAutoCreate###Fo rceCSVMapping Recreation" has been set accordingly in the SysConfig (default: "No").
`Admin::ImportExport::AutoCreateMapping --class ClassName`	Creates a new mapping for the asset class with the class name "ClassName".

Command	Description

1. `Admin::ImportExport::CreateMapping --file /path/to/file.json`
 	
2. `Admin::ImportExport::CreateMapping --file /path/to/file.json --replace`
	1. Creates a new CSV mapping for the export/import of asset data (supplementary to the default mapping), e.g.:
	`--file /tmp/CreateImportExportMapping.json`
	
	2. Removes or replaces an existing mapping with the same name, e.g.:
	`--file /tmp/CreateImportExportMapping.json --replace`
	
	Can be used to perform e.g. partial config item updates or combined asset imports of technical device data (file1.csv) and cost centre data (file2.csv).
	
	The mapping description must be available as a JSON file, e.g.:

Example: Mapping

```
{
  "A Sample Mapping Name":
  {                # Mapping Name
    "Charset": "UTF-8",
# Character set encoding (required)
    "ColumnSeparator": "Semicolon",
# Separation drawing for columns
    "IncludeColumnHeaders": "1",
# Export with column headings? (0 =
no; 1 = yes)
    "CountMax": "10",
# Max. number of array elements
    "ClassName": "Computer",
# Asset class name
    "DefaultDeploymentStateID":
"15",        # Default Deployment
state (ID)
    "DefaultIncidentStateID": "1",
# Default Incident state (ID)
    "DefaultName": "",
```

Command	Description

	```"EmptyFieldsLeaveTheOldValues": "1", # Transfer of old data from previous version for partial update "IdentifierAttribute": "Number", # Column serving as identifier column "CSVRowMapping": [# Contains the column mapping with identifier column "Number", "Name", "DeplState", "InciState", "SectionGeneral::1", "SectionGeneral::1::Type::1", "SectionGeneral::1::Vendor::1"] } }```
	
	Further attributes or their column names can be obtained from the column labels of the default mappings.
	
	Call in the backend e.g. with:
	
	```I have no name!@1234f56f7890:/opt/ kix$ ./bin/kix.Console.pl Admin::ImportExport::CreateMapping --file /tmp/ CreateImportExportMapping.json -- replace```

> ⓘ **Please note**
>
> The SysConfig provides a number of different configuration options for the mapping. All the settings begin with "*ImportExport::CSVMappingAutoCreate###[…]*". Event registration begins with "*ITSMConfigItem::EventModulePost###910-CSVMappingAutoCreate*".

7.4   Configuration settings

You can influence the behavior of the CMDB or asset management through numerous configuration settings. Here is an overview of the most important:

Content on this page:

7.4.1   Tracking link types

Configuration key	ITSM::Core::IncidentLinkTypeDirection

Defines which link types are followed in which direction in order to propagate the incident status of a disturbed asset to dependent, linked assets.

The configuration contains N relevant link types and directions as key-value pairs:

- Key: link type
- Value: Link direction (Source | Target | Both).

If an asset is set to "Incident", the link types configured here are tracked in order to also set the relevant assets to "Incident". However, if such an asset is linked to other, non-disturbed assets, the asset in question is not set to "Incident", but to "Warning".

⚠️The configuration is changed by manually executing the console command " `Admin :: ITSM :: IncidentState :: Recalculate` " to update all incident statuses.

> ✅ **Tip**
>
> The simulation of the asset graph is generated in the backend and delivered in the form of a JSON response. KIX uses this response to display the graph in the frontend. If necessary, you can couple your own automatisms to the REST API in order to generate the simulation for your purposes.

Example configuration

```
{
  "DependsOn":   "Both",
  "ConnectedTo": "Target",
  "Includes":    "Source"
}
```

7.4.2 Define usage status

Configuration key	GeneralCatalogPreferences###DeploymentStates

Defines the types of use status (productive, post-productive, pre-productive).

Agents must specify a deployment state when creating or editing assets. This describes the position of the asset within its life cycle.
You can create further individual deployment states in the General Catalogue and assign them to the status type (menu Assets > General Catalog).

Note: Assets (especially services) of the status type "post-productive" are not visible in the Asset Module and can only be found via the Complex Search (see also: Service Contracts).

KIX is initially delivered with the following usage statuses:

Deployment state	State type
Expired	productive
Retired	post-productive
Planned	pre-productive
Repair	productive
Maintenance	productive
Inactive	post-productive
Pilot	productive

Deployment state	State type
Production	productive
Review	productive
Test/QA	pre-productive

Example configuration

```
{
  "Class": "ITSM::ConfigItem::DeploymentState",
  "Data": {
    "postproductive": "postproductive",
    "preproductive": "preproductive",
    "productive": "productive"
  },
  "Desc": "Deployment State Type.",
  "Label": "Deployment State Type",
  "PrefKey": "Functionality"
}
```

7.4.3 Define assignments and visibility in the SSP

Configuration key	AssignedConfigItemsMapping

Controls

1. the assignment of assets to contacts or organisations in detail views or the sidebar "Assigned Assets" **as well as**
2. the visibility of assets in the SSP (selectable in the "Affected Assets" field and the "Assets" module).

For more information on configuration, see Control visibilities in the SSP.

Configuration key	AssignedObjectsMapping

Controls which tickets, communication steps (articles) and FAQ articles are displayed in the SSP.

For more information on the configuration, see Control visibilities in the SSP.

7.4.4 Automatic linking of ticket and asset | service

Configuration key	Ticket::EventModulePost###ITSMConfigItemLinkAdd

Automatically creates a link between ticket and asset or between ticket and service as soon as a change is made to the dynamic field of type "Asset reference".

This event only creates links. Links can only be deleted manually.

Example configuration Affected asset

```
{
  "Event": "(TicketDynamicFieldUpdate_AffectedAsset)",
  "LinkType": "RelevantTo",
  "Module": "Kernel::System::Ticket::Event::ITSMConfigItemLinkAdd"
}
```

Example configuration Affected service

```
{
  "Event": "(TicketDynamicFieldUpdate_AffectedServices)",
  "LinkType": "RelevantTo",
  "Module": "Kernel::System::Ticket::Event::ITSMConfigItemLinkAdd"
}
```

7.4.5 System Monitoring

Configuration key	PostMaster::PreFilterModule###****-SystemMonitoring

Configuration for processing incoming emails for System monitoring (see page 736) (Focus: ITIL Practice "Monitoring & Event Management").

```
{
  "AffectedAssetName": "AffectedAsset",
  "CloseActionState": "closed",
  "CloseNotIfLocked": "0",
  "ClosePendingTime": "172800",
  "CloseTicketRegExp": "OK|UP",
  "CreateChannel": "note",
  "CreateSenderType": "system",
  "CreateTicketQueue": null,
  "CreateTicketSLA": null,
  "CreateTicketState": "new",
  "CreateTicketType": "Incident",
  "DefaultService": "Host",
  "DynamicFieldContent::Article": null,
  "DynamicFieldContent::Ticket":
"SysMonXHost,SysMonXService,SysMonXAddress,SysMonXAlias,SysMonXState",
  "FromAddressRegExp": "sysmon@example.com",
  "Module": "Kernel::System::PostMaster::Filter::SystemMonitoringX",
  "NewTicketRegExp": "CRITICAL|DOWN|WARNING",
  "StopAfterMatch": "1",
  "SysMonXAddressRegExp": "\\s*Address:\\s+(.*)\\s*",
  "SysMonXAliasRegExp": "\\s*Alias:\\s+(.*)\\s*",
  "SysMonXHostRegExp": "\\s*Host:\\s+(.*)\\s*",
  "SysMonXServiceRegExp": "\\s*Service:\\s+(.*)\\s*",
  "SysMonXStateName": "SysMonXState",
  "SysMonXStateRegExp": "\\s*State:\\s+(\\S+)",
  "ToAddressRegExp": ".*"
}
```

7.4.6 Search for assets (Supplement System Monitoring)

Configuration key	TicketAutoLinkConfigItem

Enables the targeted search for assets as a supplement to the system monitoring integration via mail filter. The search is based on asset-identifying information stored in articles (especially emails) or fields, such as IP addresses or FQDNs. The specific attributes of the asset classes are used as search criteria and searches are only carried out in matching asset classes (Focus: ITIL Practice "Monitoring & Event Management").

For more information and SysConfig keys on this, see: System Monitoring .

8 Automation

The menu *Automation* enables you to create and manage automated processes in KIX.

Notifications

KIX enables you to configure notifications so that, for example, agents are notified via email when there are new tickets in their teams or new articles for their tickets. The settings for doing so can be configured under *Automation > Notifications.* Under the option "My Teams" in their preferences, agents can select the teams for which they would like to receive notifications. You can also pre-configure which teams have been set for an agent under "My Teams" in the Admin module under *User Management > Users*.

Notification Template

You can create an HTML template for notifications. With this template you design the appearance of the emails sent by KIX. It is thus possible to adapt the notifications to your corporate identity. The basic HTML structure for a template is already available in the *Automation> Notification template* menu. You can adjust these as required.

Jobs

The menu *Automation > Jobs* enables you to configure event-based and time-based jobs that KIX executes automatically. Examples of such jobs are anonymizing tickets after a certain period of time in order to comply with data protection regulations, archiving closed tickets, or moving tickets to a different team once assigned a certain state.

8.1   Jobs

KIX enables you to configure both simple and assembled time-based and event-based tasks (actions) using a graphical user interface without needing to use a script language. KIX executes these tasks automatically.

This means you can, for example,

- automatically delete the spam in your junk folder (team) once a week
- automatically close all tickets with the state "pending successful close" every Monday at 5:30 a.m.
- automatically remove the customer assignments for all tickets that have been closed for at least 6 months (compliance with GDPR regulations)
- create reports automatically.
- etc.

Jobs (formerly Generic Agents) are created and configured in the menu *Automation > Jobs.* The table in the content area lists all jobs created in the system along with their control parameters.

Jobs are executed by name. If jobs are triggered by the same event, their execution can be controlled by assigning corresponding names (alphanumeric).

Fig.: The Dashboard lists all jobs created in the system

Jobs initially delivered in KIX Start:

Job	Description	Notes
KIX Field Agent - Mobile Processing Rejected	This job is required for the Field Agent App. It enables users to return a rejected ticket to the team so that it no longer appears in the user's tasks.	A possibly blocked ticket could prevent the processor from being set. To avoid this, the actions are initially configured in the following order: 1. Set lock status: unlock 2. Set agent: 1 (admin)

Job	Description	Notes
Customer Response :: reopen from pending	The job changes the status of a ticket from "waiting" to "open" in case of a customer feedback via Self Service Portal.	Tickets are often set to a waiting status when a response from the customer is expected. If the customer responds at a later time via Self Service Portal, the job sets the status of the ticket to "open" and thus raises the relevance of the ticket. In this way, the ticket is not forgotten and can be further processed or closed. Initially, at least 5 minutes must pass from the creation of the ticket to the response before the status is changed. You can change this time span if necessary. If the filter parameter "Created on" is removed, the ticket will be set to "Open" with every reply.
Periodic Reports	The job determines the reports to be created periodically every 15 minutes and generates them in CSV format. It forms the basis for the up-to-dateness of the statistics in the Home Dashboard.	You can adjust the execution times in the job. **Important**: Chart widgets - and thus the statistics in the Home Dashboard - can only evaluate reports with output format CSV. In KIX Pro, JSON is also possible. Other output formats are not supported.

KIX Pro also comes with pre-prepared jobs that you can amend as required:

- Synchronization of login and customer data from an LDAP directory service (Active Directory)
- Anonymization of ticket data
- Setting of default values for the fulfillment time
- Set the incident status of a ticket for all affected assets
- Setting time stamps for SLA compliance
- etc.

8.1.1 Zoom View for a Job

The zoom view enables you to view the details of a job, edit a job, and manually start a job.

Navigate to *Automation > Jobs.* In the content area, a table listing all the jobs created in the system opens. In table, click job to open associated zoom view.

Fig.: Zoom View for a Job

The zoom view of a job contains multiple lanes showing the following information:

Lane	Description
Tab Lane	• Tab "Job Information": Master data for the job, such as validity, date created, and date changed. • Tab "History": Tracking of execution and download of job log files (see below)
Execution Plan	Shows the time parameters and event parameters that trigger the execution of the job.
Filter	Shows the filter criteria for the job.
Actions	Shows the actions that are executed using the job.

The buttons in the title bar of the zoom view enable you to make changes to the job or start the job manually.

Button	Description
New Job	Opens a form dialog for creating a new job.
Edit	Opens a form dialog for changing the job parameters.
Run	Click this button to manually start job.

8.1.1.1 History

KIX logs the execution of jobs. You can view the logs as a log file in the job history. This allows you to track the execution of jobs and find valuable clues for troubleshooting.

Click on a table entry to display the log information or click on the icon in the "Log Download" column to download the log file.

Fig.: Display of error information in the history of a job

8.1.1.2 Set log level

Configuration key	Automation::MinimumLogLevel

You can specify the level at which information should be logged and recorded in the log.

This is done in the SysConfig key "*Automation::MinimumLogLevel*" (Menu: *System > SysConfig*).

The following values are possible:

- **Error**: Only error messages are logged (default).
- **Debug:** All log information is captured.

⚠ **Please note**

Some macro actions (e.g. KIX Pro: "LDAP 2 Contact") contain a debug option. If you activate debugging, "*Automation::MinimumLogLevel*" must be set to " `Debug` ". Otherwise, the debug information provided by the Macro Action will not be captured and will not be displayed in the job history.

8.1.2 Creating or Editing a Job

You can configure KIX to run automatic tasks (jobs) on certain time and event-based events. For example, you can configure a job that deletes all tickets in the "Junk" team every Monday at 6:30 a.m. to clean the system from spam mails.

In one job, several consecutive actions can be executed, so that, for example, after deleting the junk mails, a new article with a completion note is generated.

When configuring jobs, you can use KIX Placeholders (see page 750) and Dynamic Fields (see page 274) . Invalid or temporarily invalid dynamic fields are not taken into account during job execution.

The creation of a job is done step by step. To move to the next step or to switch between steps, click on the small blue arrow buttons or on the blue dots in between. Please click on the "Save" button only at the end, when you have entered all job parameters.

To create a new job, proceed as follows:

1. In explorer, navigate to *Automation > Jobs.* In the content area, a table listing all the jobs created in the system opens.
2. In table, click "New Job". A form dialog for creating the new job step by step opens (see below).
   - **Step 1** (Job Information):
     Set up job by selecting type of job and assigning job a meaningful name (mandatory fields). Set validity to "valid" and optionally record a comment
   - **Step 2** (Execution Plan):
     Set the parameters that trigger the job, e.g. a specific time and/or a specific ticket event.
   - **Step 3** (Filter):
     Specify which tickets the job will be applied to. Select the parameters that the ticket should have so that the job is applied to it.
     For example, select the filter "Team" and the value "Junk" to apply the job only to tickets of the Junk team.
   - **Step 4** (Actions):
     Determine which tasks are to be carried out with the job and in which order. Pay attention to a sensible sequence. The tasks are processed in the order in which they are specified here - from top to bottom.
3. As the last step, click "Save" to save new job. The job has now been created, is active immediately and will be executed in line with its configured settings.

Note: To be able to duplicate a job, you need KIX Pro.

To edit a job, proceed as follows:

1. In explorer, navigate to *Automation > Jobs.* In the content area, a table listing all the jobs created in the system opens.
2. In table, click job to be edited. The zoom view for the job opens.

3. Click "Edit" in title bar of opened zoom view. A form dialog for editing the job opens (see below). Use small blue arrow buttons or dots to move to next step or jump between steps.

4. As the last step, click "Save" to save your changes. The job has now been changed, and if it has been set to "valid" it will be active immediately.

To delete a job:

1. In the Explorer, navigate to *Automation > Jobs*. A table opens in the content area listing all the jobs created in the system.

2. Place a tick in front of the job to be deleted. You can select several jobs.

3. Click on "Delete" in the table header. The button is only active when at least one job is selected.

4. Answer the security question with "Yes" to start the deletion process. All actions, sub-macros and execution plans are deleted recursively.
   However, if internal references still exist, the job cannot be deleted. You will then receive a corresponding message.

As an alternative to deletion, you can set a job to "invalid" or "temporarily invalid". Then the job is not triggered. An invalid job can be reconfigured at any time and set to "valid" again in order to reactivate it.

> ✅ **Tip**
>
> With the console command
> `Console::Command::Maint::Automation::DeleteNotReferencedMacros` you can delete macros that are not used (menu *System > Console*). This deletes all macros that are not referenced by any job and that are not sub-macros of other macros. Individual macros can be excluded from deletion. To do this, enter the parameter `ignore-macro-id` - several times if necessary.

8.1.2.1 Configure the job

Job Information (Step 1)

Here you store general information about the job and define the type of job.

Field	Description
Type of Job	Select the type of the job. The job type defines on which object the job will be executed: • **Asset** (KIX Pro): When importing assets, this job adapts the asset numbers to its own number range. • **Contact**: The job is applied to contacts. For example: create a ticket when a certain date is reached on the contact (start an offboarding process when an employee leaves). • **Reports:** The job automatically creates reports based on an existing report definition. Please Note: The statistics in the Home Dashboard only evaluate reports in CSV or JSON output format. • **Synchronisation:** The job syncs data (e.g. KIX contacts synced with contacts from LDAP/AD in KIX Pro). • **Ticket:** The job is applied to tickets (e.g. deleting the junk folder, moving tickets to selected queues, etc).

Field	Description
Name	Enter a meaningful name for the job. The sorting of the jobs and their order of execution is done according to their names (alphanumeric). This means that the execution sequence of jobs that are triggered by the same event can be influenced by their names. Example: • Scenario: 2 jobs that both react to "TicketStateUpdate" and create an article (Job 1 has name "02_TestJob", Job 2 has name "01_TestJob"). • Result: The article of job 2 (01_TestJob) is created first, as the name begins with "01".
Comment	Optionally save a comment. If required, record what the job does so that you can keep an overview if there are multiple jobs.
Validity	• **valid**: The job is active and will be triggered based on a point in time or an event in line with the configured settings. • **invalid/invalid-temporarily:** The job is (temporarily) inactive and will not be executed.

Execution Plan (Step 2)

Here you specify when the job is triggered (time or event).

Field	Description
Weekday(s)	Select the days of the week on which the job should be executed. In addition, a time must be specified.
Time	Select the time(s) at which the job should be executed. In addition, the day of the week must be specified. For daily execution, all weekdays must be selected.
Events	Select the events that are to trigger the job. The job will be triggered as soon as one of the events specified here occurs. The standard events (e.g. ArticleCreate) and an update event for each dynamic field are available for selection (TicketDynamicFieldUpdate_NameDesDynamicField).

> ⓘ **Please note**
>
> If you have not selected a point in time or an event, then the system will not automatically start the job. However, you can start it manually at a time of your choosing by clicking "Run" in the zoom view.

Filter (Step 3)

With filters, you restrict the execution of the job based on certain criteria. Here you define which conditions must be fulfilled for the job to be executed (e.g. all tickets in the "Junk" team).

You can combine the filters. The individual filter blocks are OR-linked. The filters within a filter block are AND-linked. By means of the buttons ➕➖ you add or remove further filter blocks.

Example: Job is executed if

- the ticket has the team "Service Desk" AND the affected asset "Computer 1" stored on it
  OR:
- the ticket contains the team "Monitoring" AND the affected asset "KIX18".

The filters are specified analogously to the logic in the complex search (see user manual):

Column	Description	Hints
1	Filter attributes of the object (according to the selected job type)	Which filter attributes are available for selection depends on the object (ticket, asset, contact, etc.) selected in step 1 under "Job type". In addition to the filter attributes of the object, Dynamic fields (except type "Checklist") are also available for selection. ℹ Only searchable attributes are available for selection in asset jobs. If necessary, set the missing attribute in the class definition to `'Searchable' => 1`.
2	Search operators (see page 884) (e.g.: contained in, begins with, ends with, is equal to, etc.)	The available search operators depend on the selected filter attribute or dynamic field. For dynamic fields of the type text and textarea the wildcard search is additionally available (search with * as placeholder for unknown text passages). For date/time-based filter attributes, relative time specifications can be defined (e.g. created within the last 24 hours).
3	Formular fields and values	KIX placeholders (e.g. <KIX_CURRENT_UserID>) can also be specified as values, e.g. to restrict the job to the current user.

If you specify multiple filters, the filters will be linked by a logical AND operation. Thus, you can specify that the job is applied to all tickets that have the type "unclassified" AND the status "closed" AND are from the team "support".

Hints:

- If dynamic fields set to "invalid" are specified, they will not be used in the execution.
- If you have selected an article-related event in the execution plan (step 2) (e.g. "ArticleCreate"), the filters "Channel" and "Sender Type" are mandatory fields. With these you further restrict the filter. Please select the options in such a way that you prevent "endless loops" that could occur when sending notifications and creating articles at the same time.

Actions (Step 4)

Here you specify which actions are to be carried out with the job in which order.

Field	Description
Macro	Corresponds to the selected job type (step 1) and cannot be changed.

Field	Description
Action	Select which tasks (Macro Actions) the job should execute. Several actions can be executed in one job. They are processed in the specified order - from top to bottom. Actions can be nested. Therefore, pay attention to a logical order. In the Macro Actions overviews, you will find instructions on how to use the individual actions. After you have selected an action, further input fields are displayed. These vary depending on the selected action. For example, the action "Set agent" requires the agent's name to be entered in order to set him as an agent on the ticket. Further informations: • Overview of Macro Actions for KIX Start (see page 804) • Overview of Macro Actions for KIX Pro (see page 819) • Macro Actions (see page 802) • and the usage examples of Macro Actions Using the buttons located at the edge, you can add further actions, remove actions, or move the order of the actions using drag & drop:

➕	Adds additional actions.
➖	Removes the action.
🔽	Collapses or expands the input fields for an action.
🔼🔽	Changes the sequence of the actions. Click on double arrow, hold down left mouse button, and drag action to different position. Place action in a dark gray area.

Field	Description
Skip	Check off this option if the action should (temporarily) be skipped. The job will be executed without this action. This enables you to exclude specific actions from the execution without needing to reconfigure the whole job.

8.2   Notifications

The menu *Automation > Notifications* enables you to specify for which ticket events email notifications should be sent to which agents and specify what these notifications should contain.

For example, you can configure that the agent set as the owner on a ticket should be notified when the ticket state changes. If this notification is set to "valid", an agent will always be notified when there is an updated state for a ticket that they are the owner of, and when this ticket is located in a team that they have saved under "My Teams". If the message has been created in multiple languages, KIX can take into account the agent's language here.

KIX comes supplied with a selection of ticket notifications. You can amend these as required and add additional notifications.

Fig.: "Notifications" Menu

You can set the sender information for notifications (see page 193) . To do this, navigate to the *System> SysConfig* menu and define the sender information in the following keys:

- *NotificationSenderEmail:* Email address; Sender of the notification (e.g .: kix-notifications@example.de)
- *NotificationSenderName:* Name / designation of the sender (e.g .: "KIX message")

8.2.1   Zoom View for Notifications

You can view the details of a ticket notification. Navigate to *Automation > Notifications*. A table opens in the content area that lists all notifications stored in the system. Click on a notification in the table to open its detail view. Click on "Edit" in the title bar of the detailed view if you want to make changes to the notification.

The buttons in the title bar of the detail view allow you to edit and create new ticket notifications

Fig.: Zoom View of a Ticket Notification

The zoom view of a notification contains multiple lanes showing the following information:

Lane	Description
Notification Information	Reference data of the notification, as well as the events that cause the message to be sent.
Filter	Lists the filter conditions to which the sending of the message is linked.
Recipients	Lists the recipients of the message.
Notification Methods	Lists additional recipients of the message.
Notification Text	Displays the message text in the relevant language.

8.2.2 Creating or Editing a Ticket Notification

You can configure KIX such that automatic emails are sent to agents if certain ticket events occur.
This means that agents can be automatically informed about various ticket events.

Fig.: Editing a Ticket Notification

To create a new ticket notification, proceed as follows:

1. In explorer, navigate to *Automation > Notifications*. In the content area, a table listing all ticket notifications saved in the system opens.
2. In table, click "New Notification". A form dialog for creating the notification (in several languages if applicable) opens.
3. In form, configure ticket notification (see table below) and set it to "valid".
4. As the last step, click "Save" to save your entries.

The new ticket notification has now been created and will be sent in line with the settings you have onfigured.

To edit a ticket notification, proceed as follows:

1. In explorer, navigate to *Automation > Notifications.* In the content area, a table listing all ticket notifications saved in the system opens.
2. In table, click notification to be edited. The zoom view opens.
3. Click "Edit" in title bar of opened zoom view. A form dialog for editing the notification opens.
4. Change information/configuration as required (see table below) and as the last step, click "Save" to apply your changes.

The changes take effect immediately.

To delete a ticket notification, proceed as follows:

1. In explorer, navigate to *Automation > Notifications.* In the content area, a table listing all ticket notifications saved in the system opens.
2. Place a tick in front of the notification to be deleted. You can select several notifications as well.
3. Click on "Delete" in the table header. The button is only active if at least one notification is selected.
4. Answer the confirmation prompt. If you choose "Yes", the notification will be deleted and no longer be sent in the future. Notifications that have already been sent remain on the tickets.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Give the notification a meaningful name.
Show in agent preferences	The notification is sent to all agents selected as recipients. • **Activated:** The agents designated as recipients can specify in their personal settings whether they want to receive this notification. • **Deactivated:** Agents designated as recipients are required to receive this notification. They cannot opt out of receiving the notification in their personal settings.
Agent Preferences Tooltip	You can specify that the agents should be informed by a tooltip about the receipt of a Preferences notification. To do so, use this option to record the Tooltip text that should be shown. If you have not saved a tooltip, the agent will not be informed that a notification has been received.
Validity	• **valid**: The notification is active and will be sent as soon as the defined event occurs. • **invalid/invalid-temporarily**: The notification is inactive and will not be sent.
Events	Select the ticket events that should cause the message to be sent. If one of these events occurs, the message will be sent. All valid events saved in the SysConfig are available for selection. You can select more than one option. **Please note:** When selecting article events, the specification of the filters "Channel" and "Sender Type" is obligatory. This prevents "endless loops" when sending notifications with simultaneous creation of articles.

Field (selection)	Description
Filter	You can use filter conditions to narrow down the sending of the message to tickets that have certain characteristics. An example could be when a new ticket (event) arrives from a certain customer (filter). The filters are specified in the same way as the logic in the complex search. The use of KIX placeholders (e.g. <KIX_CURRENT_UserID>) is possible. All ticket-based filters and dynamic fields (except checklist type) are available for selection. The wildcard search is also available for dynamic fields of the type Text and Textarea (search with * as a placeholder for unknown text passages). If you specify several filters, the filters are linked to one another by a so-called logical AND link. **Please note:** • If dynamic fields set to "invalid" are specified, they will not be used in the execution. • If you have selected an article-based event under "Event" (e.g. "ArticleCreate"), the filters "Channel" and "Sender Type" are mandatory fields. You can use them to narrow down the filter further. Please select the options such that you avoid "endless loops" that could occur when sending notifications while simultaneously creating articles.
Recipients	Select the recipients of the message. A combination of recipient groups, individual agents, and role members can be selected.

Field (selection)	Description	
	Send to	Select the recipient or recipient group of the ticket notification (multiple selection possible).
		Messages are sent depending on the basic permissions:
		• **All agents wiht read permission for the ticket:** Only recipients with READ permission on the specific ticket receive a notification.
		• **All agents with update permission for the ticket:** Only recipients with READ and WRITE permissions on the specific ticket will receive a notification.
	Send to these agents	Select specific agents as the recipients of the message. All valid agents created in the system are available for selection (you can select more than one option).

Field (selection)	Description	
	Send to all role members	The message is sent to the agents who all have the selected roles. All valid roles created in the system are available for selection (you can select more than one option).
	Send despite out of office	Check off the checkbox if the message should also be sent to recipients with the state "out of office" (e.g. if they are on vacation).
	Once a day	For each ticket, a notification is only sent 1x per day.
	Create Article	If you check off this option, an article will also Article be created on the corresponding ticket when generating a notification. For article-based events, ensure that you do not configure any "endless loops".
		If you do not check off this option, the message will be sent but an article will not be created.

Field (selection)	Description	
Notification Methods	Additional recipients	Enter additional email addresses that should also receive the notification (optional). You can choose from all of the customer contacts stored in the system and you can also enter external email addresses manually, provided the addresses are valid. **KIX Pro:** In order to make the additional recipients variable, you can specify KIX placeholders for dynamic fields (see page 281) of the type "contact reference". This can be used, for example, to inform other contacts stored on the ticket about the closing of the ticket. The email address of the contact is always taken from the placeholder, so you can use the parameters _Short and _Value: • <KIX_TICKET_DynamicField_myDFContactReference**_Value**> : "Last Name, First Name" <email>.

Field (selection)	Description	
		• <KIX_TICKET_Dyna micField_myDFCont actReference_**Short** > : email A prerequisite for correct use is that a valid and syntactically correct email address is stored in the ticket in the dynamic field.
	Subject with Ticket number	If this option is selected, the ticket number will be sent to the additional recipients in the subject line.
Notification Text	You can store the notification text in different languages. Therefore the users receive the message in the language set for them. An editor is available for each language created in SysConfig ("*DefaultUsedLanguages*" key). With the selected "Content Type" you define the format of the notification: • **Richtext:** Notification text with HTML formatting. • **Plaintext:** Notification text without HTML formatting (e.g. for technical mails or interfaces) You can use placeholders in both the subject and the message text to place variable text fragments in the messages, such as the content of the ticket subject. E.g. Subject: "Update <KIX_CUSTOMER_Subject_64> [<KIX_CONFIG_Ticket::Hook> <KIX_TICKET_TicketNumber>]". E.g. Article text: "<KIX_AGENT_From> wrote: <KIX_AGENT_Body>	

8.2.3 Standard address for notifications

Configuration key:	Sender address:	NotificationSenderEmail
	Sender name:	NotificationSenderName
	Domain name:	FQDN

KIX can send internal notifications as soon as certain ticket events occur. You can configure the default sender name and address for these notifications.

Proceed as follows:

1. Navigate to the *System > SysConfig* menu.
2. Search for the keyword "notificationsender*". You will get the two relevant keys in the overview.
3. Select both keys with a check mark and click on "Edit".
4. Change the values in the keys:
    1. NotificationSenderEmail:
        - Enter the sender address from which notifications should be sent by default (e.g. "info@myhost.example.de").
        - If the frontend FQDN was entered in the setup wizard, it is adopted here (hash value).
        - It is possible to use a KIX placeholder for the domain name (<KIX_CONFIG_FQDN> or <KIX_CONFIG_FQDN_Frontend>). Then the domain is used which is stored in the SysConfig key "*FQDN*".
    2. NotificationSenderName:
        - Enter a freely definable sender name. The notifications are marked with this name by default (e.g. "KIX message").
5. Save your changes.

The ticket notifications sent by KIX will now be sent from the changed default address with the sender identifier you named.

Fig.: The Key for Default Sender Address

Fig.: The Default Sender Name

Fig.: The Key for Domain

8.2.4 Notification Template

The layout for ticket notifications is provided by the standard template supplied with KIX. You can adapt this standard template individually in order to achieve a uniform appearance for your corporate communication. Afterwards, ticket notifications will be sent in the layout you have configured.

The standard template can be found in the Automation> Layout Notifications menu. The displayed HTML source code forms the layout of the ticket notifications. You can change the text and style information. To do this, you need knowledge of HTML and CSS. You will find good help at https://www.selfhtml.org (German Website). Our support team will be happy to help you design the notification layout. For easier editing, you can copy the source code into an HTML editor (e.g. Notepad ++).

- Contents between `<style>` and `</style>` define the formatting of the individual elements on the page, such as page background, header and footer of the mail, display of headings and links, etc.
- Contents between `<body>` and `</body>` form the page content. The page content is grouped in individual containers. These containers are formed with `<div> [...] </div>`.
- Contents between `<! -` and `->` are comments. These notes are for your information and are not taken into account in the e-mail display. If you want, you can safely remove them.
- You should not change the lines `[% Data.Subject | html%]` and `[% Data.Body [....] %]`. These are the HTML placeholders for the email subject and the content of the email.

Click on "Preview" to check the configuration you have changed before saving.

Fig .: Example of an HTML template

The following source code defines the initial standard template. With this you can replace your own template if necessary and thus restore the starting point.

Initial template

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
  <style type="text/css">

    body {
      width: 100%;
      margin: 5px;
      padding: 0px;
      background-color: #efefef;
    }

    .bodyContent,
    .bodyContent * {
      color: #323232;
      font-family: Arial;
      font-size: 13px;
      line-height: 150%;
      text-align: left;
    }

    .bodyContent a:link,
    .bodyContent a:visited,
    .footerContent div a:link,
    .footerContent div a:visited {
      color: #1491D2;
      font-weight: normal;
      text-decoration: none
    }

    #templateFooter {
      background-color: #FFFFFF;
      border-top: 0px;
    }

    .footerContent div {
      color: #707070;
      font-family: Arial;
      font-size: 13px;
      line-height: 125%;
      text-align: left;
    }

    #links {
      background-color: #FAFAFA;
      border: 0;
      color: #008DD2
    }
```

```
#links div {
  text-align: center;
}

.bodyContent a {
  color: #008DD2;
}

.bodyContent h1,
.bodyContent h2,
.bodyContent h3,
.bodyContent h4,
.bodyContent h5,
.bodyContent h6 {
  color: #04537D;
  display: block;
  font-weight: bold;
  line-height: 100%;
  margin-top: 0px;
  margin-right: 0px;
  margin-bottom: 5px;
  margin-left: 0px;
}

.bodyContent h1 {
  font-size: 26px;
}
.bodyContent h2,
.bodyContent h3,
.bodyContent h4,
.bodyContent h5,
.bodyContent h6 {
  margin-top: 5px;
  margin-bottom: 5px;
}

.bodyContent h2 {
  font-size: 22px;
}

.bodyContent h3 {
  font-size: 18px;
}

.bodyContent h4 {
  font-size: 16px;
}

.bodyContent h5 {
  font-size: 12px;
}
```

```
    .bodyContent h6 {
      font-size: 12px;
    }

    .notifyHeader {
      background: #04537D;
      border-radius: 20px 20px 0px 0px;
      padding: 20px;
      line-height: 120%;
      color: #ffffff;
      margin: 0px;
      font-family: Arial;
      font-size: 18px;
      font-weight: bold;
      text-align: center;
    }
  </style>
</head>

<body>
  <center>
  <div id="templateBody" style="padding:0.5em;width:95%">

    <div valign="top" class="notifyHeader">
      <!-- THIS IS WHERE THE NOTIFICATION SUBJECT IS PUT -->
      [% Data.Subject | html %]
      <!-- EO THIS IS WHERE THE NOTIFICATION SUBJECT IS PUT -->
    </div>

    <div valign="top" class="bodyContent" style="background-color: #ffffff; padding:
15px; font-size: 13px;">
      <!-- THIS IS WHERE THE NOTIFICATION BODY IS PUT -->
      [% Data.Body
      .replace('<p>', '<div>')
      .replace('</p>', '</div>')
      .replace('<div\s+type="cite"[^>]*>', '<div type="cite" style="background:
#f5f5f5; border: 1px solid #e0e0e0 !important; color: #555; margin: 10px 0px 0px 0px;
padding: 10px;">')
      .replace('border:none;border-
left:solid\s+blue\s+1.5pt;padding:0cm\s+0cm\s+0cm\s+4.0pt', 'border: none; border-
left: 1px solid #f92 !important; color: #555; margin: 10px 5px 10px 0px; padding: 0px
10px;')
      %]
      <!-- EO THIS IS WHERE THE NOTIFICATION BODY IS PUT -->
    </div>
  </div>
  </center>
</body>

</html>
```

9 Communication

The menu *Communication* enables you to maintain and manage the communication channels for KIX.

Email

In the menu *Communication > Email* you can configure the incoming and outgoing of emails. Sending and receiving emails are configured separately in the submenus Outbox (see page 219) and Inbox (see page 201) . Inbox and Outbox can alternatively be set up in the Setup Assistant directly after the installation of KIX.

Under Email Addresses (see page 228) you create the system addresses. These are the sender addresses for emails sent by the teams and are used to distribute incoming emails to the respective team.

Via Email Filter (see page 231) , incoming emails can be distributed to the respective teams according to filter conditions and created as tickets there. In the process, ticket information such as status or priority can be automatically entered.

Webform

The menu *Communication > Webform* enables you to generate web forms (see page 253) that you can integrate into your website. Your customers can then use these web forms to submit ticket-related information such as error reports. These reports are created as new tickets in KIX.

9.1 Email

One of the main tasks of KIX is the management of incoming and outgoing communication, e.g. emails. Setting up email communication including distribution in the system is done in the menu *Communication > Email*.

KIX distinguishes between incoming and outgoing mail. Both are configured separately from each other. The inbox receives - like a house mailbox - all incoming emails. KIX empties this "house mailbox" and distributes the emails to the respective teams according to the given configuration. There, they are either created as new tickets - or in the case of replies - as a new item on the ticket. All outgoing emails are sent by KIX via the outbox - like via a mailbox.

Alternatively, the communication channels can also be set up in the setup assistant directly after installation. The setup assistant guides you through setting up your inbox and outbox.

Inbox

> In order for KIX to receive emails, you must set up one or more email accounts in KIX. The setup is done either in the setup assistant or in the Inbox submenu. KIX retrieves these email accounts at regular intervals and creates new tickets in the assigned team folders from the emails retrieved. You can specify how the distribution is to be carried out separately for each email account.

Email filter

> Email filters (postmaster filters) are used for the granular distribution of incoming emails based on mail header information. You can use them to process incoming emails in a targeted manner and distribute them to specific teams, for example, based on their sender or subject. A typical example of this is sorting out spam emails into the junk folder. The distribution of the inbox by means of email filters takes place independently of the set standard and thus has priority over the standard distribution.

Email address

In the submenu *"Email addresses"* you set up the system addresses. These are the email addresses through which the teams communicate with the outside world. Each team usually has its own email address, e.g. support@mustermann-gmbh.de.

First create the required email addresses and then assign the addresses to the respective teams in the menu *"Ticket > Teams"*. Each email address must match an incoming mail account configured in KIX or be retrieved from a collective mailbox.

Outbox

> In order for KIX to be able to send emails, you need to set up an mail server. The setup is done either in the setup assistant or in the outbox submenu. All outgoing emails are sent through this server.

> The sender / reply address of an outgoing email is always the address that is stored on the team from which the email was sent.

9.1.1 Inbox

The inbox in KIX is automated by the regular retrieval of email mailboxes. In order for KIX to know which mailboxes are to be retrieved, you must set up one or more email accounts in KIX. Retrieval can be from POP3 or IMAP accounts - both encrypted and unencrypted.

The initial setup of the inbox is done in the Setup Assistant (see page 58) . You can add and edit further email accounts in the menu "*Communication > Email > Inbox*".

The retrieved emails are created in the teams either as new tickets or as articles on an existing ticket. You can define how the distribution is to take place separately for each email account. If the incoming emails are to be distributed specifically on the basis of email headers, e.g. according to certain senders or keywords in the subject, you can configure this via email filters (see page 231) (menu "*Communication > Email > Email Filters*").

9.1.1.1   Create or edit an email account

Menu	KIX > Communication > Email > Inbox

You can create and manage email accounts for receiving incoming emails. KIX automatically retrieves these email accounts at regular intervals. The retrieved emails are created as new tickets or new articles in the respective teams. The distribution to the teams is done according to the option selected in the "Distribution" field or the configured email filters. Each email account can be configured separately.

> ⬣ **Important!**
>
> Regardless of the account type, **all** emails are retrieved from the mail server and **deleted**. If this is not desired, you must make the appropriate settings for this directly on the mail server. There is no possibility to leave emails on an account. Please note this especially if you want to test KIX!

Have the following information ready:

- Login information for the incoming mail server:
    - Login name of the email account
    - Password of the email account
    - Email server pickup location (host:port)
    - collection method with or without encryption (e.g. "IMAP" or "IMAPTLS")
    - for IMAP: directory where the incoming emails are stored (usually "INBOX")

To create a new email account, proceed as follows:

1. In explorer, navigate to *Communication > Email > Inbox.* In the content area, a table listing all email accounts saved in the system opens.
2. In table, click "New Account". A form dialog for creating the email account opens.
3. Complete form, assign account to team under "Dispatching", and set validity to "valid".
4. Click "Save" to save your entries.

The email account has now been created and the inbound emails will be created as tickets in the assigned team. If required, you can specify the distribution of the emails more precisely under *Communication > Email > Email Filter*.

After the setup, you can check the correctness of the data. To do this, click on "Retrieve now". If the retrieval fails, you will receive a notification message.

To edit an email account, proceed as follows:

1. In explorer, navigate to *Communication > Email > Inbox*. In the content area, a table listing all email accounts saved in the system opens.
2. In table, click email account to be edited. The zoom view opens.
3. Click "Edit" in title bar of opened zoom view. A form dialog for editing the email account opens.

Amend information as required and set validity to "valid". The changes take effect immediately.

To delete an email account, proceed as follows:

1. In explorer, navigate to *Communication > Email > Inbox.* In the content area, a table listing all email accounts saved in the system opens.
2. Place a tick in front of the email account to be deleted. You can select several accounts as well. The delete button is only active if at least one account is selected.
3. Click on "Delete" in the table header. The button is only active if at least one email address is selected.
4. Answer the security question with:
    1. "Yes": The email account will be removed from the system. Afterwards, KIX cannot retrieve any emails from this account.
    2. "No": The email account is not deleted and remains ticked.

The form dialog contains the following input fields, among others:

Field (selection)	Description
User name	The email account whose inbox you would like to retrieve emails from. For example: info@mailbox.org

Field (selection)	Description

Type	Transmission protocol for email retrieval.
	The selected protocol specifies the retrieval method. If necessary, find out about the differences between POP3 and IMAP beforehand.
	
	Supported protocols:
	
	• Without encryption: IMAP | POP3
	• with SSL encryption: IMAPS | POP3S
	• with TLS encryption: IMAPTLS | POP3TLS
	• for OAuth2 profiles: IMAPS_OAuth2 | IMAPTLS_OAuth2 | POP3S_OAuth2 | POP3TLS_OAuth2
	   Notes:
	     • Outlook.de[10] currently only supports IMAPS_OAuth2
	     • Currently, POP3 SSL/TLS (POP3S) support with OAuth2 authentication is required to connect Office365 mailboxes (see: https://support.nhs.net/2023/08/microsoft-365-alert-service-degradation-exchange-online-some-users-may-be-unable-to-access-their-mailbox-using-the-imap4-connection-method-within-exchange-online/).
	
	Recommendation: First test IMAP or POP3 <u>without</u> encryption. If the wrong option is selected, authentication will fail. A corresponding message is displayed.

Password	The password that gives you access to the email account.
IMAP Folder	Directory in which the incoming emails are located on the server. From this directory KIX fetches the incoming emails (usually the folder "INBOX" or "Inbox"). This option is only available if the retrieval is done from an IMAP account.
Host	Address for the incoming mail server pickup interface. Usually there is a separate host URL for each pickup method: e.g.: pop.mailbox.org or imap.mailbox.org A special port can be specified as well (Host:Port).

---

10 http://Outlook.de

Field (selection)	Description
Accept KIX Header	**If enabled,** information from the KIX email headers is applied (e.g. ticket type, team, priority, etc.). The email header is accepted, evaluated and the information it contains is processed, regardless of what is configured in the email account. **If not enabled:** The email header is discarded when the emails are received and remains unconsidered. **Please note:** The emails sent via KIX can be provided with a KIX header. If necessary, you can read these and use them for actions based on them. **But:** Under certain circumstances, the header information supplied by KIX may contradict the email handling you have specified and lead to undesired actions or system errors! ⚠️ **Therefore, activate this option only if you trust the KIX header (e.g. from other KIX systems) and know from which senders the retrieved messages originate.**

Field (selection)	Description

Dispatching	Select the principle according to which the retrieved emails are to be distributed:
	
	• **Default Queue (SysConfig)**
	  • The retrieved emails are distributed to the Postmaster default team (Default Queue). This is the team that is stored in the SysConfig key "PostmasterDefaultQueue". In the delivery state, the default team is "Service Desk". You can specify a different team in the SysConfig key.
	• **Recipient addresses (To, CC, etc.)**
	  • The distribution of retrieved emails is based on the email recipient.
	    • If the recipient address is created as an email address in the system, the ticket will be created in the team that is stored at this system address.
	    • If the address is unknown or the email cannot be assigned to a team, the ticket will be created in the Postmaster default team.
	• **Team**
	  • If this option is selected, you can specify a team in which the collected emails will be created as new tickets.
	  • All teams created in the system are available for selection. Teams in light grey are (temporarily) invalid. They are listed but cannot be selected.
	
	Further information on distribution: see Notes on email distribution

Validity	• **valid**: The emails will be retrieved and distributed to the assigned team. • **invalid/invalid-temporarily**: The inbox of the email account will not (or no longer) be retrieved, meaning no (further) emails will be received from it. Any emails that have already been retrieved have been saved as tickets in the system and can be processed.

9.1.1.2 Notes on email distribution

KIX checks the content of incoming emails for known information such as ticket number and message ID. Based on this information, KIX determines whether a new ticket or - in the case of a reply (follow-up) - an article is to be created on an existing ticket.

The search for a possible follow-up is carried out across all teams. If a follow-up exists, a new article is created on the corresponding, already existing ticket. If no follow-up is detected, a new ticket is created based on the distribution set on the account (see below).

Content on this page:

Identifying possible follow-ups

Menu	KIX > System > SysConfig

In KIX, the modules listed below check incoming emails for possible follow-ups. If necessary, you can deactivate/activate the modules and thus determine which email contents are to be checked.

Modul	Description	Standard
PostMaster::CheckFollowUpModule###0100-Subject	Look for a valid ticket number in the subject of the email. The ticket number must always be preceded by the configured TicketHook and the TicketHookDivider. Default is: "Ticket#..." Must be <u>active</u> so that the ticket number added in the subject, determined using an external reference, can be recognised as a follow-up.	active/valid

Modul	Description	Standard
PostMaster::CheckFollowUpModule###0200-References	Searches in the "Reply-To" or References header for the message ID, which is stored, for example, in an article of the system.	active/valid
PostMaster::CheckFollowUpModule###0300-Body	Searches for a valid ticket number in the email text (body).	inactive/invalid
PostMaster::CheckFollowUpModule###0400-Attachments	Searches the content of email attachments for a valid ticket number.	inactive/invalid
PostMaster::CheckFollowUpModule###0500-RawEmail	Searches the raw data (source) of emails for a valid ticket number.	inactive/invalid

You can define in the module parameters how the respective module is to proceed when a follow-up has been detected:

- **OnlyFirstMatch:**
  - if set (1): If several follow-up tickets were detected, only the first ticket is used.
  - if not set (0): If multiple follow-up tickets were detected, the email of each detected follow-up is processed accordingly.
- **StopAfterMatch:**
  - if set (1): No further call of subsequent modules will be made once a possible follow-up has been detected by the current module.
  - if not set (0): The next registered module is always called and its detected follow-ups are added to the list.

Fig.: Flow chart of the audit according to possible follow-ups

Extended check for follow-ups

Configuration key	PostMaster::PreFilterModule###888-ExtendedFollowUp

KIX is delivered with the SysConfig key *PostMaster::PreFilterModule###888-ExtendedFollowUp* activated by default. This filter makes it possible to read external identifiers (e.g. the ticket number) in incoming emails and assign possible follow-ups based on these.

Example: Identifier = 'Sender_A'.

An email is read in that comes from the sender "info@example.de". The subject is "[SMPL::#12345] New enquiry". The filter recognises from the sender that "Sender_A" is to be used and recognises the external reference with the number "12345" (→ *ExtendedFollowUp###SenderEmail* and *ExtendedFollowUp###ExternalReference*).

The recognised number is used to look for a matching ticket in the "ExternalReferenceNumber" dynamic field. It is also noted that the recognised external reference should be stored in the "ExternalReferenceNumber" dynamic field, regardless of whether a new ticket is created or another mechanism recognises a follow-up (→ *ExtendedFollowUp###DynamicFieldMapping*).

If another email comes from a suitable sender (e.g. from "info@example.de" or "'support@example.de") with the subject "[SMPL::#12345] Request", KIX recognises the number again and the previously created ticket is found during the search. In this case, the *PostMaster::PreFilterModule###888-ExtendedFollowUp* filter adds the KIX ticket number to the subject.

This allows the "*PostMaster::CheckFollowUpModule###0100-Subject*" module to recognise the follow-up and assign the email to the existing ticket.

Configuration key	Description	Example

ExtendedFollowUp ###Identifier	Registers (initialises) an identifier for use in the SysConfig keys listed below.
	
	The filter configurations are processed according to the keys specified here.
	
	Specify a key-value pair:
	
	• Key: Used for alphanumeric sorting of the filter configurations to be applied.
	• Value: Is used as the key for the assignment in the configuration keys listed below.
		``{
		    "10": "Sender_A",
		    "20": "Sender_B",
		    "4711":
		"capeProcessCPNID"
		}```

ExtendedFollowUp ###SenderEmail	Checks the sender of the incoming e-mail for the specified e-mail domains.
	
	Specify a key-value pair:
	
	• Key: Must be one of the identifiers initialised in *ExtendedFollowUp###Identifier*.
	• Value: Email address or regular expression as sender, which must match the "From" of the imported email.
		``{
		    "capeProcessCPNID": ".
		+@cape-it.de|.+@kixesk.com|.
		+@any-example.com",
		    "Sender_A": ".
		+@example.de",
		    "Sender_B": ".+@cape-
		it.de"
		}```

Configuration key	Description	Example

ExtendedFollowUp ###ExternalReference	Checks the e-mail subject for the specified regular expressions. Recognises whether one of the specified references is specified in the subject.
	
	Specify a key-value pair:
	
	• Key: Must be one of the identifiers initialised in *ExtendedFollowUp###Identifier*.
	• Value: Must be a regular expression that captures the external reference in the subject. The first CaptureGroup of the expression is used as the external reference.
		``{
		    "CapeProcessCPNID": "##([0-9]{6})##",
		    "Sender_A": "\\[SMPL::#(.)\\]",
		    "Sender_B": "\\[T#(.)]"
		}```

ExtendedFollowUp ###DynamicField Mapping	Checks for an already noted reference in the specified dynamic fields and writes the reference found in this field.
	
	Specify a key-value pair:
	
	• Key: Must be one of the identifiers initialised in *ExtendedFollowUp###Identifier*.
	• Value: The name of the dynamic field.
		``{
		    "CapeProcessCPNID": "CPNID",
		    "Sender_A": "ExternalReferenceNumber",
		    "Sender_B": "CustomerReferenceNumber"
		}```

ExtendedFollowUp ###SortByAgeOrder	General configuration. Sorts the tickets found by age and controls whether the assignment is in ascending (Up) or descending (Down) order. The external reference can be stored on several tickets. By selecting the sorting, you can determine whether the follow-up is assigned to the oldest (Down) or youngest (Up) ticket.	`Up | Down`

Configuration key	Description	Example
ExtendedFollowUp ###AllTicketState TypesIncluded	General configuration. Specify which types of follow-up tickets should be taken into account. • First viewable tickets (0): Attempts the assignment first to tickets that correspond to the status types specified in the SysConfig key *Ticket::ViewableStateType*. If no such ticket is found, other status types are also taken into account (e.g. "closed"). • All tickets (1): Considers all tickets without restriction of the status type.	`0 | 1`

ⓘ **Info**

If "SenderEmail", "ExternalReference" or "DynamicFieldMapping" is not specified for an identifier, no processing takes place.

⚠ **Please note!**

The filter does not overwrite any references already stored in the ticket. Based on the external reference, the subject is extended by the ticket number including the hook of the ticket with a matching reference.
The SysConfig key *PostMaster::CheckFollowUpModule###0100-Subject* must therefore be active (valid).

Processing of follow-ups

Follow-ups are processed in the order in which they were detected.

1. Check whether the follow-up ticket already contains an article with the message ID of the email just read in.
    1. Yes: No further processing (no article, no ticket, no header evaluation). However, the postmaster evaluates it as: "Processing as follow-up".
    2. No: Continue with 2.
2. Check if follow-up ticket is closed

1. If the ticket is not closed, or if the relevant team has the option "Follow Up on Tickets" set to "possible", then:
    a. a follow-up item is created
    b. the follow-up header is processed
    c. automatic mechanisms for ticket lock and ticket status take effect.
2. If the option "Follow Up on Tickets" is set to "reject" at the relevant team, then:
    a. the article is deposited
    b. the follow-up headers are not processed
    c. no automatisms will take effect.
3. If the option "Follow Up on Tickets" is set to "new ticket" on the relevant team, then:
    a. the email is distributed as a new ticket at this point and then linked to the follow-up ticket.

Fig.: Scheme of processing of follow-ups

Email distribution

Menü	KIX > Communication > Email > Inbox

When creating or editing an email account, you can select the type of distribution. The type of distribution determines how KIX should proceed with the emails retrieved from the account if

- no follow-up was found
- a follow-up goes to an already closed ticket and the option "new ticket" is set in its team configuration under "Follow Up on Tickets".

With your selection, you determine in which team the emails are created as **new** tickets. However, if the message ID exists in a ticket in a team, no new ticket will be created.

You can define the distribution separately for each email account.

Fig.: Selecting the distribution of incoming emails

> ⓘ **Infos**
>
> - Configured email filters override the default behaviour. They have priority over the default.
> - If the email header 'X-KIX-Queue' is set with a valid team, the ticket is created in the specified team and thus independent of the distribution configuration.

Default Queue (SysConfig)

The emails retrieved from the email account are distributed to the default team. The default team is the team that is stored in the SysConfig key "PostmasterDefaultQueue". In the delivery state, this is the "Service Desk" team. You can define a different team in the SysConfig key.

Team

A new ticket is created in the team that is stored in the mail account.

Recipient addresses (To, CC, etc.)

The emails retrieved from the email account are distributed according to the email recipient. KIX compares the recipient list of the retrieved emails with the email addresses (system addresses) created in the system (see "Evaluation of the mail headers").

If one of the recipient addresses exists as a system address in KIX and a team is stored at this address, the incoming mail will be created as a ticket in this team. If no team is stored at any relevant system address, the new ticket will be created in the postmaster default team (default queue).

Fig.: Scheme of mail distribution according to recipient addresses

The system addresses are created and managed in the menu "*Communication > Email > Email addresses*". You can specify a team at each system address. In this team, all incoming emails to the system address are created as new tickets if "Recipient addresses" is selected as the distribution option in the email account.

> ⚠ **Note**
>
> The system does **not** ensure that the assignment of system address and team matches the system address stored on the team.

> (i) **Notes**
>
> If several system addresses exist which refer to the same address, the entry with the lowest ID is always evaluated for the team assignment.
> ⚠️ We advise against using several system addresses with the same email address.

Evaluation of the mail headers

KIX compares the recipient list of the retrieved emails with the email addresses created in the system (system addresses). The mail headers listed below are checked and the recipients evaluated.

- The first header recipient for which a system address with an assigned valid team is found determines the team in which the ticket is created.
- If no valid team was found for the recipient, the next header is checked in the same way.
- If no team was found after all checks, the ticket will be created in the postmaster default team (default queue).
- If several email addresses are specified as recipients, the check is carried out for each of these email addresses in turn.

The mail headers are checked in the specified order:

1. Resent-To
2. Envelope-To
3. To
4. Cc
5. Delivered-To
6. X-Original-To

Example

The system addresses "projects@kixdesk.com" (without team assignment) and "sales@kixdesk.com" (with team "Sales") are configured in the system. Postmaster default team is "Projects".

Email to (To)	Copy to (Cc):	New ticket in team	Description
no-systemaddress@domain.com	-	Projects	No system address available, therefore new ticket in the standard team "Projects".
projects@kixdesk.com	-	Projects	No team assignment at system address, therefore new ticket in the standard team "Projects".

Email to (To)	Copy to (Cc):	New ticket in team	Description
sales@kixdesk.com	-	Sales	Team assignment exists at system address, therefore new ticket in team "Sales".
no-systemaddress@domain.com	projects@kixdesk.com	Projects	No system address available for 'To' and no team stored for 'Cc', therefore new ticket in the standard team "Projects".
sales@kixdesk.com	projects@kixdesk.com	Sales	Team assignment exists at system address ('To') and team is stored at this address, therefore new ticket in team "Sales"- The first hit wins; the check of the 'Cc' is therefore omitted.
projects@kixdesk.com	sales@kixdesk.com	Sales	No team assignment available at system address ('To'), therefore check next address ('Cc'). Team assignment at 'Cc' available (= first hit), therefore new ticket in team "Sales".

9.1.2 Outbox

In order for KIX to be able to send emails, you need to set up an SMTP mail server for outgoing mail. The setup is done either in the setup assistant or in the *Communication > Email > Outbox* menu. You can change the mail server at any time, for example after changing the hosting service provider.

You need the system rights READ and UPDATE on / system / config to set up the outgoing mail server.

Fig .: Example outgoing mail via Gmail

Have the following information ready:

Credentials for the outgoing mail server:

- Host of the SMTP server
- the transmission protocol used including the port used
- User data and password for the SMTP server

If you use the mail server of a service provider (Gmail, IONOS, Strato etc.), you will receive the necessary login information from them. If you use your own mail server, you should know the login information.

How to configure the SMTP mail server for outgoing mail:

1. Navigate to the *Communication > Email > Outbox* menu. A dialog opens in which you can configure the outgoing mail server.
2. Enter the server information in the corresponding input fields (see table below).
3. Finally save your entries.

The outgoing mail server is now configured so that KIX can use it to send emails.

The form dialog contains, among other things. the following input fields:

Id	Description
Type	Transmission protocol for sending emails (SMTP). With the selected protocol you determine the transmission method. If necessary, you can obtain detailed information from your web host. Supported protocols: - without encryption: SMTP - with SSL encryption: SMTPS - with TLS encryption: SMTPTLS **Recommendation**: First test SMTP <u>without</u> encryption. **Note:** The "DoNotSendEmail" option does not send emails, it can be used for test systems.
Host	Address to the SMTP interface of the outgoing mail server, e.g. "smtp.mailbox.org"
Port	Port via which the emails with the previously selected "Type" are sent. The port depends on the selected transmission protocol: - SMTP: 25 - SMTPS: 465 - SMTPTLS: 587
User	Username for logging into the outgoing mail server. This often corresponds to the name of the email account. For example: mustermann@example.de This value can also be cleared via port 25.
Password	The password of the email account. This value can also be cleared via port 25.

Id	Description
Envelope	You can put your outgoing emails in an "envelope" (optional). Only what is written on the "envelope" can then be recognized by the mail transport.
	Envelope from: You can specify a sender who should appear on the "envelope", e.g. senders@example.de or mbox / RFC 5321. If no sender is specified, the team's email address is used as the sender.
	Notification Envelope from: You can enter an email address that will be used as the sender header in outgoing notifications. If no address is given, the header of the "envelope" is empty.
	Notification Envelope from Fallback: If "NotificationEnvelopeFrom" is not specified, this setting allows you to use the sender address instead of a blank envelope sender (required in certain mail server configurations).

9.1.2.1   Configuring the Email Dispatch (SysConfig)

Configuration key:	*sendmail*

You can optionally configure the outgoing mail server using the configuration key in the SysConfig menu. This option is useful if you want to configure server properties that go beyond the initial setup, or if you have your own mail server and want to directly influence the configuration.

To change a configuration key:

1. Navigate to the *System> SysConfig menu*.
2. Look for the key "* sendmail *". You will be shown several keys for the mail server configuration.
    Grayed out keys are "invalid" and can be set to "valid" if necessary.
    Keys with a lock icon are blocked for editing.
    Keys marked in orange are security-relevant keys set to "valid".
3. Click on the key to be configured. A form dialog opens in which you can edit the value of the key. The specific settings depend on the mail server used.
4. Change the value of the key as necessary. If you are using a JSON editor, make sure that all spaces and line breaks are removed before saving.
5. Save your changes with "Save". The key has now been changed.

The keys listed below are required to configure the sending of emails. Additional Sendmail keys can optionally be used.

Configuration key	Description	Example
SendmailModule	For specifying which transfer protocol should be used (e.g. SMTP | SMTPS | TLS).	Kernel::System::Email::SMTPS
SendmailModule::AuthPassword	For saving the password of the email server.	My1stMa!lservErPassW0rd
SendmailModule::AuthUser	For saving the username for logging in to the email server. This is often the name of the email account.	email@example.com
SendmailModule::Host	Enter the address of the email server.	smtp.hoster.com

Configuration key	Description	Example
SendmailModule::Port	Enter the port of the email server. The port depends on the selected transfer protocol.	The following ports are commonly used: For SMTP: 465 For TLS: 587

Optionally, you can put your outgoing emails in an "envelope". Only what is written on the "envelope" can then be recognized by the mail transport. You can use the following keys for an "envelope":

Configuration key	Description
SendmailEnvelopeFrom	You can specify a sender who should appear on the "envelope", e.g. senders@example.de or mbox / RFC 5321. If no sender is specified, the team's email address is used as the sender.
SendmailNotificationEnvelopeFrom	You can enter an email address that will be used as the sender header in outgoing notifications. If no address is given, the header of the "envelope" is empty.
SendmailNotificationEnvelopeFromFallback ToEmailFrom	If "NotificationEnvelopeFrom" is not specified, this setting allows you to use the sender address instead of a blank envelope sender (required in certain mail server configurations).

9.1.2.2   Set default addresses

KIX requires at least one valid email address as the default address for outgoing mail. All emails are sent via this address if no other email addresses have been created or if sending from an email address is not possible, e.g. because it is set to "invalid".

The initial default email address for the outbox is kix@localhost. Replace this with your own email address! This can be done directly after the installation in the setup assistant (see page 58) (step "Outbox"):

Abb.: Changing standard system address in setup assistent

Or navigate to the menu *Communication > Email > Email Addresses.* There, click on the address "kix@localhost" and then on "Edit". In the dialog that opens, replace the address specified in the "Email address" field with your own default address. Note that the email address must match an email account known in KIX.

Abb.: Changing standard address in the Email > Addresses menu

Use a placeholder for the domain

You can use a placeholder (see page 750) for the domain name in the default address (<KIX_CONFIG_FQDN>), then the domain is automatically pulled by the system. Precondition: the complete domain name to the frontend must be specified in the SysConfig key "FQDN". The FQDN can already be set in the Setup Assistant under "System".

* Name: ⓘ	FQDN	
Description: ⓘ	Defines the fully qualified domain name of the system. This setting is used as a variable, KIX_CONFIG_FQDN which is found in all forms of messaging used by the application, to build links to the tickets within your system.	
* Validity: ⓘ	valid	▾
Value: ⓘ	hostname.example.com	
Default Value: ⓘ	yourhost.example.com	

Fig.: Changing domain name

9.1.2.3 Error handling for outgoing emails

Below you will find possible solutions to known problems when sending emails:

Problem	Cause	Solution
The sent email contains the FQDN of the KIX18 system in the MessageID. (By default this is "yourhost.example.com").	Outgoing emails from Applicance Hornet security were blocked: *"Reporting-MTA: dns; mout-b-203.mailbox.org X-Postfix-Queue-ID: 4CqzHG1CwVzQljX X-Postfix-Sender: rfc822; Arrival-Date: Tue, 8 Dec 2020 12:59:06 +0100 (CET) Final-Recipient: rfc822; Original-Recipient: rfc822; Action: failed Status: 5.6.1 Remote-MTA: dns; mx14a.antispameurope.com Diagnostic-Code: smtp; 554 5.6.1 Your mailheader contains SPAM. To unblock visit http://antispameurope.com/unblock?a=0812&b=&d=1&e=195.10.208.52-mx-gate64-hz1"*	After adjusting the FQDN, the emails are sent without any problems (SysConfig key: "FQDN")
1. Ticket notifications do not go out. 2. In the item list, a red triangle is displayed in the "Channel" column.	The log contains the following error message: *"[Error] [Kernel::System::Email::SMTP::Send] [152] Can't send to 'empfaenger@company.de': 5535.7.1 <absender@company.de>: Sender address rejected: not owned by user company@kix.cloud"*	Check and adjust sender email in the team settings (e.g.: Team "Service Desk"): • Incorrect: "sender@company.de" (no processing possible). • Correct: "company@kix.cloud"

Problem	Cause	Solution
Ticket notifications do not go out	1. The log contains the following error message: *"[Error] [Kernel::System::Email::SMTP::Send][173] Can't send message: 5545.2.0 STOREDRV.Submission.Exception: SendAsDeniedException.MapiExce ptionSendAsDenied; Failed to process message due to a permanent exception with message Cannot submit message."* 2. Email server rejects notifications.	Customization of the sender address for notifications required: SysConfig-Key: *"NotificationSenderEmail"*

9.1.3 Email Addresses

Menu	KIX > Comunication > Email > Email Addresses

You can create and manage system addresses for email communication. These addresses are the sender addresses for the emails sent from the respective teams. They thus identify the sender of an email. In practice, the email addresses are often named after the associated team. For example, "support@example.de" for all emails from the "Support" team.

Each team must be assigned an email address that exists in the system so that it can send emails via this address. Only one email address can be assigned to a team, but one email address can be assigned to different teams. The assignment is done when creating or editing a team in the menu "*Ticket > Teams*". It is therefore recommended to create the email addresses before creating the teams.

Furthermore, these addresses serve the internal distribution of incoming emails to the respective teams. If a team is stored at an email address, incoming emails can be forwarded to this team. This depends on the distribution set in the inbox (see page 201) and the configured email filters (see page 231) .

Each email address must match an email account configured in KIX or be retrieved from a collective mailbox if replies to this address are to arrive at the KIX tickets.

> ⓘ **Note**
>
> KIX requires at least one valid email address to send, but can work with multiple addresses. KIX is delivered with a default address (kix@localhost). We recommend that you replace this with your own default address during the initial setup (Setup Assistant menu). Alternatively, you can set the initial default address to "invalid" and create your own default address (*Communication > Email > Email Addresses menu*).

9.1.3.1 Creating or Editing an Email Address

Create the required system addresses under "*Communication >Email > Email Addresses"* and then assign them to the teams under "*Ticket > Teams*".

Fig.: Editing an Email Address for Support Team

To create a new email address, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Addresses*. In the content area, a table listing all email addresses saved in the system opens.
2. In table, click "New Address". A form dialog for creating the email address opens.
3. Complete form, set validity to "valid", and click "Save" to save your entries. The email address has now been created.
4. You can now navigate to *Ticket > Teams* and assign the email address to a team so that the team can use this address as the sender address for emails.

To edit an email address, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Addresses*. In the content area, a table listing all email addresses saved in the system opens.
2. In table, click email address to be edited. The zoom view opens.
3. Click "Edit" in title bar of opened zoom view. A form dialog for editing the email address opens.
4. Amend information as required and set validity to "valid". The changes take effect immediately.

To delete an email address, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Addresses*. In the content area, a table listing all email addresses saved in the system opens.
2. Place a tick in front of the email address to be deleted. You can select several addresses as well.
3. Click on "Delete" in the table header. The button is only active if at least one email address is selected.
4. Answer the security question with:
    1. "Yes": The email address is deleted and is no longer available.
    2. "No": The email address is not deleted and remains ticked.

ⓘ **Notes on deleting email addresses:**

- An email address can only be deleted if it is not assigned to a team. If this is the case, you will receive an error message. Assign a new email address to the teams concerned beforehand.
- Email addresses that have already been used remain in the ticket as text after deletion (e.g. "From" or "To" fields in the article).
- The delete button is only active if at least one account is selected.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Email Address	Enter valid email address. This is the sender address for a team, e.g.: support@YourCompany.com
Display Name	This is the name/identifier for the sender, e.g. "ExampleCompany – Support"
Team	Optionally select a team to which <u>incoming</u> emails and replies to this email address are forwarded. The prerequisite is that "Recipient addresses" is set as the distribution in the inbox account. ℹ On the system side, it is not ensured that this assignment of system address and team matches the system address stored at the team. Both settings are independent of each other. This setting concerns the incoming mail, the email address stored for the team concerns the outgoing mail.
Validity	• **valid:** The address is active. This address can be used to send emails. • **invalid/invalid-temporarily:** In order to send an email, KIX requires at least 1 sender address. Please therefore ensure that you have set at least the default address saved in the SysConfig to "valid".

9.1.4 Email Filters

Menu	KIX > Communication > Email > Email Filters

You can set up email filters (so-called postmaster filters) to distribute incoming emails granularly and specifically in the system on the basis of predefined filter conditions. The distribution of incoming mail via email filters is independent of the set standard and thus has priority over the standard distribution.

Email filters manipulate the email header and proceed according to the "if-then" scheme: *If* an incoming email has certain properties (filter conditions), *then* the attributes defined in the filter are written into the email header. Based on these attributes, KIX can assign ticket properties to the incoming mails and create them as tickets in the respective team folder.

This way, you can e.g. define that the emails of a certain customer are always given the highest priority and then created as a ticket in a certain team folder. You can also create a so-called blacklist, for example, by specifying that emails from certain senders are moved to the junk folder. Other use cases include

- ignoring automatic replies such as out-of-office notifications
- forwarding unsolicited emails to a junk folder
- setting ticket parameters in incoming mail (e.g. priority high, status new, etc.).

email filters can also be useful if you work with several KIX systems. Then you can, for example, give the tickets parameters that are evaluated and applied in the receiving KIX system. The receiving KIX system must accept the KIX header. (Checkbox "Accept KIX header" in the menu "*Communication > Email > Inbox*"). However, please use this option **with caution**, as otherwise system errors may occur.

A set up, valid filter looks at all incoming emails in the system, regardless of which email account they come in from. In doing so, it evaluates the specified filter conditions and writes the attributes and values linked to the condition back into the header. This happens at the moment the email is retrieved. Only then does KIX convert the email into a ticket, performs the actions specified in the email filters and then creates the ticket in the assigned team folder.

> ⓘ **Info**
>
> The application of the email filters is based on their order in KIX. Each email passes through the email filters in the same order as they are listed in the overview of the email filters. The overview lists the email filters in alphabetical order by name. You can determine the order yourself if you name the filter with preceding numbers, e.g. "01_Spamfilter".

9.1.4.1 Zoom view of an email filter

You are able to view the details of an email filter. Navigate to *Communication > Email > Email Filter*. In the content area, a table listing all email filters saved in the system opens. In table, click email filter to open associated zoom view.

The buttons in the title bar of the zoom view allow you to edit, create and duplicate email filters.

Fig.: Zoom View of an Email Filter

Alongside the lane containing the general email filter information, the zoom view also contains other lanes:

Lane	Description
Filter Conditions	List of filter conditions for setting the email header
Set Email Headers	List of email headers that will be set.

9.1.4.2   Configuring an Email Filter

Menu	KIX > Communication > Email > Email Filters

Once a valid filter has been set up, it will take into account all inbound emails coming into the system, regardless of the account they came from. This means that emails can be precisely assigned to specific teams. Inbound emails that match the criteria defined in the email filter will be created as tickets with the set attributes and moved into the relevant team folders.

Sometimes the configurations of the individual email filters differ only slightly. You can therefore duplicate existing email filters and configure the duplicate differently.

Fig.: Configuring an Email Filter

To set up a new email filter, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Filter*. In the content area, a table listing all email filters created in the system opens.
2. In table, click "New Filter". A form dialog for saving the filter criteria opens (also see table below).
3. As the last step, click "Save" to save your entries.

The new filter is now active, and inbound emails can be filtered according to the criteria specified in the filter.

To edit an email filter, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Filter*. In the content area, a table listing all email filters created in the system opens.

2. In table, click email filter to be edited. A form dialog for changing the information opens.
3. As the last step, click "Save" to save your entries.

To duplicate an email filter, proceed as follows:

1. In the explorer, navigate to *Communication > Email > Email Filters.* In the content area, a table listing all email filters created in the system opens.
2. Place a tick in front of the email filter to be duplicated. You can only duplicate 1 filter at a time.
3. Click on "Duplicate" in the table header. The button is only active if at least 1 email filter is selected.
4. Configure the email filter in the form dialogue that opens. The form fields contain the values of the source and can be changed.
   The name of the duplicate is prefixed with "Copy of". You can give the duplicate a new name.
5. As the last step, click "Save" to save your entries.

To delete one or more email filters, proceed as follows:

1. In explorer, navigate to *Communication > Email > Email Filter*. In the content area, a table listing all email filters created in the system opens.
2. By checking off relevant option, select every email filter you wish to delete. You can select one or more filters.
3. In table, click "Delete". The button is only active if at least 1 email filter is selected.
4. To definitively delete filter(s), click "Yes" in prompt. To cancel deletion process, click "No" in prompt.

The form dialog contains the following input fields, among others:

Field (selection)	Description	
Name	Please note that the email filters will be processed sequentially, from top to bottom.	
	They will be processed in the sequence stated in the overview of email filters.	
	The overview lists the created filters by name. If you would not like the email filters to be processed in alphabetical order, you can prefix the name with a number to define the sequence yourself.	
Stop after match	Check off this option if you would like to stop the processing of the email filters after finding a match. Otherwise, all subsequent email filters will be processed until the end or until the next "Stop after match".	
Validity	valid:	The email filter will be applied.
	invalid/ invalid-temporarily	The email filter will (temporarily) not be applied and the conditions it has defined will not be taken into account. Invalid filters will be skipped when the email filters are processed.

Field (selection)	Description
Filter Conditions	KIX will check the email header for the conditions specified here. If it finds a match, the parameters specified under "Set Email Headers" will be written back to the email header. You can check the header for several conditions simultaneously. A logical AND operator is created when this happens. The specified parameters are only written to the email header if all of the conditions have been met. *Fig.: The condition applies if the email comes from the address info@example.com*

	Email Header	Specify the attributes for which you want to check the email header. For example, the sender of an email (From) or the subject. All attributes defined in the SysConfig key "*PostmasterX header*" are available for selection. If necessary, you can add further individual attributes to the key. **Note**: To use mail headers for dynamic fields, the available DF names must be entered in the SysConfig key "*PostmasterX-Header*" (format: `X-KIX-DynamicFieldName`).
	Negate	Check off this option if you would like to check in the email header that certain values are not present. The condition is considered met if the condition specified under "Pattern" is NOT included.
	Pattern	Enter the values/pattern that the selected header attribute should match in order to trigger the filter. The use of regular expressions (see page 744) is possible, e.g. ".+@example.de[11]" for all emails to the domain "example.de". Using parentheses, a so-called "Capture Group" can be created to determine the domain part of an email address, e.g. ".+@(.+)". Under "Set Email Header", the determined pattern (in the example the domain part) can then be set using "[***]".

---

11 http://example.de

Field (selection)	Description
Set Email Headers	If all filter conditions apply, KIX will write the parameters specified here (attributes and values) to the email header. You can write several parameters to the email header (1 attribute per line). When converting the email into a ticket, these parameters will be read out and the actions they define will be executed. ![Set Email Headers form showing Set Email Header: X-KIX-Queue and Set Value: Chemnitz] *Fig.: If the selected condition applies, the ticket will be moved in team "Chemnitz".*

	Set Email Header	Select which attributes are to be placed in the email header (one attribute per line). All attributes defined in the SysConfig key "*PostmasterX header*" are available for selection. If necessary, you can add further individual attributes to the key. **Note**: To use mail headers for dynamic fields, the available DF names must be entered in the SysConfig key "*PostmasterX-Header*" (format: `X-KIX-DynamicFieldName`).
	Set Value	Enter the value that the attribute selected under "Set Email Header" should assume. By means of "[***]" a value previously determined in the filter conditions by means of regex capture group can be set (see above).

9.1.4.3 Overview of Usable KIX Email Headers

Below you will find a selection of the available KIX e-mail headers. The attributes listed are stored in the SysConfig key "PostmasterX header" and can be set in the mail header via the e-mail filter. If required, you can add further, individual attributes in the SysConfig key.

Attributes (selection)	Description	Possible values
X-KIX-BodyDecrypted	Decrypted content of the mail. Only serves as a filter; cannot be set!	
X-KIX-Channel	Sets the channel of the article to be created from the inbound message	mail | note
X-KIX-SenderType	Sets the sender type of the article to be created from the inbound message.	agent | external | system
Headers for new tickets		
X-KIX-Contact	Sets the contact of the new ticket, defined by the corresponding unique com email address.	john.doe@company.com
X-KIX-Ignore	Discards the mail if the value "yes" is set for the header. No ticket/article is created.	yes | no
X-KIX-Lock	Sets the ticket lock of the new ticket directly.	lock | unlock
X-KIX-Organisation	Assigns the new ticket to the organization as identified by the customer number.	MY_ORGA
X-KIX-Owner	Sets the owner of the new ticket by specifying their login name.	loginname
X-KIX-OwnerID	Sets the owner of the new ticket by specifying their numerical user ID.	123

Attributes (selection)	Description	Possible values
X-KIX-Priority	Sets the priority of the new ticket by specifying the internal (i.e. nonlocalized) priority designation.	1 very high | 2 high | 3 normal | 4 low | 5 very low
X-KIX-Queue	Assigns the new ticket to be created to the specified team (queue). The complete name, including any superordinate teams, must be entered. Names of superordinate teams are separated by a colon. If the header 'X-KIX-Queue' is set with a valid team, all tickets will be created in the specified team. This is independent of the distribution configuration specified in the email account.	Service Desk | Junk | Service Desk::Monitoring
X-KIX-Responsible	Sets the person responsible of the new ticket by specifying their login name.	loginname
X-KIX-ResponsibleID	Sets the person responsible of the new ticket by specifying their numerical user ID.	123
X-KIX-SLA	Sets the SLA of the new ticket to be created, specifying the SLA name (KIX Pro only)	Name of the SLA, e.g. "No Escalation"
X-KIX-State	Sets the state of the new ticket by specifying the internal (i.e. non-reminder|closed localized) state designation.	new | open | closed | pending reminder | closed
X-KIX-State-PendingTime	The wait time of the associated ticket in seconds, or in the format "+<INTEGER>s", "+<INTEGER>m" (whole minutes), "+<INTEGER>h" (whole hours), "+<INTEGER>d"n (whole days). It is not permitted to combine values from different time units.	86400 | +86400s | +1440m | +24h | +1d

Attributes (selection)	Description	Possible values
X-KIX-StrictFollowUpIgnore	If the SysConfig key "*PostMaster::StrictFollowUp*" is activated, the relevant mail can be handled as if the SysConfig were deactivated by setting this header.	0 | 1
X-KIX-Subject	Sets the title/subject of the new ticket and first article to be created. With FollowUp, the header only affects the subject of the article. The ticket title remains unchanged.	some text here
X-KIX-Type	Sets the type of the new ticket by specifying the internal (i.e. non-localized) ticket type designation.	Incident | Service Request | Unclassified
Headers for setting dynamic fields on new tickets (generic)		
Please note: • For array configuration of a Dynamic Field, only the first array entry is set. • To use mail headers for dynamic fields, the names of the available dynamic fields must be entered in the SysConfig key "*PostmasterX-Header*" (format: `X-KIX-DynamicField_DynamicFieldNam` e). • The following examples use the notation based on the placeholder syntax with underscore "_". However, the notation with hyphen "-" is also accepted, e.g. `X-KIX-DynamicField-DynamicFieldName.`		
X-KIX-DynamicField_SomeTextField	Sets the value of the field "SomeTextField" some text here of the type "Text" or "TextArea" on the new ticket.	some text here
X-KIX-DynamicField_SomeDateTimeField	Sets the value of the field "SomeDateTimeField" of the type "DateTime" on the new ticket. In the case of DateTime fields, the format "YYYY-MM-DD hh:mm:ss" must be used.	2020-12-31 23:59:59

Attributes (selection)	Description	Possible values
X-KIX-DynamicField_SomeDateField	Sets the value of the field "SomeDateField" on the new ticket. In the case of DateTime fields, the format "YYYY-MM-DD 00:00:00" must be used. For additional form fields, "SomeDateField" must be replaced with the relevant field name.	2020-12-31 00:00:00
X-KIX-DynamicField_SomeSelectionField	Sets the value of the field "SomeSelectionField" on the new ticket. In the case of single/multiple selection fields, the key of the desired value must be set, e.g. "1" for "solved (work around)" in the field "CloseCode".	ItemKeyHere
Setting ticket properties for existing tickets (generic)		
X-KIX-FollowUp-XXXX	Sets the ticket attribute identified with XXXX if there is an inbound message for an existing ticket ("Follow Up"). It is used in the same way as when setting the relevant attribute when creating the ticket. Examples are: X-KIX-FollowUp-Lock, X-KIXFollowUp-State, X-KIX-FollowUp-State-PendingTime, and X-KIX-FollowUp-DynamicField_SomeTextField	See information above
X-KIX_FollowUp-KeepState	If this header is set, a follow up **does not lead** to a status change of the ticket and there is no change of the pending time, e.g. after an automatic reply or an out of office message.	Value: 1

9.1.5 OAuth2 Profiles

OAuth (Open Authorisation) refers to two different open protocols for a standardised, secure API authorisation of desktop, mobile and web applications.

In order for KIX to be able to retrieve emails from or send emails via an Office365 mail account, for example, OAuth2 authentication must be applied. In addition, SMTP with authentication may first have to be activated on the account at Office365. To provide authentication in KIX, proceed as described below:

Content on this page:

- Step 1: Microsoft Azure Portal (see page 242)
- Step 2: Create OAuth2 profile (see page 245)
- Step 3: Create email account (see page 247)
- Step 4: Configure email sending (see page 247)

> ⓘ **Official notice pages from Microsoft**
>
> Inbox (IMAP/S, POP3/S): https://docs.microsoft.com/en-us/exchange/clients-and-mobile-in-exchange-online/pop3-and-imap4/enable-or-disable-pop3-or-imap4-access
> Outbox (SMTP): https://docs.microsoft.com/en-us/exchange/clients-and-mobile-in-exchange-online/authenticated-client-smtp-submission

9.1.5.1 Step 1: Microsoft Azure Portal

1. Sign in to the Azure portal.
2. Go to "Azure Active Directory".
3. Go to "App Registrations" in the left action list.
   - Select the item "New registration" in the menu bar.
   - Enter any name for the registry. This is not relevant for KIX later on.
   - In the item "Supported account types", specify who can use the app. Set the selection to "Only accounts in this organisation directory (only "<company>" - single client)".
     ⓘ Instead of "<company>", you should use your own company name.
   - For "Redirection URI", select "Web" in the selection field and enter "<HttpType>://<FQDN>/oauth2redirect" as the URI. This can also be entered or changed at a later time.
     The redirect URI corresponds to the front-end FQDN in KIX, e.g. "https://myKIXFrontendHost.example.com/oauth2redirect".

     > ⓘ **Info**
     >
     > Microsoft requires that the redirect URI points to HTTPS.

4. Go to "Certificates & Secrets" in the left action bar.

- Select the item "New client secret" and have the key created in the right sidebar. Remember or make a note of the key (value), because it will only be partially displayed when called up again and must otherwise be stored again.

> ⓘ **Info**
>
> It is the administrator's task to ensure that a valid client key (valid for max. 2 years) is stored in KIX for the mailbox.

5. For the inbox:
   - Go to "API permissions" in the left action bar.
   - Select the item "Add permissions".
   - Then select "Microsoft Graph" in the right sidebar.
   - Select the item "Delegated permissions".
   - At this point, select "IMAP.AccessAsUser.All" and "offline_access" and confirm the selection. This is now displayed in a table, whereby the column "Status" remains empty (default value).
   - Give the administrator's consent for the operation (field "Administrator consent for (your company)" to the right of field "Add authorisation"). The column "Status" in the table with the selected permissions now shows a green tick and the addition "granted for 'your company' ".
   - Optionally, perform these steps again if you want to add permissions, e.g. for outgoing mail.

6. For outbox:

- Go to "API permissions" in the left action bar.
- Then click on "Add permissions".
- Then select "Microsoft Graph".
- Select the item "Delegated permissions".
- At this point, select "SMTP.Send", as well as "offline_access" and confirm the selection. This is now displayed in a table, whereby the column "Status" remains empty (default value).
- Give the administrator's consent for the process ("Administrator consent for (your company)" field to the right of the "Add authorisation" field). The column "Status" in the table with the selected permissions now shows a green tick and the addition "granted for 'your company' ".
- Optionally, perform these steps again if you want to add permissions, e.g. for the inbox.

7. Select "Overview" in the left action bar.

- Make a note of the "Application ID (Client)" and the "Directory ID (Client)". You can see the data in the summary.You must enter this ID in the "Client ID" field when you create an OAuht2 profile in KIX.

9.1.5.2 Step 2: Create OAuth2 profile

1. Navigate to *Communication > Email > OAuth2 Profiles*.
    In the overview that opens, you can create or delete profiles or perform a table CSV export. The table contains all profiles created so far.

2. Click on "New Profile" to create a new profile. A form dialogue opens in which you can enter the parameters of the profile.

3. Fill in all the form fields:

Field	Description
Name	Enter a name for the profile.
URL Authorization	Enter the URL for the OAuth2 authorization. Example for the structure at Office365: https://login.microsoftonline.com/**<Tenant-ID/Directory-ID>**/ oauth2/v2.0/authorize
URL Token	Enter the URL for the OAuth2 Token. Example for the structure at Office365: https://login.microsoftonline.com/**<Tenant-ID/Directory-ID>**/ oauth2/v2.0/token
URL Redirect	This field is pre-filled by the system. If the address is not the same as the one entered in the Azure portal (step 1 - item 3 - subitem 5), it must be changed accordingly. The forwarding URL is used as the "return address" for the authorisation codes from Microsoft. It must match the one in Azure under "Redirection URI". Example: https://<myFrontendHost.example.com>/oauth2redirect <myKIXFrontendHost.example.com> corresponds to the frontend FQDN that is stored in the Setup Assistant or in the SysConfig key "FQDN".
Client ID	Enter the Client ID here. For Office365 this is the 'Application ID (Client)' from the 'Overview'.

Field	Description
Client Secret	Enter the client secret here. For Office365 this is the 'client key' from 'Certificates & secrets'.
Scope	Enter the permissions to be requested here (separated by spaces). Examples for Office365: • "offline_access https://outlook.office.com/SMTP.Send" • "offline_access https://outlook.office.com/IMAP.AccessAsUser.All"
Validity	Set the form to 'valid' to use it.

> ⚠ **Note**
>
> When creating and later authenticating the profile, the user must be created who is to be used for the mailbox or mail dispatch.
> For example, if you want to use a mailbox via 'mail@test.onmicrosoft.com', authentication must be carried out via this user. A different admin user (e.g. 'admin@test.onmicrosoft.com') has no authorisation to do so.

4. Finally click on "Save" to save the form.

> ⚠ **Important!**

The verification must be carried out with the same user (email account) who collects the emails. Therefore, use a private browser window when setting up to ensure that the generated tokens match the mailbox to be retrieved!
If you receive the error message "Error: Could not select: 3 BAD User is authenticated but not connected", this indicates that you are already logged in with your Admin Microsoft account. In this case, cached session cookies or similar ensure that the token for the mailbox that KIX is supposed to retrieve is not valid.

5. In the overview of profiles in the column "Renew authorisation", click on [icon] . After submission, an automatic forwarding to Microsoft (https connection is mandatory!) takes place to release the required permissions and request a code to authenticate the system. When authenticating the profile, the user created above for whom the token was created must be used.
6. If the permissions have been accepted, you will receive a positive confirmation.

If the token request was successful, there is a tick in the penultimate column of the profile overview. The tick only indicates that a token (access and/or refresh token) is available, but not whether they are still valid (on Microsoft's side). If, for example, you change the permissions in Azure, the token request must be carried out again.

> ⓘ **Info**
>
> The system must be able to access "https://login.microsoftonline.com/". If necessary, a corresponding system proxy must be configured.

9.1.5.3   Step 3: Create email account

1. Navigate to *Communication >Email > Inbox*.
2. Click on "New account" in the table. A form dialogue opens in which you can create the email account.
3. Under "Type", select an option that ends in "_OAuth2".
   (**Note**: IMAPTLS_OAuth2 is no longer supported by Office365).
   After this selection, the profile field is displayed. The password field is hidden as it is not required.
4. Under "OAuth2 Profile", select the profile created in step 2.
5. Fill in the remaining form fields and save the mail account. If necessary, read up on this under
   Inbox (see page 201) .

9.1.5.4   Step 4: Configure email sending

1. Navigate to *Communication > Email > Outbox.* A form dialogue opens in which you can configure the outbox.
2. The form has the properties "SMTP OAuth2", "SMTPS OAuth2" and "SMTPTLS OAuth2" in the "Type" field.
   ⓘ In connection with Microsoft, "SMTPTLS OAuth2" must be used. Other types are not accepted.

3. As with the Inbox, either the OAuth2 Profile field or the Password field is displayed depending on the type selected. Select the profile created in step 2.

4. Fill in the remaining form fields and then click on "Save". If necessary, refer to Outbox (see page 219) .

> ⚠ **Important**
>
> The profile is stored in the SysConfig key "SendmailModule::OAuth2_Profile". If the profile is renamed, the profile must be set again in the configuration of the outbox or directly in the SysConfig!

9.1.6 Notes for Exchange/Office 365 Server

When using an Exchange/Office 365 server, a license must be stored in the email account (fetching user) so that the IMAP and SMTP services are available. The data required for setting up email communication in KIX can be taken from the user's settings. To do this, log on to the server with the email account and call up the settings (Options > Mail Accounts > POP and IMAP). Make a note of the information needed for the setup and store it in the Inbox and Outbox of KIX.

If no data is available, e.g. because the license is missing, this is marked with "Not available". In this case, activate the services again.

Fig.: IMAP is active, but SMTP is not

If the license is missing, KIX returns the following error screen:

- **Inbox:** After retrieving the emails, KIX reports a successful collection; however, the emails do not arrive in KIX and remain in the server's mailbox.
- **Outbox:** Error message when sending an email:
  "[Kernel::System::Ticket::Article::ArticleCreate][699] Impossible to send message to: empfaenger@company.de (Error: SMTP authentication failed: 5505.2.1 Mailbox cannot be accessed [AM6P194CA0084.EURP194.PROD.OUTLOOK.COM] !)."

9.1.7 SysConfig settings for emails

Below you will find an overview of selected configuration keys which are relevant for email traffic. You can find the configuration keys in the menu *System > SysConfig* using the search function.

SysConfig key	Description	Page links
sendmail	Set of SysConfig keys for outgoing mail configuration	Configuring the Email Dispatch (SysConfig) (see page 222)
ContactEmailUniqueCheck	Enables/disables the uniqueness check for contact email addresses. ⚠️**Once a validity check has been deactivated, it MUST NOT be reactivated!** It is imperative that you observe the information in the chapter: Allow contacts without or without a unique email address.	Allow contacts without or without a unique email address (see page 566)
Daemon::SchedulerCronTask Manager:: Task###MailAccountFetch	Defines the time interval for the mail retrieval. KIX automatically retrieves the email accounts at regular intervals. The time interval is initially 10 minutes. If necessary, you can change this default setting in this SysConfig key. To avoid too high a system load, the fetch interval should <u>not be less than 5 minutes</u>. We therefore recommend that you leave the interval at 10 minutes.	
FQDN	Complete domain name of the KIX Frontend	Section: Use a placeholder for the domain (see page 224)

SysConfig key	Description	Page links
PostmasterX-Header	Dynamic fields to be considered in mail filters • X-KIX-DynamicField-DFKeyName: sets DFKeyName on ticket creation • X-KIX-FollowUp-DynamicField-DFKeyName: sets DFKeyName on ticket creation on follow up	Overview of Usable KIX Email Headers (see page 238)
IgnoreEmailAddressesAsRecipients	Contains a RegEx for a blacklist. The filter checks subsequently, after all other checks, whether an email address corresponds to the specified RegEx. If this is the case, the email is <u>not </u>sent - regardless of all other settings. This can be used, for example, to prevent replies to a "noreply" address.	
PostMaster::CheckFollowUp Module###.....	Checking incoming emails for possible follow-ups.	Notes on email distribution (see page 207)
PostmasterDefaultQueue	Standard team (default queue), in which the collected emails are created as tickets	Inbox (see page 201)
PostMaster::FollowUp::Check FromOrganisation	Controls the visibility in the SSP (KIX Pro). Checks whether the sender of a follow-up is contained in the contact database and is assigned to the same organisation as the ticket.	Control visibilities in the SSP
PostMaster::MailAccountFetch::Debug	Setting a gradual postmaster debug level for targeted log information.	System > Logs (see page 314)

SysConfig key	Description	Page links
PostMasterMaxEmailSize	Maximum size of emails that can be retrieved via POP3, POP3S, IMAP, IMAPS. Data in KBytes	16384
Ticket::EventModulePost### 900- ArticleCustomerVisibleByReci pient	Controls the visibility of the article in the Self Service Portal based on the e-mail recipient (KIX Pro).	Control visibilities in the SSP
Ticket::EventModulePost### 910- ArticleCustomerVisibleByRef erences.	Article event that controls the visibility in the SSP based on the email reference (KIX Pro).	Control visibilities in the SSP

9.2 Web forms

KIX offers you the possibility to generate web forms which you can integrate into your website. Via these web forms, your customers can send fault reports, feedback or ticket-relevant information directly to KIX. For each message received via a web form, KIX automatically creates a new ticket.

KIX generates 2 code snippets per web form. Integrate these into your website and thus make the form available on your website.

Abb.: KIX Webform

The table in the content area lists all the web forms created in the system:

Fig.: Overview Web Forms

> ✅ **Tip**
>
> Agents can be automatically notified as soon as a ticket has been received via a web form. To do this, set up a corresponding ticket notification (see page 184) in the *Automation > Notifications* menu.

9.2.1 Zoom View for a Web form

You can view the details of a web form. Navigate to *Communication > Web Form.* A table opens in the content area listing all the web forms stored in the system. Click on a web form in the table to open its detail view. Click on "Edit" in the title bar of the detailed view if you want to make changes to a web form.

The buttons in the title line of the detail view allow you to edit and create a new web form.

Fig.: Zoom View of a Web Form

The zoom view contains three lanes:

Lane	Description
Webform Options	General Information about the web form.
Default Values	Information passed to the tickets from the form.
Code	Code snippets for integrating the form into the website.

9.2.2 Creating a Web Form

You are able to generate web forms for your website that your customers can use to send fault reports or feedback directly to KIX. KIX generates two code snippets based on the parameters you set, which you can integrate into your website. The form is called up from your website via these two code snippets.

Fig.: Configuration of a web form

Depending on your requirements, you can create several different forms. You can insert them anywhere on your website or in different websites. For each form

- a code snippet is generated for the required scripts, which generate the form at runtime.
- a code snippet is generated for the button that triggers the display of the form.

The button can be integrated several times on one or more web pages. The script must be integrated once on all pages on which the button is located.

Fig.: Code snippet for embedding in the website

To create a new web form, proceed as follows:

1. In explorer, navigate to *Communication > Webform*. In the content area, a table listing all the web forms created in the system opens.
2. In table, click "New Webform". A form dialog for creating the new web form opens.
3. Complete form (see table below) and set validity to "valid".
4. Click "Save" to save new web form. The web form has now been created and the code snippets have been generated.

The code snippets for the script and the button can be found in the zoom view of the web form (in the "Code" lane). You can copy these to the clipboard and then integrate them into your website.

To integrate a web form into a website, proceed as follows:

1. In the explorer, navigate to *Communication > Web Form*. A table opens in the content area listing all the web forms created in the system.
2. Select the desired web form by clicking on the corresponding arrow on the right-hand side of the table. The code snippets open in a dropdown.
3. Include the scripts and the CSS file in your website:
    1. Copy the code snippet beginning with `<script>` completely to the clipboard.
    2. Paste the code snippet from the clipboard into the source code of your website.
       Place it in the area between `<head>` and `</head>`.
       The code snippet must be included once on all pages where the button is located.
    3. Replace the placeholder <YOUR_FRONTEND_HOST> with the URL of your KIX frontend domain.
       The code already contains the URL of the KIX frontend domain if it is stored in the Setup Assistant or in the SysConfig key "*FQDN*".
    4. Optional: replace the line `<link href="[...]kix-form.css" rel="stylesheet">` with the file path to your own CSS file if you want to use your own form design.
    5. Save the change to the website.

4. Include the button:
    1. Copy the code snippet beginning with `<button>` in its entirety to the clipboard.
    2. Paste the code snippet from the clipboard into the places in the source code of your website where the button to open the form should be located.
        The code snippet creates a button on the website, which opens the form when clicked. Each button can be integrated several times and on several pages.
        The ID of the button is used to reference the respective form (error message or feedback). Since each button has its own ID, several different buttons can be integrated on one website and thus several forms can be provided on one website. It is only important that the associated <script> code snippets (point 3) are integrated once on all these pages. Save the change to the web page.
5. Upload the changed web pages to your website host. The form can now be used on the website.

Fig.: HTML code of a website with integrated form

To edit a web form, proceed as follows:

1. In explorer, navigate to *Communication > Webform*. In the content area, a table listing all the web forms created in the system opens.
2. In table, click web form to be changed. A form dialog for editing the web form opens.
3. Amend information as required and click "Save" to apply changes.

Your changes take effect immediately. You may also need to correct or re-paste the modified code snippets on your website.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name Activate Button	The name specified here is shown on the button that calls up the form, e.g. "Feedback" or "Write message".
Form title	The form title is shown in the header of the web form.
Show title in form	If you have checked off this option, the text saved in the "Form title" field will be shown in the header of the web form.
Name Submit Button	The name specified here is shown on the "Send" button of the form, e.g. "Send" or "Submit".
Information text	Freely definable text for information in the form.

Field (selection)	Description
Info after Sending	Freely definable text shown after sending the form, e.g. "Thank you for your inquiry. We will contact you as soon as possible."
Start Modal Dialog	If you check off this option, the form will open in a lightbox (modal). The website in the background of the web form will remain inactive while the dialog is open.
Use KIX CSS	If you check off this option, the web form will be shown in the KIX design. You can use your own CSS files to design the form. To do so, modify line of code `<link href="[…]/kix-form.css" rel="stylesheet">` in code snippet by referencing your CSS file.
Enable attatchments	If you check off this option, users can add attachments to tickets created via the web form.

Field (selection)	Description
Accepted domains	Enter the domain of your website here, e.g. "https://www.MyHomepage.com". You can specify multiple domains by separating them with a comma or semicolon.

Use KIX CSS:

Enable attachments: ☐

* Accepted domains: ⊘ `/.*/`

* Validity: ⊘ `invalid ✕` ▾

It is possible to enter regular expressions (RegEx). These must be placed in slashes ("/"). KIX will check your entries for use of the correct RegEx syntax. KIX will set any missing slashes and change your entries into a regular expression that the web form can interpret. For example, an entry of * or .* or .+ will be changed to /.*/. You cannot use commas within the RegEx syntax because KIX interprets them as domain separators.

Entries in the following formats will be ignored and will not be saved, even if they are part of a list separated with commas or semicolons:

- / (any number of slashes)
- Only blanks
- Combination of slash and blank (e.g. /// or // ///// or / /// // /// or / /)
- `*` or `.*` or `.+` as part of a list; otherwise KIX will automatically correct the expression (see above)
- Example: An entry of `/my\.domain/;/ /;//;*` would be changed to `/my\.domain/`

Examples of valid entries:

- my.domain.com; /test/; /.+\.my\.domain/
- /.*/
- *
- /foo/,/bar/

Examples of invalid entries (invalid RegEx):

- my.domain.com; /test(123/; /.+\.my\.domain/ (missing closing bracket)
- /*/

Field (selection)	Description
	ⓘ **Note on existing data** Existing values are prepared internally as RegEx. A validity check is not performed. A previous value e.g. "my.domain", "/test.+123/", "foo", "my.second-domain.com" is changed to the following regex: "(my.domain|test.+123|foo|my.second-domain.com)". So KIX takes all the values, removes any slashes (/), joins the values with | and puts the expression in parentheses. This does not represent a migration. The value is only adjusted in the code, but not stored by the system. **Tip:** Since the adjusted value is displayed in the edit dialog, it only needs to be applied with "Save".
Validity	• **valid:** The web form is valid and can be used. • **invalid/invalid-temporarily:** The web form is invalid and (temporarily) cannot be used.
Team	Specify the team that tickets created via the web form should be assigned to.
Priority	Specify the priority that tickets created via the web form should be assigned.
Type	Specify the type that tickets created via the web form should be assigned to.
State	Specify which state is assigned to tickets created via the web form.
Assigned agent	In order for tickets to be created in KIX via the web form, an agent whose role is at least "Ticket Creator" must have been assigned on the form. If possible, do this by creating a separate agent that is not a natural person (user).

Field (selection)	Description
Password	In this field, save the password of the assigned agent so that tickets can be created via the web form. **Please note:** If you changed the agent's password in the agent's settings, you will also need to change it here so that the agent concerned can continue to receive messages via the web form. However, since the agent's password should be changed at regular intervals, we recommend creating a separate agent (who is not a natural person) to receive messages from the web form.

10 Internationalisation

KIX supports multiple languages. KIX comes supplied with 2 languages: German (de) and English (en_US). The source language is English. The German language library is additionally supplied for German-speaking countries, ensuring that the texts and designations for menu entries, buttons, help texts, etc. that are provided by default are automatically translated into German if "German" has been set as the user's language. The texts and designations comprise basic character strings, which are also referred to as **patterns**.

The menu *Internationalisation > Translations* enables you to edit and add to the existing patterns in the system as well as all patterns generated by the user (designations for assets, dynamic fields, notifications, etc.).

Fig.: The use of multiple languages in KIX is managed using translations

The language that has been set for a user is the decisive element determining whether a translation is used. If a particular basic character string does not have a corresponding character string saved in a different language, then the system will reference the English language. If a particular basic character string does
have a corresponding character string saved in a different language, the character string of the language assigned to the agent will be referenced. This applies to all texts and designations in the system, and to all content generated by the user (assets, asset classes, dynamic fields, notifications, etc.).

The language used by a user is preset when creating/editing the user (menu User *Management > Users*). In their preferences, each agent can choose for themselves which language they would like to use for KIX.

10.1 Translations

The menu *Internationalisation > Translations* enables you to add to or change the translations
(patterns) specified by the system. This concerns both existing language schemes and the creation of new
language schemes, such as:

- the texts and designations in the system (menu entries, buttons, help texts, widget titles, table
  headers, etc.)
- the ticket notifications that you have created
- the designation of the attributes that you use in asset classes (form fields)
- the designations of the assets, teams, states, priorities, and dynamic fields that you have created
- and the like.

No translation is provided for signatures. Since they are linked to a team, signatures are generally
also created in the "team's language". Likewise, no translation is provided for text modules either. Instead,
a separate text module is created for each language.

In their preferences, each user (agent or contact) can specify which language they would like to use for
their work in KIX. The language is preselected when creating or editing a user. The language setting
determines not only the designations in the user interfaces, but also the notifications that are used. If the
user has not set a language scheme or if they have not been authenticated yet, the browser's language
identifier will be used. If this cannot be assigned or if no corresponding scheme exists, then the system's
default language (global configuration) will be used.

The table in the Dashboard lists the patterns based on a search term. In search field, enter term to search for
specific patterns. You can use wildcards such as an asterisk (*) to do this, e.g. "ticket*". To display
all patterns, enter only asterisk in search field. It may take a moment for all patterns to load. The
column "Languages" indicates for which languages translations already exist.

Fig.: Overview of selected Patterns

10.1.1 Creating or Editing Translations

You yourself can maintain the translations for the basic character strings (patterns) used in KIX. This concerns both existing language schemes (labels, help texts, inscriptions, etc.) and the creation of new language schemes (attributes in asset classes, designations of dynamic fields, ticket notifications, etc.). Patterns are basic character strings that can be made up of individual words or entire texts.

Fig.: New Pattern for Dynamic Field

By using HTML tags, you can format both the texts of the patterns and those of the translations (e.g. <i>*italics*</i>). The use of a string placeholder (%s) allows variable text in the pattern.

To create a new pattern, proceed as follows:

1. In explorer, navigate to Internationalisation > Translations.
2. In content area, click "New Translation". A form dialog for creating the pattern and the corresponding translations opens. A separate text field is available for each language.
3. Under "Pattern", enter basic character string requiring translation.
4. Enter translation for each language into which pattern is to be translated. A separate text field is available for each language (see info box below).
5. As the last step, click "Save" to save your entries.

The system can now use the translation of the new pattern. Agents may need to refresh their system for the translation to take effect for them.

To edit a pattern, proceed as follows:

1. In explorer, navigate to Internationalisation > Translations.
2. Enter search term to find required pattern. If necessary, use asterisk (*) for search.
3. In table, click pattern to be edited. A form dialog for editing the pattern and the correspondingtranslation opens. A separate text field is available for each language.
4. Change or add information as required and, as the last step, click "Save" to save your entries.

The system can now use the amended translation of the pattern. Agents may need to refresh their system for the translation to take effect for them.

> ⓘ **Note**
>
> You can create additional languages in the SysConfig key "*DefaultUsedLanguages*". For these, however, the patterns have to be maintained manually.
> If you have created additional languages, you can specify the links to the manuals on the welcome page for each language. Use the following SysConfig keys: *KIX::SelfServiceManual, KIX::UserManual, KIX::AdminManual*.

11   Knowledge Database

The knowledge base contains a compilation of frequently needed information and answers to frequently asked questions (FAQ). The FAQ articles can cover a wide range of topics: Notes on working with KIX, product data sheets, installation instructions, special features to be considered on a machine, and much more. So that all this information does not lie unordered in the system, it is grouped into FAQ categories and thus structured thematically.

As administrator, you create the FAQ categories and their structure. The individual FAQ articles are maintained by the authorised agents and assigned to the categories. A FAQ article is assigned to a category when it is created or edited.

The FAQ categories are created in the Admin module in the menu *Knowledge Base > FAQ Categories*. When delivered, KIX already contains the category Misc, to which you can add further categories and subcategories. By assigning FAQ categories to superordinate FAQ categories, a structure tree is created, which the agents can find again in the FAQ dashboard to view and edit FAQ articles there.

Fig.: FAQ Categories Shown in FAQ Dashboard

Agents can search for FAQ articles via the complex search and limit their search to a FAQ category. In addition, the sidebar widget "Recommended FAQ" displays relevant FAQ articles for the ticket. The table entries in the widget initially result from the keywords entered in the ticket subject/title plus the relevant assets selected in the ticket. You can change this configuration.

11.1   Creating or Editing a FAQ Category

To create a new FAQ category, proceed as follows:

1. In explorer of the admin module, navigate to *Knowledge Database > FAQ Categories.* In the content area, a table listing all FAQ categories created in the system opens.
2. In table, click "New Category". A form dialog for creating a new FAQ category opens.
3. Complete form, select icon if required, and set validity to "valid".
4. If required, select existing FAQ category in selection field "Parent Category" in order to assign new FAQ category to it as a sub-category.
5. Click "Save" to save new FAQ category.

The new FAQ category has now been created and is available to agents in the FAQ explorer for creating FAQ articles.

To edit a FAQ category, proceed as follows:

1. In explorer of the admin module, navigate to *Knowledge Database > FAQ Categories*. In the content area, a table listing all FAQ categories created in the system opens.
2. In table, click FAQ category to be changed. The zoom view for the category opens.
3. In opened zoom view, click "Edit". A form dialog for editing the FAQ category opens.
4. Amend information as required and click "Save" to apply changes. The changes take effect immediately.

To delete a FAQ category, proceed as follows:

1. In explorer of the admin module, navigate to *Knowledge Base > FAQ Categories*. A table will open in the content area listing all FAQ categories created in the system.
2. Place a tick (1st column) on each category to be deleted.
3. Click on "Delete" in the table header. The button is only active if at least 1 category is selected.
4. Confirm the security query with "Yes" to carry out the deletion process.

Note: FAQ categories are deleted recursively, i.e. including all subcategories. Categories can only be deleted if they do not contain FAQ articles. If necessary, assign the articles to another category. As an alternative to deletion, you can also set FAQ categories to "invalid" (see table below).

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Give the category a meaningful, unique name so that agents can sensibly assign the FAQs.
Icon	The icon is used to graphically identify the category. It is used, for example, in the explorer of the FAQ Dashboard and in the overview of FAQ categories in the Admin Dashboard. You can choose an icon from one of the provided libraries or upload a graphic file.
Parent Category	By selecting a parent category, you can generate a tree structure like the one used in an explorer. If possible, do not break down the categories too far, so as to maintain a clear structure.

Field (selection)	Description
Validity	• **valid:** FAQ articles can be assigned to this category and are available in the explorer of the FAQ dashboard. • **invalid/temporarily invalid:** No FAQ articles can be assigned to the category. The category is no longer listed in the explorer of the FAQ dashboard. The FAQ articles of this category can now only be found via the complex search.

12 System

KIX enables you to intervene directly in the system configuration and the core. As a result, you can tailor the system to meet the specific needs of your company and, if required, also execute commands at command line level. Please do not hesitate to contact us if you require any assistance. We would be glad to help you.

> ⬥ **Attention!**
>
> Commands that you execute via the console are considered fundamental interventions in KIX, as are changes to the system configuration. We would like to highlight that you should only use these functions if you have sufficient expertise and provided that you take great care. Otherwise, these actions may result in serious **system errors** or even **complete system failure.**

Dynamic Fields

The menu *System > Dynamic Fields* enables you to create the specific input fields and selection fields and checklists you need in order to expand the information that can be saved for a ticket. By Configuration of User Interface (GUI) , the dynamic fields created here are integrated into the program interface (menu *System > SysConfig*).

Console

The menu *System > Console* contains a command console with a graphical user interface for executing predefined commands at command line level. However, this is predominantly reserved for our Support team, to ensure that they can intervene in the KIX core if required.

Logs

KIX logs all system messages and saves them in what are known as log files. The log files are available at *System > Logs*. You can view and download them at any time, e.g. for troubleshooting purposes.

SysConfig

The menu *System > SysConfig* contains the system configuration, which is where you can configure general settings in order to tailor KIX to your needs. For example, you can

- save the default team for receiving new tickets
- save the default email address from which all outbound emails are sent
- amend the program interface (GUI configuration), by e.g. integrating dynamic fields or
- changing views and widgets as required
- amend the widgets in the sidebar
- connect an LDAP/AD server to manage users.

12.1 Console

In the *System > Console* menu you will find a graphical user interface for executing system commands on command line level. Here you can select from predefined commands and execute them. The console allows you to import, create and save complete data packages and to migrate from KIX 17 to KIX 18.

Fig.: Console

> ⚠️ **Attention!**
>
> **Once a command has been started, it must be executed! It can neither be aborted nor undone.**
> Commands executed via the console are a fundamental intervention in the core area of KIX. The console is therefore mainly reserved for our support or only for very experienced administrators. We would like to point out that you should only use the console with sufficient knowledge and with caution. Otherwise, serious **system errors** or even **system failure** may result.
> Please do not hesitate to contact us if you need assistance. We will be glad to help you.

12.1.1 Using the Console

The command console in the menu *System > Console* enables you to execute predefined console commands using a graphical user interface. Only use the console if you have sufficient expertise in how to use it!

Only use the console if you have sufficient knowledge!

Fig.: Console Structure

Field	Description	Figure
1	**Choose a command.** Select command to be executed from selection field.	
2	**Help button** Clicking the Help button displays information to assist you. If no command has been selected, the info area shows all commands that can be executed. Alternatively, you can also execute the command "help". If a command has been selected, the info area shows the arguments and parameters for the selected command.	
3	**For entering a command** Enter command to be executed here. Then click "Execute" to start execution of command. The info area shows the result of the execution.	
4	**Info area** Displays the possible commands and parameters as well as the result of the execution.	

Field	Description	Figure
5	**"Execute" button** Click button to start execution of command.	

> ✔ **Tip**
>
> Execute the console command `Console::Command::Lis` t to get a list of all console commands including explanations.

12.2   Dynamic Fields

You can integrate additional input and selection fields into the program interfaces and thus store individual information at an object. These additional fields are called dynamic fields in KIX.

It is thus possible to store additional information on tickets, organisations or contacts, for example:

- Date of commissioning
- Serial numbers
- Customer and vendor numbers
- Status of AV agreements
- VIP identifier
- newsletter subscriptions
- extended notes
- and much more.

This information is then available when editing, viewing and using organisations, contacts and tickets.

Dynamic fields can be included in the following objects:

- all ticket-relevant interfaces (e.g. dialogs "new ticket" and "edit ticket")
- FAQ articles
- Organisations
- contacts.

The values stored in dynamic fields are directly linked to the object. This makes it possible to search specifically for this information, evaluate it, and display it in overviews, detailed views, and in the sidebar.

Dynamic fields are first created and configured in the *System > Dynamic Fields* menu. Then, the Dynamic Fields are integrated into the user interface at the desired location by configuring the corresponding system keys (*System > SysConfig* menu).

Each Dynamic Field can be integrated at several places, so that the data stored in it is available at all relevant places. For example, a dynamic field that is integrated into the ticket creation screen can (and should) also be integrated into the ticket editing screen, and the values entered in the dynamic field can be displayed in the ticket detail view.

Furthermore, dynamic fields are used in jobs and ticket notifications. In a job, for example, it can be defined that a predefined value is set in a dynamic field under certain conditions. In ticket notifications, for example, the value of a dynamic field can be read out and communicated in the message.

The values of dynamic fields can be referenced via KIX placeholder. Use the following syntax for this: <KIX_[KIX-OBJECT]_DynamicField_[DynamicFieldName]_[Value]>. Examples:

<KIX_TICKET_DynamicField_AffectedAsset> or <KIX_TICKET_DynamicField_AffectedAsset_ObjectValue>. In the appendix you can find an overview of possible KIX placeholders.

KIX Pro users can additionally use dynamic fields in ticket actions or in ticket templates.

A complex search for the contents of a dynamic field is also possible. Dynamic fields can also be made available in the Self Service Portal, so that your customers can save information on the ticket and see the saved information.

Fig.: Dynamic Field as Additional Input Field in Ticket Creation Screen

> ⓘ **Note**
>
> Invalid and temporarily invalid dynamic fields are not used and considered when executing:
> - Ticket notifications
> - Search and filter tickets
> - in jobs

12.2.1 Creating, Editing and Incorporating Dynamic Fields

Before a Dynamic Field can be included in the program interface, it must be created and configured in the *System > Dynamic Fields* menu. You can create different types of Dynamic Fields, depending on the type of data you want to capture. You can choose between

- Single-line and multi-line text fields
- Select fields (dropdown with single and multiple selection)
- Date and date-time fields
- Checklists and
- Asset reference fields
- Ticket reference fields (KIX Pro).

See the Object and Field Types of Dynamic Fields chapter for an overview of these.

Fig.: Creating a Dynamic Field of the Type "Text" (Example)

To create a new dynamic field, proceed as follows:

1. In explorer, navigate to *System > Dynamic Fields.* In the content area, a table listing all the dynamic fields created in the system opens.
2. In table, click "New Field". A form dialog for creating a new dynamic field opens.
3. Complete the form (see table below) and configure field parameters (also see "Object and field types of dynamic fields ").
4. Set validity to "valid" and click "Save" to save new dynamic field.

The dynamic field is now created and can now be integrated into the program interfaces via the configuration of the corresponding keys in the *System > SysConfig* menu. The chapter "Configuration of the user interface (see page 327) " is dedicated to this topic in detail.

To edit a dynamic field, proceed as follows:

1. In explorer, navigate to *System > Dynamic Fields*. In the content area, a table listing all the dynamic fields created in the system opens.
2. In table, click dynamic field to be changed. A form dialog for editing the parameters of the dynamic field opens. However, you cannot change the field type or object type, as doing so would remove the underlying basis for the data that was previously saved.
3. Amend information as required and click "Save" to apply changes.

The changes are now effective.

To incorporate a dynamic field in the program interface, proceed as follows:

After creating a dynamic field, you can incorporate it in all ticket-related program interfaces. This is done by editing the configuration keys in the menu *System > SysConfig*. To do this, please read chapter "Integrating a Dynamic Field (see page 423) ".

The form dialog contains the following input fields, among others:

Field (selection)	Description	
Name	The name acts as an internal identifier and is only used in the Admin area.	
	The system permits uppercase and lowercase letters, and numbers, but not blanks or special characters. Each dynamic field is incorporated in the program interface by referencing its name. Therefore, please use a meaningful, unique name. If you change the name retrospectively (which is not recommended), you will also need to change all interfaces, jobs, notifications, etc. which reference the field.	
Label	The label is shown in front of the field. It describes the expected content of the dynamic field.	
Field Type	The field type depends on what information is stored in the dynamic field. For detailed information on the field types and their configuration, see Object and field types of dynamic fields (see page 281)	
	AssetReference	Generates a selection field for selecting multiple assets.

Field (selection)	Description	
	Checklist	Generates a list of tasks in the form of a checklist, which users can answer and comment on.
	Date	Date field; enables the user to enter a date.
	DateTime	Date and time field; enables the user to enter the date and time.
	FAQ Article Reference (KIX Pro)	Creates a selection field for selecting FAQ entries.
	Contact Reference (KIX Pro)	Creates a selection field for selecting Contacts.
	Organisation Reference (KIX Pro)	Creates a selection field for selecting Organisations.
	Selection	Selection field; contains a list of predefined values that an agent can choose from.
	Table	Table; enables data in a ticket to be entered in tabular form.
	Text	Single-line text field; for recording a word or a line of text.
	Text Area	Multi-line text field; for recording a longer text.
	Ticket Reference (KIX Pro)	Creates a selection field for selecting Tickets.

Field (selection)	Description	
Object Type	Your selection determines where or in which context the dynamic field can be included. The selected object type is binding for the dynamic field. For example, a dynamic field of the "Ticket" object type can only be included in ticket-relevant interfaces, but not in FAQs or organizations, for example. For detailed information on the object types and their use, see Object and field types of dynamic fields (see page 281) .	
	Contact	The dynamic field can be included in contacts. This allows • the contact data available in KIX can be supplemented with additional, individual data • extended contact data from other systems are also available in KIX.
	FAQ	The dynamic field can be included in FAQ articles. In connection with the field type "AssetReference", assets can be referenced to FAQs so that, for example, instructions, product data sheets, etc. can be displayed in the sidebar when an asset is selected ("RelatedAssets" dynamic field).
	Organization	The Dynamic Field can be integrated into organizations. This allows • the existing organizational data in KIX can be supplemented with additional, individual data • extended organizational data from other systems are also available in KIX
	Ticket	The dynamic field can be integrated in all ticket-relevant interfaces (e.g. ticket creation mask, ticket edit mask, lane "ticket information", etc.).

Field (selection)	Description	
Show in Self Service Portal	You can specify whether a dynamic field that has been incorporated into the user interface should also be available in the self-service portal. If you check off this option, customers, too, will be able to enter data into this field and view the information saved therein. If you do not check off this option, then the dynamic field will only be available in the agent portal.	
Validity	valid:	The dynamic field is active; agents can save information in it and the values can be used for analyses.
	invalid/ invalid-temporarily:	The dynamic field is not shown anywhere. However, any values that were already saved remain "hidden" in the system so that you can run an Advanced Search for these values. In the Admin module you can also select invalid fields, which means you can create elements as part of preparatory work, for example.
Configuration	After selecting a field type, the corresponding configuration options are loaded. An overview of the configuration options is provided in "Object and field types of dynamic fields (see page 281) "	

12.2.2 Object and field types of dynamic fields

When creating Dynamic Fields, you must select both an object type and a field type. Your selection determines the context in which a Dynamic Field is used, as well as its appearance and behavior.

Object type

The selection of the object type depends on where the Dynamic Field is to be included. It is binding for the dynamic field. Dynamic fields of the "Ticket" object type can only be included in ticket-relevant interfaces (dialogs "New ticket", "Edit ticket", zoom view, etc.), but not in FAQ articles or in organizations, for example.

Regardless of the object type, the values stored in the dynamic fields can be displayed to the sidebar widgets.

Field type

The selection of the field type depends on what information is stored in the Dynamic Field. Different configuration options are available depending on the selected field type. The configuration of a Dynamic Field determines its display and behavior in the interfaces.

Content on this page:

The following overviews provide you with an overview of the possible object and field types as well as their use and configuration options.

12.2.2.1   Object types

Object type	Description
Contact	The dynamic field can be included in contacts. Dynamic fields of this object type allow e.g. • the addition of further, individual data to the contact data available in KIX (e.g. separate address) • information about the origin of the data record, e.g. from which of the connected Active Directories • that extended contact data from other systems are also available in KIX • individual identifiers such as participation in the newsletter, priority identifier ("A customer"), VIP identifier. Scheme for placeholder: `<KIX_CONTACT_DynamicField_XYZ>` (XYZ = name of dynamic field)
FAQ	The dynamic field can be included in FAQ articles and in the sidebar. In connection with the field type "AssetReference", assets can be referenced to FAQs, so that, for example, instructions, product data sheets, etc. can be displayed in the sidebar when an asset concerned is selected ("RelatedAssets" dynamic field). Scheme for placeholder: `<KIX_FAQ_DynamicField_XYZ>` (XYZ = name of the dynamic field)

Object type	Description
Organisations	The dynamic field can be integrated into organizations. Dynamic fields of this object type allow e.g. • the recording of extended information such as organization type (supplier, customer, third party) • the addition of further, individual data to the organizational data available in KIX • information about the origin of the data record, e.g. from which of the connected Active Directories • that extended organizational data from other systems are also available in KIX • individual identifiers such as the existence of an AV contract, priority identifier ("A customer"), VIP identifier. Scheme for placeholder: `<KIX_ORG_DynamicField_XYZ>` (XYZ = name of dynamic field)
Ticket	The Dynamic Field can be included in all ticket-relevant interfaces (e.g. ticket creation mask, ticket edit mask, lane "ticket information" etc.), but not in articles, organizations, contacts or FAQ. Scheme for placeholder: `<KIX_TICKET_DynamicField_XYZ>` (XYZ = name of the dynamic field).

12.2.2.2   Field types KIX Start

All field types

The following configuration options apply to all field types except type "Checklist" and "Table".

Configuration options

Option	Description
Name	Unique identifier of the Dynamic Field. The name is used in KIX placeholders to reference the dynamic field, e.g. `<KIX_TICKET_DynamicField_DFSerialNumber>.` The name is also used when you integrate the dynamic field into the interfaces (e.g. dialogue "new ticket" or ticket details).

Option	Description
Label	Labelling of the dynamic field in the interfaces. The label is placed in front of the dynamic field. It serves as orientation for the user as to which entries are expected in the dynamic field or which data can be selected. The integration of the dynamic field in templates or actions and the use in filters of jobs is based on the label. ⚙ New Job Filter ‹ • • ● • › Filter: ⑦ seria Serial number ‹ • • ● • › Filter
Field type	The field type determines the data basis of the dynamic field (see top of page).
Object type	Defines the context in which the dynamic field is used (see top of page).
Show In Customer Portal	Check the box if the dynamic field should also be available in the Self Service Portal. This is particularly necessary when you create templates or actions for the Self Service Portal. If the tick is not set, the field is not displayed in the Self Service Portal.
Validity	• valid: The dynamic field is active and can be used according to its configuration. • (Temporarily) invalid: Invalid and temporarily invalid fields are not displayed in the interfaces and cannot be used by actions, jobs or similar. Data already stored on the object is retained and is available again in the event of reactivation.

Option	Description
Configuration	The configuration options available here depend on the selected field type. However, some options are valid for all field types except checklist and table:

Count Min	The number entered specifies the minimum number of times that the field should be included in the form.
	"0" – The field is not shown when loading the form, but can be displayed by clicking "+" "1" – The field is contained once in the form. "2" – The field is contained twice in the form (etc.)
	Exception: In selection fields (e.g. types Selection), this specifies the minimum number of values to be selected. This means that the field becomes a mandatory field if the value is > 0.
	Examples:
	- CountMin = 1, CountMax = 2: In the dialog, at least one of the maximum 2 possible selection fields is shown. - CountMin = 0, CountMax = 1, CountDefault = 0: In the dialog, only the label is shown, without the selection field. The "+" symbol for displaying the selection field is provided.

Option	Description	
	Count Max	The number entered specifies the maximum number of times that the field is permitted to appear in the dialog. If it is permitted to appear several times, a plus and a minus symbol will be shown after the field so that the field can be duplicated for further entries. However, this number specifies the maximum number permitted. "0" – The field is not permitted to be duplicated. "1" – The field is permitted to appear only once. "2" – The field is permitted to be supplemented by one more field. "3" – The field is permitted to be supplemented by two more fields. (etc.) ***Example:*** *Field "Assigned Contact" When Creating a New Asset* **Exception**: In selection fields (e.g. Selection or AssetReference types), this number specifies the maximum number of selectable values. If you enter a number > 1, the selection field will become a multiple selection field.
	Count Default	The number entered here specifies how often the field is shown initially when the form is called up. **Example**: CountMax = 5, CountDefault = 5: Initially, all 5 input fields are shown at the same time in the form.

Option	Description	
	Item Separator	Separator symbol for displaying multiple values on a single line (generally a comma or semicolon). If several values are entered into the field (e.g. multiple selection for the "Select" and "AssetReference" fields), the individual values will be concatenated with this separator symbol if the values are being shown on a single line (e.g. in tables).
	Default Value	Preassigned value in the field. The field is populated with this default value if no other value is entered. **Exception:** In the field types "Date" and "Date / Time", the time is predated in this way (current + number of seconds).

Text and Textarea

In single-line text fields of the type "Text", you are only able to enter a word or sentence.

Country:

In multi-line text fields of the type "Textarea", you are able to save longer texts.

Comment: ⑦

Configuration options (selection)

Option	Description
RegEx List	Regular expressions (RegEx) can be used to check whether the entries in text fields are valid. For example, they allow you to specify that certain expressions in the text will not be accepted, or that certain characters are essential (e.g. the @ character in email addresses). A small selection of regular expressions is available in the appendix at: Regular Expressions (see page 744) . Click "Add item" to save one or more regular expressions and the associated error messages (optional). **RegEx:** Here, enter regular expression against which entered text should be checked. You can use multiple regular expressions. E.g.: " `^[A-Za-z0-9]+$` " to permit a character string of any length comprising letters and numbers. **RegExErrorMessage:** Here, enter text displayed as error message if text entered in text field contradicts regular expression.

Selection

The field type "Selection" generates selection fields for both single selections and multiple selections.
The values to be selected are specified when creating the dynamic field.

Configuration options (selection)

Option	Description
CountMin	The number entered specifies the minimum number of selectable values. If the value is > 0, the field becomes a mandatory field.
CountMax	Indicates how many values can be selected simultaneously. Entering a value > 1 will generate a multiple selection field. At the same time, the value entered corresponds to the maximum number of selectable values. **Examples:** • Count Max > 1: A maximum of "Count Max" selection values can be selected (multiple selection field with maximum selection). • Count Min > 0: At least the "Count Min" number of values must be selected. • Count Max < 0: Any number of selection values can be selected (multiple selection field) • Count Max ==1 OR Count Max ==0 or Count Max == NULL: Precisely one value can be selected (single selection)
Translatable Values	If you check off this setting, the values will be translated into the user's language. **Please note:** You may need to manually add the translations to the translation files.

Option	Description
Possible Values	Here, save the possible selection values that are listed in the selection field. Click "Add item" (several times if required) to save these values as key value pairs. Click "Delete item" to delete selection value from selection field. **Key:** internal value for processing the selection value. This value is only permitted to occur once in the selection field. A key must be specified. **Value:** value displayed; the agent sees this when they open the selection field.

Date and Date/Time

Dynamic fields of the types "Date" and "Date/Time" generate date fields in the form. You can use them to save scheduled completion dates or agreed delivery dates in the ticket, for example.

In fields of the type "Date", the date can be entered in the format "DD.MM.YYYY".

Warranty Expiration Date:	TT . MM . JJJJ

In fields of the type "Date/Time", the date and time can be entered in the format "DD.MM.YYYY" and "hh:mm".

Plan Begin:	11 . 01 . 2021	15 : 30	⊟

Date and DateTime fields do not support dates beyond the year 2200.

Configuration options (selection)

Option	Description
Default Value	Initial preassigned value for the time. If you do not enter a value, the current date will be shown. Type "Date": 0 = current date, 1 = tomorrow, 2 = day after tomorrow Type "Date Time": If you enter a number in seconds, it will be added to the current time (3600 = current + 1 hour; 86400 = current + 1 day).
Years in Future	Number in years Specify how many years in the future the calendar should stretch, starting from the current date.
Years in Past	Number in years Specify how many years in the past the calendar should stretch, starting from the current date.

Option	Description
Date Restriction	none: No restrictions apply when selecting a date. DisableFutureDates: Stops the user entering dates that are in the future, starting from the current date. DisablePastDates: Stops the user entering dates that are in the past, starting from the current date.

Checklist

Checklists enable you to integrate a list of predefined tasks in the ticket. For example, service technicians can use them as a guideline for ensuring they provide the service in full and thus comply with quality standards. Service technicians can work through and answer the checklist items one by one and add a comment for specific items if required.

Checklists could also be used to create a conversation guideline to help the agent ask targeted questions in order to localize the reported damage and find out the specifics. In this case, the conversation guideline could, if required, be shown via a template in or action on the ticket, for example, and used (KIX Pro).

A progress bar shown in the ticket screens and on the Dashboard indicates how much of the checklist the user has worked through. However, checklists could also be used for other workflows, such as for putting together all documents required to deliver a machine, or the completion of interdisciplinary tasks.

The status of a checklist can also be displayed in tables. An example of the integration can be found under: Configuration of dashboard tables .

Fig.: Example of a Checklist

Configuration options (selection)

Note: The configuration options Count (min), Count (max), Count (default) etc. are not available in checklists.

Option	Description
ID	Any desired value for internal processing purposes (optional). The ID is only permitted to occur once in the checklist. If no ID is entered, it will be automatically assigned by the system.
Title	The title identifies a checklist entry ①
Description	More accurate description of the checklist entry (optional) ②
Input	Specifies whether a selection field or text field is shown to answer the checklist item. **Text:** The system shows a single-line text field in which the user can save notes, for example. E.g. issues that occurred when performing work . ③ **Textarea:** A multiline text field is displayed where users can enter longer texts including line breaks. **ChecklistState:** The system shows a selection field enabling the state of the checklist item to be selected: ④ • Completed/OK (OK) • Not completed/not OK (NOK) • Waiting (Pending) • Not specified (n. a.) • Not required
Value	Preset value (optional). The field is pre-populated with this value or text.
Sub	Creates further sub-items for a checklist entry, thereby generating a hierarchical tree structure for the checklist.
Add Item	Adds further checklist entries.
Delete Item	Deletes the checklist entry including any sub-items.

Advanced configuration option

Checklists can also be created or assigned via automation (macro action "Set dynamic field"). In this way, a checklist field can be used in multiple forms and multiple configuration is avoided. The following example shows the complete value to be assigned in a macro action.

Example of checklist content Preboarding when hiring new employees (technical)

```
[
  {
    "id": "100",
    "title": "Send informationen",
    "description": "Send/call a welcome and information email before First Day.
Provide important information: where, when, expected by
     whom, corporate mission statement, organizational chart or communication
guidelines. Aim: to clarify general questions in advance.",
    "input": "ChecklistState",
    "value": "-"
  },
  {
    "id": "200",
    "title": "Get documents",
    "description": "Demand the involvement of the employee. Social security card,
health insurance, income tax card, tax ID. Target:
     Avoidance of organizational questions in the first week.",
    "input": "ChecklistState",
    "value": "-"
  },
  {
    "id": "300",
    "title": "Name contact person",
    "description": "Fixed contact person who is at your side - also with inquiries of
all kinds in advance - aka mentor / godfather / buddy.",
    "input": "ChecklistState",
    "value": "-",
    "sub": [
      {
        "id": "310",
        "title": "Buddy",
        "input": "Text",
        "value": ""
      }
    ]
  },
  {
    "id": "400",
    "title": "Setup workspace",
    "description": "The workplace must be set up on the First Day. These include:
provision of passwords, printer access, telephone,
```

```
    Office chair/desk, other aids. Goal: Avoid technical blockers in the first
week.",
    "input": "ChecklistState",
    "value": "-"
  },
  {
    "id": "500",
    "title": "Create an induction plan",
    "description": "Must be early, before starting work. Content: detailed work plan
for the first week; Tasks; future projects. Goal: prevent
     chaos and idle time.",
    "input": "ChecklistState",
    "value": "-",
    "sub": [
      {
        "id": "510",
        "title": "URL-induction plan",
        "input": "Text",
        "value": ""
      }
    ]
  },
  {
    "id": "600",
    "title": "Organize the search for accommodation",
    "description": "Support for outsiders in finding accommodation, moving or dealing
with authorities. Goal: Avoiding distraction, enabling
     Concentration on familiarization.",
    "input": "ChecklistState",
    "value": "-"
  }
]
```

References:

- General use of checklists: Use of checklists
- Application example: Configuring and providing checklists

AssetReference

Dynamic fields of the type "Asset Reference" create selection fields in which one or more assets (up to a maximum of 15) can be linked to a ticket or FAQ article. Such dynamic fields are, for example, the selection fields "Affected Asset" in tickets and "Related Assets" in FAQ articles. It is possible to adapt the configuration of these Dynamic Fields to the company's own requirements and to create further Dynamic Fields of the type "Asset Reference".

Dynamic fields of type "AssetReference" reference one or more assets. When used, the objects available for selection depend on the authorisations of the logged-in user and the field configuration (e.g. restriction to certain deployment states).

With KIX placeholders you can directly address each of these assets as well as attributes.'For notes on this, see Overview KIX Placeholders (section: Placeholders for referencing dynamic fields).

Fig.: Field "Affected Asset" as Example of Dynamic Field of Type "AssetReference"

Configuration options (selection)

Option	Description
CountMin	The number entered specifies the minimum number of selectable values. If the value is > 0, the field becomes a mandatory field.
CountMax	Indicates how many values can be selected simultaneously. Entering a value > 1 will generate a multiple selection field. At the same time, the value entered corresponds to the maximum number of selectable values.
Deployment States	Save one or more deployment states for assets here. The form field will then only show assets for selection if they have the deployment state specified here. To do this, click "Add item" (several times if necessary). Click "Delete item" to delete selection value. If no DeploymentStates are selected, there will be no restrictions and the form field will take into account all deployment states.
ITSMConfig-ItemClasses	Save one or more asset classes here. In the form field, only the assets of the asset classes specified here will then be available for selection. To do this, click "Add item" (several times if necessary). Click "Delete item" to delete asset class from selection. If no ITSMConfigItemClasses are selected, there will be no restrictions and the form field will allow the user to select the assets of all asset classes created in the system. However, the maximum limit is 15 assets.

Placeholder for Asset Reference Fields

You can access the assets selected in the dynamic field as well as their attributes via KIX placeholders. This allows you to reference this information in jobs or actions, for example, and make it available to other objects. How you can use the placeholders can be found in the overview of KIX Placeholders.

Linking assets to the ticket

The assets selected in a field of the type "AssetReference" are initially linked to the ticket as "Relevant to".
The SysConfig key "*Ticket::EventModulePost###ITSMConfigItemLinkAdd*" enables you to change the type of link.

```
{
    "Event": "(TicketDynamicFieldUpdate_AffectedAsset)",
    "LinkType": "RelevantTo",   //DependsOn|RelevantTo|RequiredFor|normal
    "Module": "Kernel::System::Ticket::Event::ITSMConfigItemLinkAdd"
}
```

Table

Dynamic fields of the type "Table" create tables in order to be able to save values in tabular form on the ticket. For example, to record the quantity, item and shelve number as well as a comment for inventory. The column headers are predefined when the dynamic field is created and cannot be changed in the editing screens.

Fig.: A dynamic field of the type "Table" in the new ticket dialog

Configuration options (selection)

Option	Description
Columns	Define the columns including headings here. Click on "Add item" to create further columns. Write the column heading in the text field of the item.
Number of rows (min)	Minimum number of table rows to be displayed. Values <1 are not possible.
Number of rows (init)	Number of table lines to be displayed initially. This way, you determine that, for example, the table is available with 3 lines when creating a new ticket.
Number of rows (max)	Maximum number of table lines. You thus specify that the table may contain a maximum of 5 lines, for example. The user cannot add more lines by clicking on the plus symbol.
Translate Table Columns	If activated, the column headings are translated into the language used by the user. Provided that the corresponding patterns have been stored under *Internationalisation > Translations*.

Integrating into the interfaces

The integration of the table into the KIX Start interfaces is basicly done as described under Integrating a Dynamic Field (see page 423) . In KIX Pro, the tables can be provided comfortably via templates and actions.

Tables can be declared as mandatory fields:

- KIX Start: Set the "required" parameter to "true".
- KIX Pro: Activate the checkmark in front of the dynamic field when creating the action or template.

If the table is declared as a required field, the table must not contain any empty rows and the table cannot be removed or shown.

In order to display the table in the ticket details, you can insert the following code block into the key "*ticket-details-info-card*" (KIX Pro: *Ticket Info Widget*). Proceed as described under Displaying Values of Dynamic Fields (see page 442) . Replace the exemplary descriptions "Inventory" or "DFInventoryList" with the real descriptions.

```
{
  "title": "Translatable#Inventory",
  "style": "",
  "separator": true,
  "values": [
    [
      {
        "componentId": "dynamic-field-value",
        "componentData": {
          "name": "DFInventoryList"
        },
        "conditions": [
          {
            "property": "DynamicFields.DFInventoryList",
            "operator": "NE",
            "value": null,
            "useObjectService": false,
            "useDisplayValue": false
          }
        ]
      }
    ]
  ]
},
```

KIX Placeholder

To access the data of the table, you can use the following placeholders:

- <KIX_TICKET_DynamicField_TableFieldKey>: displays the table in a plain text representation.
- <KIX_TICKET_DynamicField_TableFieldKey_Short>: represents the table in the form "X rows" (X = number of rows).
- <KIX_TICKET_DynamicField_TableFieldKey_HTML>: represents the table with heading and values in the body of the notification/article/email.

12.2.2.3   Field types KIX Pro

KIX Pro contains additional field types for the configuration of referencing dynamic fields. The placeholders for these fields return an array whose individual values can be addressed concretely. For notes on this, see Overview KIX Placeholders (section: Placeholders for referencing dynamic fields).

TicketReference

Dynamic fields of the type "TicketReference" create selection fields in which one or more tickets can be referred to (e.g. field "Related tickets"). The input behaviour corresponds to that of a "Select" field.

When used, the objects available for selection depend on the permissions of the logged in user and the field configuration (e.g. restriction on ticket types and status). Searching for tickets corresponds to a full-text search by ticket number or ticket title. You can also enter the username to search for tickets from that user.

Dynamic fields of the type "Ticket Reference" can be used, for example, to assign parent/child tickets, to establish structures for main/subtasks, dependencies or other references. The fields "Parent Tickets", "Child Tickets" and "Related Tickets" are delivered pre-configured. To use parent/child tickets, the fields must be added in the configuration of the input masks (see Actions).

Entering values in these fields automatically adds links between the current ticket and the tickets referenced in the fields according to the configuration of the dynamic field.

Fig.: The field "Related tickets" in ticket edit dialog as an example for a dynamic field of the type "TicketReference

Configuration options (selection)

Option	Description	Possible Values
CountMin	With the entered number you define the minimum number of selectable values. The field becomes mandatory if the value is >0.	0...n
CountMax	Indicates the maximum number of values that can be entered in a field. If a value >1 is set, a multiselect field is created.	0...n
Ticket States	Hinterlegen Sie hier einen oder mehrere Ticketstatus. Bei Eingabe werden dann nur Status dieser Typen betrachtet. Werden keine Ticketstatus ausgewählt, gibt es keine Einschränkung und im Formularfeld werden alle Verwendungsstatus berücksichtigt.	new	open	pending reminder	pending auto	closed	...
Ticket-Types	Enter one or more ticket types here. Only tickets of the corresponding ticket types can be selected in the form field. If no ticket types are selected, there is no restriction and tickets of all types can be selected in the form field.	Unclassified	Incident	Service Request	...
Link Type	Defines which link is created between the current object (ticket) and the target ticket. The target ticket is <link type> of the source ticket. Without specification, no link will be established. **Important**: The dynamic field must be stored in the SysConfig key "*Ticket::EventModulePost###900-AddTicketReferenceLink*" so that links to other tickets can be created. *Fig.: Depositing a dynamic field of the type TicketReference in the event handler*	Parent	Child	Normal

OrganisationReference

Dynamic fields of the type "OrganisationReference" refer to organisations. They create selection fields for selecting one or more organisations. The input behaviour corresponds to that of a "Select" field. In the configuration, you can define, among other things, which organisations are available to the user for selection and with which organisation the field is to be preset (multiple selection possible). In addition, the objects available for selection depend on the authorisations of the logged-in user.

Fig.: A dynamic field of the type "OrganisationReference" in a ticket template

Configuration options (selection)

Option	Description
Count Min	Defines the minimum number of organisations that must be selected. The field becomes mandatory if the value is >0.
Count Max	Sets the maximum number of organisations that can be selected.
Default Value	Specifies the organisation with which the dynamic field is to be preset. Specify the ID[1] of an organisation. The pre-selection made here can be changed by the user by selecting other organisations. If no default value is defined, the field is displayed without preassignment (empty).

Option	Description
Organisation	Restriction of the organisations available for selection. Only the organisations specified here are available for selection in the field. To do this, click (several times if necessary) on "Add Item" and enter the ID[1] of an organisation in each item. If nothing is specified here, all organisations are available for selection.

[1] You can find the ID in the footer of your browser if you move the mouse pointer over an organisation in the overview of organisations.

ContactReference

Dynamic fields of the type "ContactReference" create selection fields for selecting one or more contacts on the ticket. It is thus possible to add further contacts to a ticket. The input behaviour corresponds to that of a "Select" field.

In the field configuration, you can restrict the selection of contacts to selected organisations and specific user groups (agents, customers). Only the contacts that fulfil the criteria defined here are then available for selection in the interfaces. The contacts available for selection also depend on the authorisations of the logged-in user.

Fig.: A dynamic field of the type "ContactReference" in a ticket template

Using KIX placeholders, dynamic fields of the type "ContactReference" can also be set as additional recipients in ticket notifications. The ticket notification can thus be sent variably to the respective contacts set in the ticket. The email address of the contact is taken from the placeholder so that you can use the parameters _Short, _Value and _HTML:

- <KIX_TICKET_DynamicField_myDFContactReference**_Value**> : "Last Name, First Name" <email>

- <KIX_TICKET_DynamicField_myDFContactReference_**Short**> : email
- <KIX_TICKET_DynamicField_myDFContactReference_**HTML**> : "Last Name, First Name" <email> (as "mailto-HREF")

A prerequisite for correct use is that a <u>valid and syntactically correct email address</u> is stored in the dynamic field in the ticket.

Configuration options (selection)

Option	Beschreibung
Count Min	Defines the minimum number of contacts that must be selected. The field becomes mandatory if the value is >0.
Count Max	Sets the maximum number of contacts that can be selected.
Item Separator	Separator between the individual contacts if several contacts can be selected in the field (usually comma or semicolon). ⚠ **Important!** The separator must be a **comma** if the field is referenced in the additional recipients of ticket notifications (see page 184) .
Default Value	Specifies the contact with which the dynamic field is to be preset. Specify the ID[2] of a contact. The pre-selection made here can be changed by the user by selecting other contacts. If no default value is defined, the field is displayed without preassignment (empty). **Note:** If the default value is not compatible with the organisation or context usage set in the configuration, the respective contact is not set.
Organisations	Restriction of the contacts available for selection. Only the contacts of the organisations selected here are available for selection. To do this, click (several times if necessary) on "Add Item" and enter the ID[2] of an organisation in each item. If nothing is specified here, all contacts are available for selection (possibly depending on the selected context of use).

Option	Beschreibung
Usage Context	Restricts the selection based on the context of use. To do this, click on "Add Item". One context of use can be selected per item. • Agent: Only users who have access to the Agent Portal can be selected. • Customer: Only users who have access to the Self Service Portal can be selected. The combination of the usage context agent and customer is possible. If nothing is defined, there is no restriction to users, so that contacts (without access to the Agent or Self Service Portal) can also be selected.

[2] You can find the ID in the footer of your browser if you move the mouse pointer over a contact in the overview of contacts.

FAQArticleReference

Dynamic fields of the type "FAQArticleReference" create selection fields for selecting one or more FAQ entries. They are used to create simple references to FAQ entries in order to initialise templates and actions with them.

A link is created between the ticket and the referenced FAQ entries, which is particularly helpful for application scenarios such as maintenance planning or in connection with the Field Agent App.

In this way, you can store FAQ entries (manufacturer's notes, product data sheets, etc.) in templates for maintenance plans, for example, so that they are automatically available on the ticket when it is created.

The input and selection behaviour corresponds to that of a select field. When used, the FAQ entries available for selection depend on the authorisations of the logged-in user and the field configuration (e.g. limiting the FAQ entries to selected categories and languages).

Configuration options (selection)

Option	Description
Count (min.)	Defines the minimum number of FAQ entries that must be selected. The field becomes mandatory if the value is >0.
Count (max.)	Defines the maximum number of FAQ entries that can be selected.

Option	Description
Item Separator	Separator between the individual FAQ entries, if more than one can be selected in the field (usually comma or semicolon).
Default Value	Specifies which FAQ article the dynamic field should be pre-populated with. Enter the ID[3] of the FAQ article. The pre-selection made here can be changed by the user by selecting other FAQ articles. If no default value is defined, the field is displayed without a default value (empty).
FAQ Categories	Restrict the FAQ entries available for selection to FAQ categories. In the dynamic field, only those FAQ entries are displayed that are assigned to the FAQ categories specified here. If necessary, click on "Add Item" several times and select the corresponding FAQ categories.
FAQ Languages	Restrict the FAQ entries available for selection to the languages specified here. If necessary, click on "Add Item" several times and select the corresponding languages.
FAQ Customer Visible	Restricts the FAQ entries available for selection in the field based on the set visibility in the Self Service Portal. • **Only visible in the Self Service Portal:** Only FAQ entries with the "Show in customer portal" option activated are available for selection. • **Only not visible in the Self Service Portal**: Only FAQ entries with the "Show in customer portal" option deactivated are available for selection. • **Both**: The FAQ entries visible and not visible in the Self Service Portal are available for selection.
Link Type	Select the link type with which the object type and FAQ entry/entries are to be linked.

[3] You can find the ID in the footer of your browser if you move the mouse pointer over a contact in the overview of contacts.

12.2.3 Initial Dynamic Fields

FKIX is initially delivered with the dynamic fields used by the system. You can adjust the configuration of these fields if necessary. However, do not change the field labels. Otherwise the fields cannot be correctly referenced and used by the system.

Note the object type when integrating dynamic fields! The object type defines the context in which the dynamic field can be used. Dynamic fields of the "Ticket" object type can only be integrated into ticket-relevant interfaces, but not, for example, in FAQs or organisations.

The following overview lists the dynamic fields initially delivered with KIX Start. Additional initial dynamic fields are delivered with KIX Pro.

The following dynamic fields are delivered with KIX:

Field name	Field type	Object type	Description	Configuration options
AffectedAsset	AssetReference	Ticket	Is required for the selection of affected assets in the ticket. The configuration defines the selection options in the "Affected asset" field.	• Limit the number of selectable assets (Count Max); Maximum: 15 • Limitation of the selectable assets to certain usage states (deployment state) • Limitation of the selectable assets to certain asset classes (ITSMConfigItemClass)
MobileProcessingState	Selection	Ticket	Is required for the Field Agent App. Contains the processing status of the ticket in the Field Agent App (accepted | rejected | finished | etc.)	• Change of names • Addition and removal of (un) required processing status

Field name	Field type	Object type	Description	Configuration options
PlanBegin/ PlanEnd	Date/Time	Ticket	Used to record planned times in the ticket. The configuration defines the preset value in the fields and defines the possible calendar period	• Limitation of the calendar • Adjustment of the automatically set times
RelatedAssets	AssetReference	FAQ	Used for selecting assets in FAQ. The configuration defines the selection options in the "Related Asset" field in the FAQ article.	• Limit the number of selectable assets (Count Max); • Limitation of the selectable assets to certain usage states (deployment state) • Limitation of the selectable assets to certain asset classes (ITSMConfigItemClass)

Field name	Field type	Object type	Description	Configuration options
Source	Text	Contact	Is used to provide individual contact entries from LDAP/AD directory services. The field can be specified in the UserSyncMap in order to include contact attributes from the LDAP/AD. If the field contains a value, this is displayed in the contact detail view. The field thus offers the possibility, for example. • to store in a contact entry from which AD the contact originates • to provide the identifiers for contact data of other systems in KIX as well.	not mandatory

Field name	Field type	Object type	Description	Configuration options
Typ	Selection	Organisation	Is used to provide organisational data and is included in the interfaces for creating/editing organisations. If the field contains a value, this is displayed in the detail view of the organisation. The field can be specified in the UserSyncMap, for example, in order to record organisational data from the LDAP/AD. So it is possible • to deposit at an organisation from which AD the organisation originates • to provide the identifiers for organisational data of other systems in KIX.	• Define the number of selectable entries • Define the selectable entries

System Monitoring Connection

The following dynamic fields of the field type "Text" and object type "Ticket" exist for the connection to system monitoring systems (e.g. Nagios):

- AcknowledgeName
- SysMonXAlias
- SysMonXAddress
- SysMonXHost
- SysMonXService
- SysMonXState

For instructions on how to use it, see: .

12.3 Logs

KIX creates a log for all important system messages and saves this in so-called log files (log files). Among other things, this allows the causes of problems to be discovered. The log files are saved as text files in the menu *System > Logs* and contain the data from the KIX system log. The overview contains the last 50 log entries, which corresponds approximately to a period of 2 days.

If required, you can download the log files or open them for viewing in a new KIX tab. To download a log file, click on it in the table. Afterwards, the browser functions for opening and saving are available. To view the log file directly in KIX, click on the eye symbol belonging to the log file in the column "View". The selected log file will be opened in a new KIX tab. You can filter the display of the logfile.

Content on this page:

- Refreshing Log Files (see page 315)
- Debug level setting for postmaster (see page 316)

Option	Description	Possible values
Log Level	Limits the display to the selected categories.	Debug | Error | Info | Notiz
Tail Lines	Limitation of the number of lines displayed in the log. The last n lines are displayed. The number of lines always refers to the selected log level, e.g. the last 10 error lines.	Number of lines Default: 100 Unlimited number of lines: 0 or empty
Refresh Interval (sec)	Determines every how many seconds the display should be completed.	Number in seconds Default: 10 Deactivation Refresh: 0 or empty
Wrap Lines	Switching line breaks on and off	yes | no
Filter	Filtering on specific entries (full text search). RegEx are supported. The filter refers exclusively to the current display in the log.	e.g. "Auth" or "System" or "Nov 24"

Fig.: A filtered log file

12.3.1 Refreshing Log Files

You are able to clear the log files of your KIX environment or have them explicitly rotated. This enables you to free up storage space and obtain log files of a manageable size.

To clear log files:

Navigate to *System > Console* and execute console command "*Maint::Log::Clear*". This clears all log files of the current KIX environment.

To rotate log files:

Navigate to *System > SysConfig.* You can define the rotation behaviour in the following SysConfig keys:

Key	Possible Values	Description
LogModule::LogFile::Rotate:	never, hourly, daily, weekly, monthly Default: onthly	Defines the interval at which the log files should be rotated.

Key	Possible Values	Description
LogModule::LogFile::Keep	Numerical entry Default: 12	Defines the number of old files that entry should be retained. Only applies if the value Default: 12 "File" (Kernel::System::Log::File) has been selected in the SysConfig key "LogModule". "12" by way of example: The 12 most recent log files are not deleted. "0" by way of example: No files are deleted.
LogModule::LogFile::Max Size	10M, 5K, 3G oder 10m, 5k, 3g	Max. size indicated in bytes. The unit can be specified in either uppercase or lowercase letters, but must come directly after the number (no blank)! If the files are rotated by size, all log files will be shifted by 1 and the current file suffixed with "_1"

> ⓘ **Please note**
>
> The changes do not affect Daemon debug logs as they have their own setting (key "*Daemon::Log::DaysToKeep*").

12.3.2 Debug level setting for postmaster

SysConfig key	PostMaster::MailAccountFetch::Debug

In the SysConfig key "*PostMaster::MailAccountFetch::Debug*" you can set the debug level for Postmaster incrementally. The setting can be set independently of other or global debug options so that you get targeted debug information for mail fetching in the log. The debug level is observed for postmasters with mail retrieval and email filtering mechanisms (Email Filter, SysMonX Filter).

4 levels can be set:

Level	Description
0	Off
1	Info (active) Activates the debug output for MailAccount and PostMaster (lowest level). The debug information from the IMAP/POP3 connections is output. This includes basic information about PostMaster FollowUp, Reject, DestQueue and NewTicket as well as some PostMaster filter information.
2	On (general) Extends the debug outputs with further PostMaster information, especially PostMaster filters. They depend on the respective PostMaster filter.
3	On (with Get Header parameters) Extends the debug output with further PostMaster information, such as "PostmasterX header".

The configuration only applies to the MailAccountFetch and is in 2nd place. This means: if a debug (via console command) is already active, then this is preferred.

> ⓘ **Info**
>
> Debugging is dependent on the set Minimum Log Level. Therefore, set the value "debug" in the SysConfig key "*MinimumLogLevel*".

12.4   SysConfig

The menu *System > SysConfig* contains the KIX system configuration (hereinafter referred to as SysConfig). This is where you can make changes to the basic KIX settings and configure the program interface. As a result, you are able to tailor KIX precisely to suit your company's specific needs. All changes to the system configuration are made by changing the values of a system key. Virtually all configuration parameters of the central framework and additional modules can be changed. For example, you can

- save the default team (see page 625) for receiving new tickets, unless specified otherwise in the email accounts.
- save the default email address (see page 224) from which all outbound emails are sent, provided that no other email address has been specified in the team settings.
- amend the user interface (see page 327) (GUI) by e.g. integrating dynamic fields, changing table columns, changing the default values in fields
- customise the display in the sidebar widgets (see page 412) and much more.

> ⬣ **Attention!**
>
> Changes made to the system configuration always constitute a fundamental intervention into the central KIX system. We would like to highlight that making careless or ill-considered changes to the configuration files can cause serious system errors and even complete system failure. Please do not hesitate to contact us if you require any assistance. We would be glad to tailor KIX to your needs or help you to configure the system.

> ⚠ **Important!**
>
> Make changes to the system configuration at a time when the system is not being used. This will prevent potential conflicts in the system.

> ⓘ **Please note**
>
> - Some of the terms used in the KIX frontend are aimed specifically at Technical Service. However, ITSM-compliant terms are used internally within KIX. The following terms are affected:
>   - Team => Queue
>   - Asset(s) => CI/Config Item(s)
>   - Asset database => CMDB
>   - Organization/Organisation => Customer
> - The SysConfig keys for your own organisation use the English notation with "z" (e.g. OrganizationAddress).
>
> If you are using our KIX Cloud and need to access a software backup, KIX Cloud users can contact our Support team as provided for in the support contract. If you are hosting the system yourself, you will need to access your own backup.

12.4.1 Table in Dashboard

The table in the Dashboard lists the configuration keys of KIX based on a search term. In search field, enter name to search for specific keys. You can search through the properties in the columns "Name", "Context", and "Metadata". If you are only searching for part of a name, it must be placed in * (wildcard search, e.g. "ticket-new*"). To display all keys, enter only asterisk in search field. It may take a moment for all keys to load. You can also use the search functions and filters within the table to search for a specific key.

The configuration keys can be marked as follows:

- Modified keys are marked with a check mark in the "Modified" column.
- To find all modified keys, enter "**modified**" in the search field. To find all keys that have not been modified, type "**!modified**".
- Keys set to "invalid" are displayed in light gray.
- Keys marked with a lock icon are locked for editing in SysConfig.
- Keys highlighted in orange are security-relevant keys set to "valid".

Click on a key to open it for editing.

Fig.: Table in Dashboard Contains the KIX Configuration Keys

Column	Description
Name	Name of the key. Keys marked with a lock icon are already preconfigured and locked for editing in SysConfig.
Modified	Indicates whether a key has been modified. The system checks whether the key differs from the original configuration.

Column	Description
Context	Keys are that relevant for configuring the user interfaces contain the entry "kix18-web-frontend".
Metadata	Additional information on the key. **agent-portal-configuration**: Key for the settings of the agent portal **Chart**: Key for the content in a chart widget (e.g. displayed state in widget "Ticket by state") **ChartWidget:** Key for a chart widget (e.g. widget "Tickets by state") **Form:** Key for a form (e.g. ticket creation screen) **FormField**: Key for a form field (e.g. field "Organisation" in ticket)
Access Level	Security level and access permission for a key. **public:** Unrestricted access **confidential:** security-critical key; requires the role "System Admin" **internal:** Key for internal technical use Keys of the access level "confidential" are highlighted in orange font if they are "valid". They require the "Superuser" and "System Admin" roles for editing. A user with the "Agent User" role - without "Superuser" and without "System Admin" - can therefore not read out any configurations for authentication and connection of third-party systems from SysConfig. Keys generated manually or by development, such as "Authentication##[...]", are assigned to the "confidential" access level.
Validity	Indicates whether a key has been set to "valid" or "invalid (-temporarily)". Invalid keys are shown in light gray.
Value	Short form of JSON string used to define the key.
Changed at:	Time when the key was last changed.
Changed by:	User who last changed the key.

12.4.2 Editing a SysConfig Key

You can change the configuration keys in order to configure KIX to meet the needs of your industry and company. Although KIX offers you the option of resetting keys back to their original value, please take **great care** when making any changes because making changes constitutes a direct intervention in the system!

To make it easier to edit the value of a key, you can copy it to a JSON editor (e.g. https://www.jsonformatter.io), edit it there, and then copy it back in minified form, i.e. without blanks or line breaks.

Changes made to the system configuration are retained after updating the system. Despite this, it may be necessary to reconfigure individual keys. To ensure the system remains secure, you cannot add any of your own keys.

> ⚠ **Attention!**
>
> Changes to the configuration keys can lead to serious errors and even system failure!
> You should therefore have sound knowledge of the system and in dealing with source code - especially with JSON - when making changes. Feel free to contact our support if you need help or if you want us to customize the user interface.

To configure a key, proceed as follows:

1. In explorer, navigate to *System > SysConfig*.
2. In search field, enter character string of required key. You can use wildcards, e.g. ticket-new-form*. In the content area, the table lists all the keys that were found as the result of your search.
3. If necessary, filter table further.
4. In table, click key to be edited. A form dialog for configuring the key opens.
5. If necessary, activate or deactivate key by specifying its validity.
6. Change value of key as required. If you are using a JSON editor, make sure to delete all blanks and line breaks before saving.
7. Click "Save" to save your changes. The key has now been changed.

> ⚠ **Important!**
>
> **After changing the user interface, e.g. after integrating a dynamic field, you will need to reload the frontend. To do so, click "Reload Frontend" button.**

To configure several keys at the same time, proceed as follows:

1. In explorer, navigate to *System > SysConfig*.
2. In search field, enter character string of required key. You can use wildcards, e.g. sendmail*. In the content area, the table lists all the keys that were found as the result of your search (in this example this is all SysConfig keys that are relevant for setting up email accounts).

3. If necessary, filter table further.

4. Select all keys to be edited by checking them off. Click "Edit" button. A form dialog for configuring the keys opens. You can use the navigation buttons to switch between the selected keys ‹ • • ● • › .

5. If necessary, activate or deactivate key by specifying its validity.

6. Change value of key as required. If you are using a JSON editor, make sure to delete all blanks and line breaks before saving.

7. Click "Save" to save your changes. The key has now been changed.

To reset a key to the original value (default state), proceed as follows:

1. In explorer, navigate to *System > SysConfig*.

2. In search field, enter character string of key. You can use wildcards, e.g. "ticket-new-form*". In the content area, the table lists all the keys that were found as the result of your search.

3. If necessary, filter table further. Modified keys are shown with a check mark in the column "Modified".

4. In table, click key to be reset. A form dialog for resetting the key opens.

5. Click "Reset to Default". The default value is applied to the field "Value".

6. Click "Save" to apply your changes. The key has now been reset to the default state.

> ✅ **Tip**
>
> After changing the GUI configuration (e.g. after integrating dynamic fields), you can copy correctly functioning keys to a text document and save this document as a local backup copy. This will enable you to access the document easily if required, negating the need to reconfigure the entire key.

The form dialog contains the following input fields, among others:

Fields (selection)	Description
Name	Name of key (cannot be changed)
Description	Explanation of key (cannot be changed)
Validity	- valid: The key will be used - invalid/invalid-temporarily: The key will (temporarily) not be used.
Value	The JSON string defines the key. Edit string to change configuration of key. Make sure to use the correct syntax and spelling for the attributes, and if required delete blanks and line breaks. Please do not hesitate to contact us if you require any assistance.

Fields (selection)	Description
Default Value	The original value of the key (cannot be changed). You can copy it and paste it into the field "Value" to reset the key.
"Reset to Default"	Button for resetting the value to the default value. Also click "Save" to save your change.

13 System settings

In the system settings part you will find a selection of different use cases for working with KIX.

> ⬢ **Attention**
>
> This manual contains configuration **examples** to explain the respective topics. If you transfer these examples to your system, you do so at your on own responsibility. We do not assume any liability for this.

- Use URL of the browser (see page 605)
- Display of support information (see page 607)

13.1   Configuration of User Interface (GUI)

KIX allows to customize the configuration of the user interface (GUI configuration). It is therefore possible

- add dynamic fields to input masks
- display the values of dynamic fields in overviews and widgets
- remove and insert columns in overview tables
- remove the widgets contained in the sidebar or integrate your own widgets into the sidebar
- configure the content to be displayed in widgets
- etc.

The possibility of GUI configuration is therefore a very powerful gadget to build a KIX perfectly adapted to the needs of your industry and especially your company.

Fig.: A Dynamic Field Integrated into the Ticket Creation Screen

> 🔺 **Attention!**
>
> Making changes to the GUI configuration can cause **serious errors** in the KIX system. For this reason, you must have sound knowledge of the system and JSON in order to work on the GUI configuration.
> Please do not hesitate to contact our Support team if you need help or would like us to modify the user interface.

Almost all elements in KIX (widgets, lanes, charts, tables, forms, form fields, etc.) can be customised. The configuration is done in the SysConfig key of the respective element (menu *System > SysConfig*). Each element defines itself via the JSON string in its key ("Value" field). All properties and parameters of the element are defined in the JSON string. For better editing, you can copy the string into an external JSON editor (e.g. https://www.jsonformatter.io) and copy it back into the key after editing.

Tip: KIX Pro users can comfortably customise the user interface in the menu *System > GUI Configuration > Agent Portal*.

For example, to include a dynamic field in the "New ticket" dialogue, open the configuration key of this dialogue (ticket-new-form-group-data) and add the name and parameters of the dynamic field to the value. After reloading the frontend, the dynamic field will be available in the ticket creation screen.

Fig.: The value of the SysConfig key "ticket-new-form-group-data" defines the structure and content of the dialog "new ticket"

If you make any mistakes while editing the JSON string, you can restore the default configuration of the key. To do this, click "Reset to default". This writes the initial JSON string contained in the "Default value" field back into the "Value" field. However, this function **does not release you from your duty of care** when configuring the user interface!

> ✅ **Tip**
>
> After changing the GUI configuration (e.g. after integrating dynamic fields), you can copy correctly functioning keys to a text document and save this document as a local backup copy. This will enable you to access the document easily if required, negating the need to reconfigure the entire key.

13.1.1 Personalising the Home Dashboard

The "Customise Dashboard" button allows agents to personalise their home dashboard. Among other things, they can determine the widgets to be displayed and adapt the display in the tables to their personal needs.

As an administrator, you can create the basis for these selections. You can configure which widgets with which content are available to all agents. And you can define the content to be displayed in the dashboard

tables. The agent can then define for himself which of the widgets his personal dashboard should contain and which of the table columns he needs with which properties.

You can define the widgets available in the dashboard in the SysConfig key "home" by integrating additional widgets into the dashboard (see page 485) or removing widgets that are included. The contents to be displayed in the widgets are defined in the SysConfig key of the respective widget.

You define the basic configuration of a dashboard table (see page 497) in the key of the respective table. There you can, for example, insert further table columns, remove table columns that are not required or define the data to be displayed by modifying the search in the table (see page 480) .

You can reset the personal widget settings of a user. To do this, select the relevant user(s) under *User Management > Users* and click on the "Reset User Widgets" button. You can also reset the widget settings in the detailed view of a user.

Fig.: Reset User Widgets

13.1.2 Basics of GUI Configuration

13.1.2.1 Configuration Hierarchy

All elements of the user interface are arranged hierarchically - analogous to the "matryoshka principle". The hierarchy must be adhered to when configuring the user interface. The hierarchy is achieved by nesting the individual SysConfig keys. The form fields in the form dialogs ("New Ticket", "Edit Ticket", etc.) are always child elements of groups. Groups are always child elements of pages. Pages are in turn child elements of forms etc. Therefore, a dynamic field must be integrated into the group "ticket-[new|edit]-form-group-data". Every page must always have a group in order to be able to accommodate form fields. At the same time, the widgets, tables and lanes in the frontend are always child elements of a ConfiguredWidget. The ConfiguredWidget, in turn, is a child element of the context (sidebar, lane, explorer, etc.). You can find out which hierarchy level a SysConfig key belongs to in the "Metadata" column in the overview of the SysConfig keys.

Structure of the hierarchy

- context
  - widget
    - table widget
      - table configuration
        - table column configuration
    - object information card widget
    - linked objects widget
    - help widget
    - slider widget
    - ticket overview widget
    - tab widget
    - asset overview widget
  - main menue
  - formular
    - site
      - group

- field
  - ..field

Fig.: Configuration Hierarchy in the form

13.1.2.2   Context

The linchpin for the pages in the interface are always the contexts. A context can be, for example, a dashboard, a detail page or a dialog. This context is always based on a configuration. Part of this configuration are various lists with widgets, which are then displayed accordingly in the interface. The lists are divided, for example, into:

- sidebars
- explorer
- lanes
- content
- others
- .....

Within these lists are the respective configurations for the sidebar, the explorer, the lanes, the dashboard and so on. If you want to integrate an additional widget into the sidebar, this is done in the respective context (e.g. ticket details) in the list "sidebars". If you want to integrate an additional table in the ticket dashboard, this is done in the context "ticket dashboard" in the list "content", etc.

```
{
    "id": "ticket-details",
    "name": "Ticket Details",
    "type": "Context",
    "contextId": "ticket-details",
    "sidebars": [⊂▭⊃],
    "explorer": [],
    "lanes": [⊂▭⊃],
    "content": [⊂▭⊃],
    "generalActions": [⊂▭⊃],
    "actions": [⊂▭⊃],
    "overlays": [⊂▭⊃],
    "others": [⊂▭⊃],
    "dialogs": [],
    "customizable": false
```

13.1.2.3   ConfiguredWidget

An object "ConfiguredWidget" is expected in the lists. This forms the frame for the concrete widget, which is given by `configuration` .

A Configured Widget has the following relevant properties:

- **instanceId**: Unique string (context-wide), which is needed to identify the concrete widget instance.
- **configurationId**: Reference ID to a concrete widget configuration.
- **configuration**: If no reference is used, then the configuration can be specified directly here.
- **permissions**: permissions necessary for the widget to be displayed in the interface (e.g. READ on / tickets)
- **size:** display size of the widget (large = large widget, small = small widget).

```
{
    "id": "ticket-details",
    "name": "Ticket Details",
    "type": "Context",
    "contextId": "ticket-details",
    "sidebars": [🖮],
    "explorer": [],
    "lanes": [🖮],
    "content": [
        {
            "instanceId": "ticket-details-article-list
                -widget",
            "configurationId": "ticket-details-article
                -list-widget",
            "configuration": null,
            "permissions": [],
            "size": "large"
        }
    ],
    "generalActions": [🖮],
    "actions": [🖮],
    "overlays": [🖮],
    "others": [🖮],
    "dialogs": [],
    "customizable": false
```

13.1.2.4 WidgetConfiguration

This configuration in turn expects a so-called WidgetConfiguration. This configuration is used to configure the widget in its concrete form:

- widgetId - id of the widget which should be displayed (see examples)
- title - the title text of the widget
- icon - the icon to be displayed in the header
- actions - actions to be displayed in the widget header
- minimized - minimized true/false
- minimizable - true/false
- subConfigurationDefinition - reference to a subconfiguration
- configuration - the concrete configuration of the widget*.
- (contextDependent) - depends on the context
- (contextObjectDependent) - depends on the object of the context
- (formDependent) - dependent on a form
- (formDependencyProperties) - form values on which the widget depends

```json
{
  "id": "ticket-details-article-list-widget",
  "name": "Article List Widget",
  "type": "Widget",
  "widgetId": "table-widget",
  "title": "Translatable#Article Overview",
  "actions": [],
  "subConfigurationDefinition": {
    "configurationId": "ticket-details-article-list
      -table-config",
    "configurationType": "TableWidget"
  },
  "configuration": null,
  "minimized": false,
  "minimizable": true,
  "icon": null,
  "contextDependent": true,
  "contextObjectDependent": false,
  "formDependent": false,
  "formDependencyProperties": []
}
```

The widget has its own configuration (subConfigurationDefinition). This contains the individual configurations of the individual widgets. For example, the table widget contains the specific table configuration at this point:

- **Context** (e.g. Sidebar, Lane, Explorer, etc.)
  - **Content** (list of content widgets)
    - **ConfiguredWidget** (widget shell)
      - **table widget** (e.g. ticket table, recommended FAQ, etc.)
        - **TableConfiguration** (concrete widget content)
          - Configuration of table columns
    - **object-information-card-widget** (e.g. Contact Information widget, Ticket Information lane, Help widget, etc.)
      - configuration of avatar and rows
    - **object-dialog-form-widget** (e.g. dialogs New Ticket, Edit Ticket, etc.)
      - **Form**
        - **Page**
          - **Group**
            - **Field**
            - **Field**
              - **[Field] ...**

```
{
  "id": "ticket-details-article-list-table-config",
  "name": "Article Table",
  "type": "TableWidget",
  "objectType": "Article",
  "tableConfiguration": {
    "id": "ticket-details-article-list-table",
    "name": "Article Table",
    "type": "Table",
    "objectType": "Article",
    "loadingOptions": null,
    "displayLimit": 10,
    "tableColumns": null,
    "tableColumnConfigurations": null,
    "enableSelection": null,
    "toggle": true,
    "toggleOptions": {⌨},
    "headerHeight": 2.25,
    "rowHeight": 1.75,
    "emptyResultHint": "Translatable#0 data sets found.",
    "fixedFirstColumn": false,
    "additionalTableObjectsHandler": [],
    "intersection": true,
    "searchId": null
  },
  "headerComponents": [
    "article-attachment-count"
  ],
  "showFilter": true,
  "shortTable": false,
  "predefinedTableFilters": [],
  "cache": false,
  "resetFilterOnReload": true
}
```

✅ **Tipp**

You can also find a technical description for configuring the KIX frontend at: https://github.com/kix-service-software/kix-frontend/tree/master/doc.

You can view the HTML view of this description at: https://github.com/kix-service-software/kix-frontend/blob/master/doc/doc.tar.gz herunterladen (Klick auf Download-Button).

13.1.2.5 The configuration of widgets

A widget consists of the widget itself (①)
and the content to be displayed in it (②).
Both have their own configuration block
(" `configuration":{...}`). The
configuration block of the widget contains the
configuration block of the widget content.

Contents on this page:

Fig.: Widget and widget content using the object-information-card-widget "Contact information" as an example

In the configuration block of the widget are defined among other things:

- the properties and behavior of the widget
- widget title and widget type
- Configuration of the widget content.

In the configuration block of the widget content are defined among others

- the content to be displayed in the widget (placeholders, table definitions, dynamic fields, etc.)
- Conditions for displaying the content
- links and routing information
- etc.

Types of Widgets

The most commonly used widgets in the GUI configuration are object-information-card-widget and table-widget.

The object-information-card-widget is a card widget with object information displayed by rows and columns. The object information configured in the widget can be of different types (formatted text, icon, links, dynamic fields). Object-information-card-widgets are, for example, the Contact Information widget or the ticket information in the ticket details (as part of the lane). For configuration information about this widget, see "The object-information-card-widget (see page 355) ".

The table-widget is a table widget for displaying individual object tables. The content displayed in the table can be designed dynamically. Table-widgets are, for example, the ticket and item overview in the home dashboard or the widgets "Suggested FAQ", "Tickets to Assets" and "Tickets to Contact". For configuration information about this widget, see "The table widget (see page 393) " .

Basic configuration of a widget

The following code example shows the basic structure in a simple object-information-card-widget.

```
{
    //creates an instance of the widget
    "instanceId": "name-of-widget-instance"
    //Widget configuration
    "configuration":{
        "id": "ID od widget",
        "name": "name-of-widget",
        "type": "Widget | TableWidget",
        "widgetId": "table-widget | object-information-card-widget",
        "title": "Translatable#translatable widget title",
        "actions": [],
        "subConfigurationDefinition": null,
        // Configuration of the widget content
        "configuration": {
            //Configuration of the avatar
```

```
            "avatar": {string | ObjectIcon},
            //Defines which contents are displayed per line
            "rows": [
                {... see the widget content...}
            ],
        }
        //properties/behavior of the widget
        "minimized": false,
        "minimizable": false,
        //Icon in the title bar
        "icon": "kix-icon-man-house",
        "contextDependent": false
    },
    "permissions": [],
    "size": "large"
},
```

The widget content

The widget content is based on the values of search results. Therefore, you can freely configure the content to be displayed.

In object-information-card widgets, depending on the object attributes, almost all types of information can be displayed (regardless of the object type). You can find an example in the chapter Configuring an object-information-card-widget (see page 473) .

- Dynamic field values
- external links (URL, e-mail, phone, web CRM)
- internal links and quick clicks (tickets, contacts, etc.)
- image files (icons, avatars)
- placeholders
- dynamic and static parameters
- conditional contents
- etc.

The display in table widgets can be configured so that the table content changes dynamically. For this purpose, you can control the search mechanisms and use filters in the configuration. You can find an example in chapter Configuration of table widget "Suggested FAQ" (see page 480) .

Simplified basic structure of the widget content in an object-information-card widget

```
{
    "avatar": {string | ObjectIcon},
        "rows": [
            {//content of a line
```

```
            "margin": true | false,
            "values": [
                {
                    "icon": string | ObjectIcon,
                    "text": string,
                    "linkSrc": string,
                    "routingConfiguration": RoutingConfiguration |
    DialogRoutingConfiguration,
                    "routingObjectId": string
                }
            ]
        }
        {//optional content of further lines
            "margin": true | false,
            "values": [...]
        }
    ]
}
```

General widget attributes

Attribute	Description	Example
id	Unique ID of the object in KIX	
name	Name of the object of widget	
type	Type of the object of widget	Context | Widget | TabWidget | HelpWidget
widgetID	Type of the widget	object-information-card-widget | table-widget
title	Widget title in the widget header; KIX placeholders can be used.	"Contact Information"
actions	Ticket actions to be available initially (optional)	"Edit", "Duplicate", "new Article", "CSV-Export" etc.
subConfigurationDefinition	optional sub-cononfiguration	

Attribute	Description	Example
configuration	content of widget	• Integration of additional widgets • Configuration of widget content (avatar/rows or table configuration)
minimized	Specifies whether the widget is displayed minimized (collapsed).	true | false
minimable	Specifies whether the widget can be minimized. If true, then it contains small arrow buttons for collapsing	true | false
icon	optional configuration of the icon which is displayed next to the widget title	• "kix-icon-man-house" | "fas fa-phone-volume" • Configuration block of an ObjectIcon
contextDependent	Controls the dependence on the context	true | false
contextObjectDependent	Controls the dependency on the object of the context This can be used, for example, in table widgets to control that the loading of the table contents is not done by the table but by the context.	true | false
formDependent	Controls the dependency on the form	true | false
formDependencyPropertys	Form values on which the widget depends	"DynamicFields.myDynamicFieldName"

Widgets implementation types

Widgets can be implemented in the sidebar in 2 ways:

Fixed implemented

These widgets are permanently implemented in the sidebar. They have no central configuration. Their configuration is specified directly in the widget and can therefore vary in each instance. If a general behavior of the widget is to be changed, it must be changed in all instances. However, this also makes it possible, for example, to specify a different help text in each help widget. However, permanently implemented widgets can be

- be shown or hidden via the configuration of the context.
- By copying and pasting their instance, they can also be permanently implemented in other sidebars. For example, to additionally display the "Notes" widget in the ticket details.

Widgets permanently implemented in the sidebar are for example:

- Notes (Home Dashboard)
- Time booking (KIX Pro)

Instance of a permanently implemented widget (here: widget "Time booking" in KIX Pro)

```
{
    "instanceId": "ticket-details-simple-time-accounting-widget",
    "configurationId": "ticket-details-simple-time-accounting-widget",
    "configuration": {
        "id": "ticket-details-simple-time-accounting-widget",
        "name": "Simple Time Accounting",
        "type": "Widget",
        "widgetId": "ticket-simple-time-accounting-widget",
        "title": "Translatable#Simple Time Accounting",
        "actions": [],
        "subConfigurationDefinition": null,
        "configuration": null,
        "minimized": false,
        "minimizable": true,
        "icon": "kix-icon-time",
        "contextDependent": false,
        "contextObjectDependent": false,
        "formDependent": false,
        "formDependencyProperties": []
    },
    "permissions": [],
     "size": "large"
},
```

Flexibly implemented

Flexibly implemented widgets have a central configuration in their own configuration key. They are implemented by reference to their configuration key. This allows them to be flexibly integrated not only in the sidebar, but in almost all interfaces.

Since these widgets have a central configuration, changes to the configuration are applied to all instances of this widget - regardless of the context in which the widget is included. They therefore have a greater configuration diversity than permanently implemented widgets.

The content to be displayed in the widget can be dynamically designed by using placeholders, filters and conditions. Since the content is based on search results and only the result values are displayed, almost all types of information - regardless of the object type - can be integrated. It is thus possible, for example:

- integrate an additional widget for displaying SLA information in the sidebar
- display a photo (avatar) of the contact person or customer, if a picture of the contact/customer is stored)
- to link the contact address for display in online map services
- display the values of dynamic fields - regardless of the object type
- to link to external URLs (e.g. to external web pages, to content in a web CMS)
- link to e-mail addresses and phone numbers to open them in an external program,
- quick clicks on internal links (e.g. create new asset, go to contact, etc.)
- display additional information, such as the number of assigned assets or open tickets,
- use KIX placeholders to display content dynamically,
- display conditional content, i.e. display content only when certain properties are present on the contact or customer,
- etc.

Flexible widgets include:

- Contact information
- Tickets to assets
- Suggested FAQ
- Tickets to Contact
- Open child tickets (KIX Pro)

Example of flexibly implemented widgets in the ticket details sidebar

```
{ ...
   //Sidebar in the ticket details
   "sidebars": [
   { //instance of the Contact Information widget
       "instanceId": "ticket-details-contact-card-widget",
       "configurationId": "ticket-details-contact-card-widget",
       "permissions": [],
```

```
            "size": "large"
        },
        { //instance of the widget suggested faq
            "instanceId": "ticket-details-suggested-faq-widget",
            "configurationId": "ticket-details-suggested-faq-widget",
            "permissions": [],
            "size": "large"
        },
        { //instance of the widget affected assets
            "instanceId": "ticket-details-affected-asset-tickets",
            "configurationId": "ticket-details-affected-asset-tickets",
            "permissions": [],
            "size": "large"
        },
        { //instance of the widgetTickets to contact
            "instanceId": "ticket-details-contact-tickets",
            "configurationId": "ticket-details-contact-tickets",
            "permissions": [],
            "size": "large"
        }
    ],
```

The chart widget

Configuration Key	home

Chart widgets can display statistics graphically. The display can be a bar chart, pie chart or line chart.

A report definition serves as the data basis for the chart widget. Their last created report is displayed as a statistic.The prerequisite for this is that CSV is defined as the output format in the report definition. In KIX Pro, JSON is also possible. Other output formats cannot be processed by the Chart Widget.

Content on this page:

- Stucture of a chart widget (see page 345)
- Configuration parameters of the chart widget (see page 347)
- Parameters of the chart configuration (see page 348)
- Widget configuration in KIX Pro (see page 350)
- Display your own report in the chart widget (see page 351)

The periodic creation of the reports and thus the updating of the statistics is done by the initial job "Periodic Reports (see page 170) ". The charts in the Home Dashboard are also generated in this way.

Fig.: The initial chart widgets in the Home Dashboard

The statistics are not displayed if the report cannot be evaluated.

Reasons for this can be:

- No report exists for the selected report definition.
- The permissions for the report are missing.
- The report data could not be evaluated.
- The report is not available in CSV format (or JSON).

Stucture of a chart widget

The initial chart widgets are embedded in the configuration of the Home Dashboard (SysConfig key "*Home*"). They are located within the configuration block " `content:[...]` ". Each chart widget consists of a widget frame and the widget configuration, which in turn contains the configuration of the chart widget together with information on the data basis (report definition) and the formatting of the chart (CSV mapping).

```
"lanes": [],
"content": [
        "instanceId": "home-dashboard-ticket-chart-widget-priorities",     Widget Frame
        "configurationId": null,
    "configuration": {
        "id": "home-dashboard-ticket-chart-widget-priorities",           Widget Configuration
        "name": "Priority Chart Widget",
        "type": "Widget",
        "widgetId": "report-chart-widget",
        "title": "Translatable#Number of open tickets by priority",
        "actions": [],
        "subConfigurationDefinition": null,
        "configuration": {
            "id": "home-dashboard-ticket-chart-widget-priorities-chart",     Chart Widget
            "name": "Translatable#Priority Chart",
            "configuration": {
                "id": "home-dashboard-ticket-chart-widget-priorities-config",
                "name": "Translatable#Priority Chart",
                "type": "Chart",
                "chartConfiguration": {
                    "type": "bar",                           Chart Configuration
                    "options": {
                        "legend": {
                            "display": false
                        },
                        "scales": {
                            "yAxes": [
                                {
                                    "ticks": {
                                        "beginAtZero": true,
                                        "maxTicksLimit": 6
                                    }
                                }
                            ]
                        }
                    }
                }
            },
            "reportDefinitionId": 4,
            "reportOutputFormat": "CSV",
            "formatConfiguration": {
                "datasetProperties": [                       CSV Mapping
                    "name"
                ],
                "labelProperty": "name",
                "valueProperties": [
                    "count"
                ],
                "textSeparator": "\"",
                "valueSeparator": ","
            },
            "useReportTitle": true,
            "type": "ChartWidget"
        },
        "minimized": false,
        "minimizable": true,
        "icon": null,
        "contextDependent": false,
        "contextObjectDependent": false,
        "formDependent": false,
        "formDependencyProperties": []
    },
    "permissions": [],
    "size": "small"
```

Fig.: Configuration of the chart widget "Number of open tickets by priority" in the configuration key of the Home Dashboard

Configuration parameters of the chart widget

The configuration of the chart widget requires the following parameters:

```
"content": [
    {
        "instanceId": "home-dashboard-ticket-chart-widget-priorities",
        "configurationId": null,
        "configuration": {
            "id": "home-dashboard-ticket-chart-widget-priorities",
            "name": "Priority Chart Widget",
            "type": "Widget",
            "widgetId": "report-chart-widget",
            "title": "Translatable#Number of open tickets by priority",
            "actions": [],
            "subConfigurationDefinition": null,
            "configuration": {
                "id": "home-dashboard-ticket-chart-widget-priorities-chart",
                "name": "Translatable#Priority Chart",
                "configuration": {⬜},
                "reportDefinitionId": 4,
                "reportOutputFormat": "CSV",
                "formatConfiguration": {
                    "datasetProperties": [
                        "name"
                    ],
                    "labelProperty": "name",
                    "valueProperties": [
                        "count"
                    ],
                    "textSeparator": "\"",
                    "valueSeparator": ","
                },
                "useReportTitle": true,
                "type": "ChartWidget"
            },
            "minimized": false,
            "minimizable": true,
            "icon": null,
            "contextDependent": false,
            "contextObjectDependent": false,
            "formDependent": false,
            "formDependencyProperties": []
        }
```

Fig.: Configuration parameters in the Chart Widget

Parameters	Description
reportDefinitionId	The ID of the report definition (data basis of the chart). • **KIX Start:** You can change the ID to refer to your own report. Enter the ID of the corresponding report definition. • **KIX Pro:** Changing the ID is not necessary to refer to another report. All reports readable by the Chart Widget can be selected in the dashboard personalization.
reportOutputFormat	The output format of the report to be used according to the report definition.
useReportTitle	Use the name of the report definition as widget title, otherwise title from widget configuration

Parameters	Description	
formatConfiguration	Specific configuration for the output format. The information requires detailed knowledge of the report definition (e.g. result of the report, which data is contained in which form) Following for CSV:	
	datasetProperties	Name of the columns in the CSV that are to be displayed in the chart as a dataset.
	labelProperty	Name of the column in the CSV that contains the label of the Y-axis.
	valueProperties	Designation of the columns containing the values of the datasets
	textSeparator	Separator, which is used for character strings in the CSV.
	valueSeparator	Separator for values in CSV

Info: The parameters (attributes) of the widget configuration can be found under: The configuration of widgets

Parameters of the chart configuration

The chart configuration defines the display of the chart. Listed below are the configuration parameters used initially. Further parameters can be found on Chart.js: https://www.chartjs.org/docs/latest/configuration/.

```
"configuration": {
    "id": "home-dashboard-ticket-chart-widget-priorities",
    "name": "Priority Chart Widget",
    "type": "Widget",
    "widgetId": "report-chart-widget",
    "title": "Translatable#Number of open tickets by priority",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
        "id": "home-dashboard-ticket-chart-widget-priorities-chart",
        "name": "Translatable#Priority Chart",
        "configuration": {
            "id": "home-dashboard-ticket-chart-widget-priorities-config",
            "name": "Translatable#Priority Chart",
            "type": "Chart",
            "chartConfiguration": {
                "type": "bar",
                "options": {
                    "legend": {
                        "display": false
                    },
                    "scales": {
                        "yAxes": [
                            {
                                "ticks": {
                                    "beginAtZero": true,
                                    "maxTicksLimit": 6
                                }
                            }
                        ]
                    }
                }
            }
        }
    },
    "reportDefinitionId": 4,
    "reportOutputFormat": "CSV",
    "formatConfiguration": {
        "datasetProperties": [
            "name"
```

Fig.: Parameters of the chart configuration

Parameters	Description	
type	Specifying the diagram type • bar: bar diagram • pie: pie diagram • line: line diagram • bubble: bubble diagram • polar: polar diagram • radar: radar diagram	
options	Contains options for displaying the legend in the widget and for scaling the X/Y axes. Further parameters are possible.	
	legend	Determine whether the legend should be displayed in the chart (display: true | false)
	scales	Information on the scaling of the X- and Y-axis

> ⓘ **Note**
>
> The colours used in the charts are based on the random principle and can currently not be adjusted.

Widget configuration in KIX Pro

Each agent can define the display and presentation of the chart widgets for himself via the personalization of the Home Dashboard. After clicking on "Advanced", both the configuration parameters for the widget and those for the chart are available.

General information on personalising the Home Dashboard can be found in the KIX Start User Manual in the chapter *Home Dashboard > Personalize your Home Dashboard*.
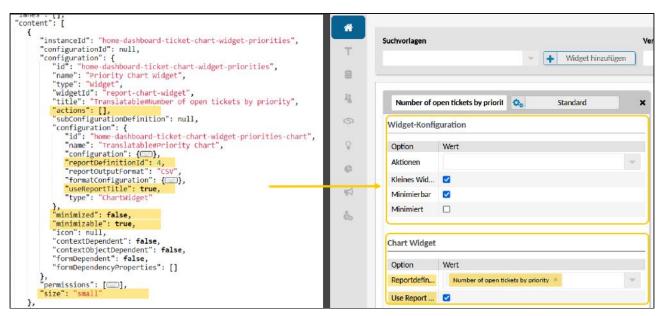
Fig.: Configuration parameters of the widget with selection of the report definition as data basis for the chart
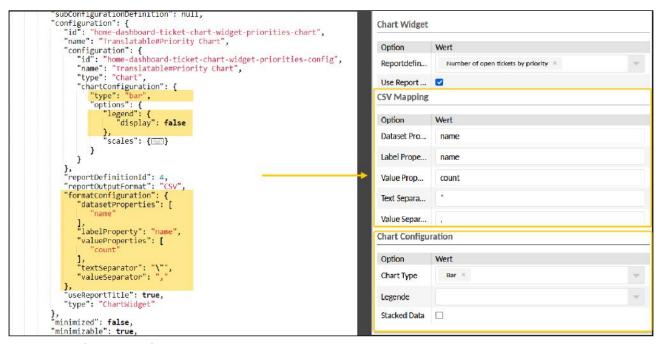
Fig.: The configuration of the chart display

Display your own report in the chart widget

You can create your own reports and make them available as statistics in a chart widget. To do this, configure a report definition (see page 710) with the output format CSV.

- If the report is to be created and updated automatically at regular intervals, activate the checkbox "Periodic creation" in the report definition.
  Initially, periodic reports are created by the "Periodic Reports" job. Alternatively, you can create a separate job for this purpose.
  If the checkbox is not activated, the report must be created and updated manually or by another event- or time-based job.
- Specify in the report definition under "Max. reports" how many reports are to be kept. This will automatically delete surplus, older reports.

Which report is displayed in which chart widget and how the graphical representation is done is defined in the configuration of the respective widget:

- **KIX Start:** The personalization of the Home Dashboard in KIX Start does not allow the selection of the reports to be displayed. To display custom reports, the configuration of one of the existing chart widgets must therefore be changed manually:

  1. Navigate to *KIX > System > SysConfig* and open the configuration key "*Home*".
     For easier editing, you can copy the content of the field "Value" into an external JSON editor.

  2. Navigate in the configuration block " `content:[...]` " to the configuration block of one of the 3 chart widgets

- Open tickets by priority: "instanceId": `"home-dashboard-ticket-chart-widget-priorities" [...]`.
- Open tickets by status: "instanceId": `"home-dashboard-ticket-chart-widget-states" [...]`
- Tickets created within the last 7 days: `"instanceId": "home-dashboard-ticket-chart-widget-new" [...]`

3. Change the ID (number) of the report definition within the chart widget configuration.
   To do this, enter the ID of the report definition whose data is to be displayed in the chart at the parameter "`reportDefinitionID`". This will change the reference to the data basis of the chart.

```
lanes : [],
"content": [
  {
    "instanceId": "home-dashboard-ticket-chart-widget-priorities",
    "configurationId": null,
    "configuration": {
      "id": "home-dashboard-ticket-chart-widget-priorities",
      "name": "Priority Chart Widget",
      "type": "Widget",
      "widgetId": "report-chart-widget",
      "title": "Translatable#Number of open tickets by priority",
      "actions": [],
      "subConfigurationDefinition": null,
      "configuration": {
        "id": "home-dashboard-ticket-chart-widget-priorities-chart",
        "name": "Translatable#Priority Chart",
        "configuration": {...},
        "reportDefinitionId": 4,
        "reportOutputFormat": "CSV",
        "formatConfiguration": {
```

> ⚠️ **Important!**
>
> By specifying the ID in the parameter "`reportDefinitionID`", a global reference is made to a report definition without checking authorisations. The administrator or the report manager is responsible for making the data available to authorised groups of people!
> For more important information on roles and permissions for reports,
> see: Reporting (see page 702)
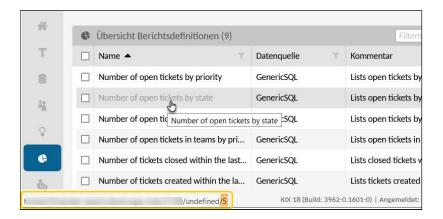
You can find the ID of a report definition in the status bar in the footer of your browser window. To do this, move the mouse pointer over a report definition.

4. Optionally, you can change the display of the chart in the parameter
" `chartConfiguration:{...}` " . The configuration parameters can be found on this page
in the section: Parameters of the chart configuration (see page 348)
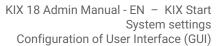
```
"configuration": {
    "id": "home-dashboard-ticket-chart-widget-priorities-chart",
    "name": "Translatable#Priority Chart",
    "configuration": {
        "id": "home-dashboard-ticket-chart-widget-priorities-config",
        "name": "Translatable#Priority Chart",
        "type": "Chart",
        "chartConfiguration": {
            "type": "bar",
            "options": {
                "legend": {
                    "display": false
                },
                "scales": {
                    "yAxes": [
                        {
                            "ticks": {
                                "beginAtZero": true,
                                "maxTicksLimit": 6
                            }
                        }
                    ]
                }
            }
        }
    }
}
```

5. Optionally, you can adjust the CSV mapping in the section " `formatConfiguration:` `{...}` ". The configuration parameters for reports in CSV format can be found in the section Configuration parameters of the Chart Widget (see page 347)

6. Minimise the source code in the JSON editor and copy it back into the "value" field of the configuration key.

7. Save your changes and click "Reload front-end configuration" in the configuration key overview to refresh the view in the Home Dashboard.
After that, the Chart Widget can display the report as statistics.

- **KIX Pro:** Via the personalization of the Home Dashboard, agents can customise the initial chart widgets. Each agent can define for himself which report he would like to have displayed as statistics and how it should be displayed. Thus, you only need to create one or more report definition(s) (see page

702) with output format CSV or JSON and, if possible, periodic creation, so that agents can select the report as a statistic in the Home Dashboard.

The object-information-card-widget

Object-information-card-widgets show different object information in rows and columns, regardless of the type of information. For example:

- static texts
- Texts with placeholders
- Avatars and icons
- individual UI components (e.g. for displaying dynamic fields)
- internal and external hyperlinks

You can also

- Conditions are specified which must apply in order for information to be displayed.
- individual CSS style information for icons and texts can be specified.

Object-information-card-widgets are for example the widget "Contact information" or the ticket information in the ticket details (as part of the lane).

The widgets can be used in all contexts; the limits here are the placeholder support.

The icons in the widget also support placeholders so that they can be used dynamically.

A configuration example can be found at: Configuring an object-information-card-widget (see page 473)

Fig .: The widget "Contact information" as an example of an object information card widget.

> ⓘ **Notes for displaying the content in the widget:**
>
> - If the information requested with the configuration is not available at runtime, this information will not be displayed.
> - If the contact of a ticket is not known, the links "New Organization" and "New Contact" are displayed.
> - If the contact of a ticket is known, the links "Open organization" and "Open contact" are displayed.

Basic structure of the widget

Each widget has its own individual configuration. This configuration can bring specific settings (sub-configurations) for the content.

The configuration of an object information card widget is structured as follows:

```
{
    "avatar": {string | ObjectIcon},
    "rows": [
      {  //Content of a row
        "separator": true | false,
        "values": [
          {
            "icon": string | ObjectIcon,
            "text": string,
            "linkSrc": string,
            "routingConfiguration": RoutingConfiguration |
DialogRoutingConfiguration,
            "routingObjectId": string
          }
        ]
      }
      {  //optional content of another row
        "separator": true | false,
        "values": [...]
      }
    ]
}
```

The Avatar

The avatar is a photo or icon of a contact person or organization. This can be stored when creating or editing a contact or an organization. If an avatar is stored for the contact or the organization, this is automatically displayed in the "Contact information" widget. If this is not desired, you can remove the code block " `avatar: {...}` " in the source code of the Contact Card widget.

Fig .: The avatar in the "Contact Information" widget

If you want to integrate an avatar into your own widget, for example, copy the following code block into the "configuration":{...}" block of the widget content. The following data types are permitted:

- String (e.g. " `avatar": "kix-icon-gear` ")
- Object icon (icon from / system / objecticons)

Initially, the avatar has a fixed height of 98px, the width is adjusted according to the image format. We recommend using square or portrait images.

Standard Codeblock for the Avatar

```
"avatar":
   {
      "displayValues": [],
      "KIXObjectType": "OBJECT_ICON",
      "ObjectId": "<KIX_CONTACT_ID>",
      "Object": "Contact",
      "ContentType": null,
      "Content": null,
      "fallbackIcon": "kix-icon-man"
   },
```

Rows

Rows are the lines that contain the information to be displayed. Each line within the Rows block begins and ends with a curly bracket. Within the individual lines there are one or more values that are placed in square brackets. The definition of the individual values are in turn placed in curly brackets.

- separator: optional horizontal dividing line between the lines
- values: the actual information values; multiple values can be specified.

Example code: Structure of rows with groups and values

```
{
  "rows": [
    {
        "separator":false,
        "values": [
            [
               {Value 1},
               {Value 2},
               {Value 3}
            ],
            [
               {Value 4}
            ]
        ]
    },
    {
        "values": [
            [
               {Value 5}
            ],
            [
               {Value 6}
```

```
            ],
            [
                {Value 7}
            ]
        ]
    }
  ]
}
```

The example code delivers the following result:

Value 1 **Value 2** **Value 3**	**Value 4**	
Value 5	Value 6	Value 7

Values

Values define the content of the line.

Values are a property of rows. They are a list of groups. A group is a list of values. A value has the following properties:

Attribute	Descripon	Example
icon	Icon in front of the object (see also Icons (see page 364))	`"icon": "kix-icon-mail"`
iconStyle	CSS styling for the icon	`"iconStyle": "margin:0.5em"`
text	Information text (can contain placeholders)	`"text": "Dynamic Field Checklist"`

Attribute	Descripon	Example
textPlaceholder	List of placeholder values for the text	`"textPlaceholder": [` `"<KIX_TICKET_AccountedTime>"` `]`
textStyle	CSS styling for the text	`"textStyle": "color:#5b5b5b;font-style:italic",`
linkSrc	HTML href-Specification (see also External Link (see page 366))	• `"linkSrc" : "http://www.example.de"` • `"linkSrc": "mailto:max.mustermannl@example.de`[12]`"` • `"linkSrc":"tel:037112343"` • `"linkSrc":"https://www.kixdesk.com"`
routingConfiguration	system-internal linking (see also Internal Link (see page 365))	`"routingConfiguration": { ... }`
routingObjectId	ID of the object to be linked to (can also contain placeholders)	`"routingObjectId": "<KIX_TICKET_OrganisationID>"`

---

12 http://example.de

Attribute	Descripon	Example

conditions	Sets the conditions for displaying the values. The values are only displayed if the conditions are fulfilled.
	(see also: Conditions (see page 367))
	
	Conditions can be defined based on object properties or based on values.
	
	Note: The attribute "propertyValue" is no longer supported. Instead, only "property" is used.
		``
		"conditions": [
		  {
		    "property":
		"DynamicFields.DFText",
		    "operator": "NE",
		    "value": null,
		    "useObjectService":
		false,
		    "useDisplayValue":
		false
		  }
		]
		``

componentId	ID of the display component to be used, e.g. • object-avatar-label (e.g. for information stored on the ticket (ContactID, QueueID etc.) • dynamic-field-value (e.g. for dynamic fields of the type checklist or table)	• `"componentId": "object-avatar-label"` • `"componentId": "dynamic-field-value"`

componentData	Parameters for the selected display component.
	
	Each component has individual parameters that are given as `componentData:`
	
	• object-avatar-label expects the parameter "property".
	• dynamic-field-value expects the parameter "name".
		• `"componentId": "object-avatar-label"`, `"componentData": {`
		    `"property":`
		`"DynamicFields.CloseCode"`
		  `}`
		• `"componentId": "dynamic-field-value"`, `"componentData": {`
		    `"name": "DFTable"`
		  `}`

Attribute	Descripon	Example
detailViewWidthFactor	Extending the display width over several columns (see examples below). Maximum value: 4 Can be used, for example, • to display long email addresses in the contact details in full, • fully display the "Work order" field in the ticket details. • Display dynamic field of type "Table" completely.	`"componentId": "dynamic-field-value",` `"detailViewWidthFactor": "3",` `"componentData": {` `"name": "DFTable"` `}`

```
"rows": [
      {
          "separator": false,
          "values": [
            [
                {
                    "icon": null,
                    "text": "<KIX_CONTACT_Firstname> <KIX_CONTACT_Lastname>",
                    "linkSrc": null
                },
                {
                    "icon": {
                        "displayValues": [],
                        "KIXObjectType": "OBJECT_ICON",
                        "ObjectID": "<KIX_ORG_ID>",
                        "ObjectId": "<KIX_ORG_ID>",
                        "Object": "Organisation",
                        "ContentType": null,
                        "Content": null,
                        "fallbackIcon": "kix-icon-man-house"
                    },
                    "text": "<KIX_ORG_Name>",
                    "linkSrc": null
                }
            ]
          ]
      },
```

References:

Further information can also be found on GitHub:

- https://github.com/kix-service-software/kix-frontend/tree/master/doc
- https://github.com/kix-service-software/kix-frontend/commit/
  492f279a5013222c544fecb338a084003008dc72

Icon

Icons are symbols that can be placed in front of information and the widget title.

If the widget title is preceded by an icon, this can also be found on the button for showing and hiding the sidebar. Icons are integrated either of the data type `string` or `ObjectIcon`.

Data type `ObjectIcon` : Use this data type to display the avatar (picture / icon) stored on a contact, user or organization.

```
icon": {
    "displayValues": [],
    "KIXObjectType": "OBJECT_ICON",
    "ObjectId": "<KIX_ORG_ID>",
    "Object": "Organisation",
    "ContentType": null,
    "Content": null
},
```

Data type `string` : Use this data type to display a KIX icon or another font icon (e.g. fontawesome.com[13]).

```
{
    "margin": true,
    "values": [{
        "icon": "far fa-address-card",
            [...]
        }]
},
{
    "margin": true,
    "values": [{
        "icon": "kix-icon-mail",
            [...]
        }]
},
```

Internal links

You can create internal links, e.g. to open the organization details or to open the dialog for creating a new asset.

The configuration takes place via the specifications of `"routingConfiguration": {...}.`

With " `routingObjectId` ": you can specify the ID for the object to which a link should be made. You can also use placeholders here.

```
"routingConfiguration":
    {
        "contextId": "contact-details",
        "objectType": "Contact",
        "contextMode": "DETAILS",
        "objectIdProperty": "ID",
        "history": false,
        "resetContext": true,
        "contextType": "MAIN"
    },
    "routingObjectId": "<KIX_CONTACT_ID>"
```

External links

You can link to external URLs, for example to

- call up the URL to an external web CMS
- go to the customer's website
- link the email address and telephone number in order to open them with an external communication program
- and much else

To do this, enter the link under " linkSrc " e. g.:

Linktyp	Beispiel
URL	`"linkSrc" : "http://www.example.de"`
Email (static)	`"linkSrc" :` `"mailto:info@mailadresse.de"`
Email (dynamic with placeholders)	`"linkSrc" :` `"mailto:<KIX_CONTACT_Email"`

Linktyp	Beispiel
Telefon (static)	`"linkSrc" :   "tel:+49123456789"`
Telefon (dynamic with placeholders)	`"linkSrc" :   "tel:<KIX_CONTACT_Phone"`

```
{
  "margin": false,
  "values": [
    {
      "icon": null,
      "text": "<KIX_CONTACT_Email>",
      "linkSrc": "mailto:<KIX_CONTACT_Email>"
    }
  ]
},
```

Specify conditions

You can specify conditions. For example, to define which conditions must be met for an information to be displayed in the widget.

In the following code example, an additional text "VIP contact" is displayed in red font in the widget if the comment field of the contact contains the entry "VIP". If this entry is missing in the comment field or the entry in the comment is not identical to the string "VIP", this line will not be displayed in the widget.

```
{
   "text": "Translatable#VIP Contact",
   "textStyle": "font-weight:bold;margin-bottom:0.5rem;color:red",
   "conditions": [
     {
       "property": "<KIX_CONTACT_Comment>",
       "operator": "EQ",
       "value": "VIP"
     }
   ]
}
```

The conditions can be defined based on object properties or based on values.

If the property references an existing property, then the value of this property is used.
If the property references a non-existing property, then an attempt is made to replace the placeholder.

The value specified under "property" is always the initial value to be checked against. E.G.:

- DynamicFields.myDynamicFieldName - Value of a Dynamic Field

- "QueueID" - specification of a property that exists on the ticket, for example.
- "<KIX_TICKET_QueueID>" - placeholder, which will be resolved
- "Any value:" - specification of a fixed string.

Items 3 and 4 can also be mixed with this: "Any value <KIX_TICKET_...>"

Note: The attribute "propertyValue" is no longer supported. Instead, only "property" is used.

Possible comparison operators can be found in the table below: Operator (see page 380)

Attributes of an object-information-card-widgets (selection)

The configuration of an object-information-card-widget contains, among other things. the following attributes:

Keyword	Description	Examples and Hints
`avatar {...}`	Section for the configuration of the avatar (contact photo). Several avatars can be configured.	Data types: `string` | `ObjectIcon` • `string:` "kix-icon-gear" or "fas fa-phone-volume" • `ObjectIcon` : the avatar stored for the contact / organization (Icon aus /system/objekticons)

Keyword	Description	Examples and Hints

`conditions [...]`	Conditions (see page 367) under which a value is to be displayed.
	
	Conditions can also be applied to dynamic fields.
		Condition: Is executed if the dynamic field is not empty
		
		`"conditions":[`
		`  {`
		
		` "property":"DynamicFields.myDynamicFieldName",`
		`    "operator":"NE",`
		`    "value":null`
		`  }`
		` ]`

Keyword	Description	Examples and Hints
	Hiding blank values depends on the display component used and is only possible to a limited extent:	

Keyword	Description	Examples and Hints
	• If the configured text is empty, nothing is displayed.	

Keyword	Description	Examples and Hints
	• If a component is used, it is the responsibility of the component.	

Keyword	Description	Examples and Hints
	• If a group has no values, it is not included.	

Keyword	Descri ption	Examples and Hints
	• If a lin e h a s n o gr o u p s, it is n ot in cl u d e d.	
`configuration {...}`	Contai ns the config uratio n block	```json "configuration": { "id" : "1632723315884", "name" : "1632723315884", "type" : "null", "avatar" : {...}, "rows" : [...] } } ```

Keyword	Description	Examples and Hints
`configurationId`	ID of the configuration	`"name-of-my-own-widget"`
`fallbackIcon`	Icon that is displayed if the icon specified under "icon" cannot be displayed, e.g. "kix-icon-man"	Data type: `string`

Keyword	Description	Examples and Hints
`icon {...}`	Icon which is placed in front of information (e.g. "kix-icon-men") Also defines the icon next to the widget title and on the button for showing and hiding the widget.	Data types: `string` | `ObjectIcon`

Keyword	Description	Examples and Hints
`iconStyle`	CSS style for formatting the icon	`"iconStyle": "color:#ff0000",`
`instanceId`	Creates an instance of the widget.	`"name-of-my-own-widget"`
`linkSrc`	HTML linking of external URLs, KIX URLs, email addresses, telephone numbers, web CRM etc. You can link to a URL	Datentyp: `string` • URL: `"linkSrc": "https://www.domainname.de/optionalerPfad"` • KIX-URL: `"linkSrc": "https://kixURL.de/search?search=%7B%22objectType%22:% [ ... ] name%22:null%7D"` • Email: `"linkSrc": "mailto:max.mustermann@company.de"` • Telephone: `"linkSrc":"tel:+49123456789"`

Keyword	Descri ption	Examples and Hints
	gener ated by KIX in order to call up a specif ic page in the KIX (e.g. a searc h result, a specif ic ticket or a view). To do this, copy the URL of the page and link it with "linkSr c".	

Keyword	Description	Examples and Hints
`margin`	Creates a space upwards if true	`true` | `false`
`minimizable`	Defines whether the widget can be minimized.	`true` | `false`
`minimized`	Defines whether the widget is displayed minimized.	`true` | `false`
`name`	Name of the widget	`"My Own Widget",`

Keyword	Description	Examples and Hints

`operator:`	Operator in conditions
	
	Defines how the value and property of a condition are compared.
	
	· EQ → Equal to
	· NE → Not equal to
		```"conditions": [```
		  `{`
		    `"property": "QueueID",`
		    `"operator": "EQ",`
		    `"value": "2"`
		  `}`

Keyword	Description	Examples and Hints
	• LT → Less than • LE → Less than or equal to • GT → Greater than	

Keyword	Description	Examples and Hints
	• GE → Greater than or equal to • AND → logical AND • OR → logical OR	

Keyword	Descri ption	Examples and Hints
	• IN → lo gi c al IN • C O N T AI N S → c o nt ai n s	

`permissions:`	Roles / permi ssion s for the displa y	```
		"permissions": [
		  {
		    "target": "system/slas",
		    "permissions": [
		        2
		    ],
		    "OR": false,
		    "permissionCheckMaxLayer": "Resource",
		    "value": 2
		  }
		],
		``

Keyword	Description	Examples and Hints
`routingConfiguration {...}`	internal linking, e.g. on contact details	
`routingObjectId`	ID of the object to be linked to (can contain placeholders), e.g. <KIX_CONTACT_ID>	Data type: `string`

Keyword	Description	Examples and Hints
`rows {...}`	Rows are a list of rows (content block). Contains all information that should be displayed in the widget.	Has the following properties: • values - values of the row • title - line title, fixed left lettering • style - individual CSS styling for the line • separator - horizontal dividing line for the line
`separator`	Creates a horizontal separator if true	`true` | `false`

Keyword	Descri ption	Examples and Hints
`text`	Infor matio n text (can contai n place holder s)	Data type: `string`
`textStyle`	CSS styles for text forma tting	`"textStyle": "color:#ff0000",`

Keyword	Description	Examples and Hints
`title`	Title of the widget in the frontend; KIX placeholders can be used. "Translatable #" makes the title translatable. A corresponding pattern must be stored in the Internationalization menu.	`Translatable#Simple Time Accounting`

Keyword	Description	Examples and Hints
`type`	Type of integrated element	`Widget	TableWidget` • Widget: simple map widget like "contact information" • TableWidget: Table widget like "Tickets to Contact"
`values [...]`	Values are a list of groups for displaying values (icon, text, linkSrc, routingConfiguration, etc.)	```"values": [{ "icon": null, "text": "<KIX_CONTACT_ID>", "linkSrc": null }]```

Configuration examples for "value":

Example	Codeblock	Illustration
Use of free text with placeholders (see also configuration in "ticket-details-contact-card-widget"):	```json\n{\n "avatar": [],\n "rows": [\n {\n "style": "",\n "separator": true,\n "values": [\n [\n {\n "icon": "kix-icon-man",\n "text": "This is free text (with placeholder) Owner: {0} & Responsible: {1}",\n "textPlaceholder": [\n "<KIX_TICKET_Owner>",\n\n"<KIX_TICKET_Responsible>"\n],\n "textStyle": "font-weight:bold;font-style:italic;color:red;"\n }\n]\n]\n }\n]\n}\n```	
Use of a component for value representation (Avatar with attribute and name)	```json\n{\n "componentId": "object-avatar-label",\n "componentData": {\n "property": "OrganisationID"\n }\n}\n```	

Example	Codeblock	Illustration
Using a component to display an icon	```{ "componentId": "icon", "componentData": { "icon": { "displayValues": [], "KIXObjectType": "OBJECT_ICON", "ObjectId": "<KIX_CONTACT_ID>", "Object": "Contact", "ContentType": null, "Content": null, "fallbackIcon": "kix-icon-man-bubble" }, "style": "width: 5rem;height:5rem;font-size:5rem;" } }```	

Example	Codeblock	Illustration
Use of a component to display a dynamic field (including conditions)	(see code below)	

```
{
  "text": "Mobile Processing Checklist
010",
  "textStyle": "font-
weight:bold;margin-bottom:0.5rem",
  "icon": "kix-icon-ci",
  "componentId": "dynamic-field-value",
  "componentData": {
      "name":
"MobileProcessingChecklist010"
  },
  "conditions": [
    {
      "property":
"DynamicFields.MobileProcessingCheckli
st010",
      "operator": "NE",
      "value": null,
      "useObjectService": true,
      "useDisplayValue": null,
      "propertyValue": null
    }
  ]
},
{
  "text": "Affected Assets",
  "textStyle": "font-
weight:bold;margin-bottom:0.5rem",
  "icon": "kix-icon-ci",
  "componentId": "dynamic-field-
value",
  "componentData": {
      "name": "AffectedAsset"
  },
  "conditions": [
    {
      "property":
"DynamicFields.AffectedAsset",
      "operator": "NE",
      "value": null,
      "useObjectService": true,
      "useDisplayValue": null,
      "propertyValue": null
    }
  ]
}
```

Example	Codeblock	Illustration
Extension of the display width by means of "detailViewWidthFactor"	Full display of long email addresses in the contact details. (SysConfig key: *"contact-details-info-widget"*) ```\n{\n "componentId": "object-avatar-\nlabel",\n "detailViewWidthFactor": 2,\n "componentData": {\n "property": "Email"\n }\n}\n```	
	Complete display of the field "Work order" in the ticket details (SysConfig key: *"ticket-details-info-card"*) ```\n{\n "componentId": "dynamic-field-\nvalue",\n "detailViewWidthFactor": 4,\n "componentData": {\n "name": "WordkOrder"\n },\n "conditions":[{...}],\n}\n```	

The table widget

The table widget is used to display individual object tables. Table widgets are, for example, the ticket and article overview in the home dashboard or

the "Recommended FAQ", "Tickets to assets", "Assigned Assets" and "Tickets to contact" widgets.
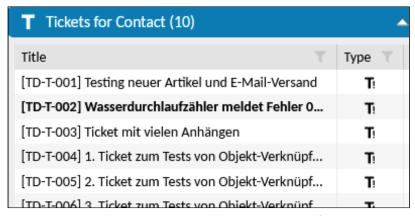
Fig .: The widget "Tickets to Contact" as an example of a table widget

The values in a table widget always result from a configured search. The result of this search is displayed in the table configured in the widget. In the table configuration you can, among other things, define:

- which object is referenced (object type). I. E. which data is displayed in the table (data from tickets, assets, FAQ, contacts or organizations)
- which columns should be displayed
- the table properties (initial sorting, fixing of the 1st column, row height)
- Widget title and icon of the widget
- Behavior (can be minimized / minimized)
- Filters and Actions
- selection
- toggle (expanding a line)
- etc.

The configuration of the widget allows the specification of:

- fixed values of the ticket search
- the use of placeholders to use dynamic values of the current object for the search.

Basic structure of the table widget

Each widget has its own individual configuration. This configuration can bring specific settings (sub-configurations) for the content. In the table widget, these are the settings for the table to be displayed.

Vereinfachter Aufbau eines Tabellenwidgets

```
{
    Widget {
        Widget configuration {
            Table in the widget {
                Table configuration{
                    LoadingOptions: {
                        filter,
                        sort,
                        limit
                    },
                    tableColumns: [
                        {Configuration Column 1},
                        {Configuration Column 2},
                        {Configuration Column 3},
                        {etc.}
                    },
                    optional ObjectHandlers[],
                }
            }
        }
    }
}
```

Table configuration attributes (selection)

The table is configured within the widget configuration (code block "configuration {...}"). This configuration defines the object to be referenced, the table properties and the behavior of the table.

It also contains a configuration block in which the table content (table columns to be displayed along with properties, filters and loading options) are specified (code block "tableConfiguration {...}").

Attribute	Description	Example
id	Unique ID within the configuration	"ticket-new-ticket-affected-asset-table-widget"
name	Name of the widget	"Asset Tickets"
type	Type of the widget	"TableWidget"
objectType	The type of object to be displayed in the table.	Ticket | ConfigItem (Asset) | FAQ | ...
sort	initial sorting of the table	"Age","Up" (ascending by age)
subConfigurationDefinition	optional sub-configuration	
tableConfiguration {...}	Contains the specific configuration of the table in the widget (see below)	
headerComponents	Additional components for the widget header	
showFilter	Defines whether the filter (funnel) should be displayed in the header	true | false
predefinedTableFilters	Predefined filters that are available for selection in the header dropdown. e. g. in the ticket dashboard:	

Attributes of the configuration of the table content (selection)

The table configuration is defined under " `tableConfiguration {...}` " (optional). If this configuration is removed, a standard table for the objectType is used.

Attribute	Description	Example
objectType	Object type to be loaded in the table	ticket | asset | configItem | ...
loadingOptions {...}	Specific loading options for the objects to be loaded (API filter) Defines the table columns and their properties (see below)	
displayLimit	Display limit for the number of table rows; determines the maximum height of the widget. The specification defines the default value. In the "Personalisation of the Home Dashboard", the agent can set the display limit himself by specifying the parameter "Table Height".	10
tableColumns {...}	Defines which columns should be displayed (see below). One configuration block is specified for each column. If nothing is specified, a standard configuration is used for this object.	

Attribute	Description	Example
tableColumnsConfigurations	Configuration of individual columns (optional. If not specified, standard columns are used.	
enableSelection	Activate / deactivate selection in the table	
toggle	Defines whether lines can be expanded (e.g. show article content)	
toggleOptions	Defines what should be displayed when a line is expanded (individual UI component)	
sortOrder	Defines the sorting of the table contents when the table is loaded	null
headerHeight	Height of the header in pixels	1.75 (with dot)
rowHeight	Height of the table rows in pixels	1.75 (with dot)
emptyResultHint	Defines the text to be displayed when no lines are available.	"No data found"
fixedFirstColumn	If true, the first column is fixed and does not scroll horizontally	true | false

Attribute	Description	Example

additionalTableObjectsHandler	List of the handler configurations that determine further objects for the table (optional).
	
	The determination depends on the handler.
		The following attributes are possible:
		• **id:** Unique identifier for internal processing.
		• **name:** name (description)
		• **handlerConfiguration:** specific options of the handler.
		For example, the SuggesteFAQHandler currently supports:
		   • `minLenght:` Indicates how long the words in the title / subject must be at least (separated by spaces) so that they can be used for the search (default: 3).
		   • `onlyValid:` Indicates whether only valid objects are determined (default: false).
		• **dependencyPropertys:** List of object properties (attributes) which are taken into account for the search.
		The information is relevant for the handler. For example, the SuggestedFAQHandler uses it to determine the properties of the words. The list also determines whether this configuration makes it necessary to reload the table in the event of changes in the dialog/form.
		It is used to avoid unnecessary searches for the handlers involved. If this list is not available, the table is reloaded with each change.

Attribute	Description	Example
intersection	Defines how the search is carried out and how the results are combined in the display (true | false) Notes can also be found under: Configuration of table widget "Suggested FAQ" (see page 480)	• **true** = intersection Searches, for example, for keywords in the ticket subject / title **AND** for affected assets. Both must be specified, otherwise the table is empty • **false** (default)= combine = all results (without duplications) Searches, for example, for keywords in the ticket subject / title **OR** for the assets concerned. One of the two must be specified.

Attributes of the LoadingOptions

By specifying the LoadingOptions, you can define which properties an object (Ticket/Asset/FAQ/...) must have in order for it to be loaded. A corresponding filter can be defined for each property (attribute) of the object.

By using placeholders in the LoadingOptions, it is possible to refer to the currently displayed object and thus to design the display in the tables dynamically. The agents can thus be provided with the following information in a table, for example:

- the open inquiries of an organization
- all closed or open child tickets
- open tickets of the team, which are of the same type
- the contacts of the current organization
- new tickets from an organization that are of the same type
- Assets associated with the current contact
- etc.

You can configure separate LoadingOptions for each context for each object type. This is possible in the:

Context	Configuration key (KIX Start)	Configuration key (KIX Pro)
Ticket Dashboard	tickets	Ticket Dashboard Configuration
Ticket-List Context (the toolbar icons at the top)	ticket-list	User Ticket List

Context	Configuration key (KIX Start)	Configuration key (KIX Pro)
FAQ Dashboard	faq	faq
Organisation/Customer Dashboard	organisations	organisations
Asset Dashboard.	cmdb	cmdb

In this way, for example, the organisations and contact table in the organisations dashboard or the table in the ticket dashboard can be presorted and the maximum number of entries to be loaded can be determined.

The LoadingOptions then consist of nested arrays specifying the object and the LoadingOptions relevant for the object. The first value is the relevant object type (e.g. ticket) and the second value contains the definition of the LoadingOptions (limit, searchLimit, sortOrder, etc.).

```
"loadingOptions": [
  [
    "Ticket",
    {
      "searchLimit": 300,
      "limit": 5,
      "sortOrder": "Ticket.-Age:numeric"
    }
  ]
],
```

If you have changed the LoadingOptions of a context (1), you should then remove the sorting in the subordinate Widget Settings (2) (lines: `"sort":["Age", "Up"],` delete). Otherwise, the configuration set in the context does not apply; the widget settings have priority. After changing the configurations, click on "Reload Fontend Configuration".

Note: After changing the LoadingOptions, clear the cache in the *System > Console* menu with the command: `Console::Command::Maint::Cache::Delete` .
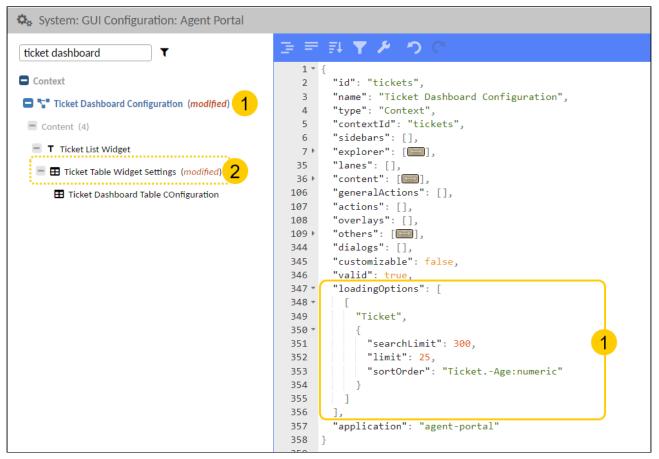
Fig.: Sorting the table in descending order according to the age of tickets (newest first) with limited display quantity

Attribute	Description	Example
expands	Resolves the object IDs received with "includes" to obtain their objects. To do this, specify the same term as for "includes".	Further information can be found at: https://github.com/kix-service-software/kix-backend/blob/master/doc/API/V1/KIX.html (Right click on "Download" > use "Link | Save target as..." in the context menu)

Attribute	Description	Example
filter	Specification of filter conditions for specifying the objects to be displayed. In the example, the table contains all open tickets of the contact. ⚠ **Please note!** In the backend, the status type (`StateType == Open`) is <u>automatically added</u> when searching for a ticket! This affects both the complex search and all configurations that do not have any restrictions on the status type in the filter or have a specific status, e.g. Ticket tables or individually created/ modified widgets and tables.	```json"filter": [{ "property": "StateType", "operator": "EQ", "type": "STRING", "filterType": "AND", "value": "Open" }, { "property": "ContactID", "operator": "EQ", "type": "NUMERIC", "filterType": "AND", "value": "<KIX_TICKET_ContactID>" }]```
includes	Receiving further information on the requested objects, such as links, dynamic fields or existing sub-resources (e.g. articles or history of tickets). For some 'includes', not objects are delivered but only the respective IDs (e.g. for "links"). These IDs can be resolved with "expands" to get their objects.	Further information can be found at: https://github.com/kix-service-software/kix-backend/blob/master/doc/API/V1/KIX.html (Right click on "Download" > use "Link | Save target as..." in the context menu)

Attribute	Description	Example
limit	Number of objects contained in the API response (see info box below). The specified limit controls the size of the "Page" (display limit of the table) and how many data records are reloaded using the "Load More" button. An existing sort order will not be changed during reloading, but will be applied to the reloaded data. Default: 20, if no limit is specified. The specified limit influences the performance. Select a value that is optimal for you, depending on the number of data records in the system. With your specification, you define a system-wide default limit. Each agent can define the table properties for themselves in the customization of the home dashboard.	10

Attribute	Description	Example
searchLimit	Limits the entire search query at database level (see info box below). It defines the maximum number of records to be loaded from the database. The first n records are fetched from the database. The specified limit influences the performance. Select a value that is optimal for you, depending on the number of data records in the system. With your specification, you define a system-wide default limit. Each agent can define the table properties for themselves in the customization of the home dashboard.	100 **T** New Tickets (10/22) □ SLA Kriterien ▼ · Prio ▼ · Ticket# ▼ □ · ■ · 2023040517000311 Info on the illustration: The database contains only 22 data records. Therefore, only a maximum of 22 tickets will be displayed, even if a higher value is set for the searchLimit.

Attribute	Description	Example
sortOrder	Defines the sorting of the table contents when the table is loaded. If nothing is specified, the initial sorting is in descending order: • for tickets: creation time • for assets: asset name, asset number • for organisation: name of the organisation, customer number • for contact: by surname, first name • Dynamic fields • Text, Textarea, Date, Date/Time: by value • Selection: by key • Array values: Sorting the keys within the array and using the first key for sorting the ticket • Other types of dynamic fields cannot be sorted. The entire requested list (SearchLimit) is sorted.	• `null | "Age","Up" | "ID","Down"` • `Ticket.-Age:numeric` (The object *must* be specified; the data type (e.g. numeric) *can* be specified. Specifying the minus sign (-) causes the sorting to be reversed).

ⓘ **Info**

`Limit` and `SearchLimit` have an effect on performance. The higher the values, the longer the loading times may be.
Example:
Given are:

- Number of records in the database: 200
- limit: 10
- searchLimit: 100

> The first 100 records are retrieved from the database. Of these, 10 are displayed in the table. If the agent clicks on the "Load more" button, the next 10 of 100 records in the table are reloaded. This means that the agent can click on "Load more" 10 times to display a maximum of all 100 data records in the table. If the record being searched for is not among these 100 records, the agent can use the search options of the full-text search and the complex search 🔎.

Special feature in the table widget "Assigned Assets":

SysConfig-Key:	ticket-new-assigned-assets

The table widget shows the assets that are assigned to the selected contact. For this purpose, the filter in the LoadingOptions uses the search attribute "AssignedContact", which expects a ContactID. Initially, this is determined by a placeholder (<KIX_TICKET_ContactID>).

The first column of the table allows an interaction with the dynamic field "Affected Asset" (in "New Ticket" and "Edit Ticket"). If an asset is selected by a tick, this asset will automatically be set as "Affected asset" on the ticket. This interaction is controlled by "add-to-affected-assets-cell" in the attribute "componentID". The column can also be implemented on other asset tables if required.

Attributes of TableColumns

The table columns are defined in the code block "tableColumns {..}". Each column begins and ends with a curly bracket: {...}. The attributes of the table column, including their values, are specified within the brackets.

The basis of the values in the table columns is the respective referenced object (ticket / asset / FAQ / ...) and its attributes (properties such as ID, name, title / subject, priority, status, etc.)

Attribute	Description	Example
id	Specification of the column ID; if not defined: null	null
name	Specification of the column name (belonging to the ID); if not defined: null	null
type	Specification of the column type; if not defined: null	null

Attribute	Description	Example
property	Attribute (property) of the object Defines which attribute is displayed in the column	Title / TicketNumber / OwnerID / DynamicFields.myDynamicFieldName
showText	Defines whether a text is displayed for the value of the attribute (true | false)	new | open | closed | ...
showIcon	Defines whether an icon is displayed for the value of the attribute (true | false)	Icon of State
showColumnTitle	Defines whether the name of the attribute is displayed in the column header (true | false)	State
showColumnIcon	Defines whether the icon of the attribute is displayed in the column header (true | false)	Icon of State
size	Size of the column in pixels	320
sortable	Defines whether the column is sortable	true | false
filterable	Defines whether the column can be filtered (filter symbol in the column header)	
hasListFilter	Defines whether a selection list of all values available in the column is offered for filtering (otherwise free text)	
dataType	Data type of the attribute (the property)	
resiziable	Defines whether the user can change the column size	

Attribute	Description	Example
componentID	Defines whether an individual UI component is used in the content of the cell to represent the value. **Note**: The ComponentID can also be set in the advanced configuration mode of the dashboard personalisation. (go to: Home Dashboard > Customise Dashboard > "Advanced" button > "Component ID" column).	Possible values (selection of the most important): • Value of a dynamic field: `"dynamic-field-value"` • Progress bar of a checklist: `"dynamic-field-checklist-cell"` • Label of an SLA criterion: `"sla-criteria-cell"`
defaultText	Text that is used as a column heading if no specific property of the object is represented in the column.	
translatable	Defines whether the attribute values in the cells should be translated automatically If necessary, the corresponding patterns must be stored under *Internationalization> Translations*.	
titleTranslatable	Defines whether the column title should be translated automatically If necessary, the corresponding patterns must be stored under *Internationalization> Translations*.	
useObjectServiceForFilter	For internal use only. (Defines whether the table uses an object service or not.)	

Example Configuration of the table widget "Tickets for Contact"

```
{
   "id": "ticket-details-contact-tickets",
```

```
"name": "Tickets for Contact",
"type": "Widget",
"widgetId": "table-widget",
"title": "Translatable#Tickets for Contact",
"actions": [],
"subConfigurationDefinition": null,
"configuration": {
   "id": "ticket-details-contact-ticket-widget",
   "name": "Contact Tickets",
   "type": "TableWidget",
   "objectType": "Ticket",
   "sort": null,
   "subConfigurationDefinition": null,
   "tableConfiguration": {
      "id": "ticket-details-contact-tickets-table-config",
      "name": "Contact Tickets",
      "type": "Table",
      "objectType": "Ticket",
      "loadingOptions": {
         "filter": [
            {
               "property": "StateType",
               "operator": "EQ",
               "type": "STRING",
               "filterType": "AND",
               "value": "Open"
            },
            {
               "property": "ContactID",
               "operator": "EQ",
               "type": "NUMERIC",
               "filterType": "AND",
               "value": "<KIX_TICKET_ContactID>"
            },
            {
               "property": "TicketID",
               "operator": "NE",
               "type": "NUMERIC",
               "filterType": "AND",
               "value": "<KIX_TICKET_TicketID>"
            }
         ],
         "sortOrder": null,
         "limit": 100
      },
      "displayLimit": 10,
      "tableColumns": [
         {
            "id": null,
            "name": null,
            "type": null,
            "property": "Title",
```

```json
                "showText": true,
                "showIcon": false,
                "showColumnTitle": true,
                "showColumnIcon": false,
                "size": 320,
                "sortable": true,
                "filterable": true,
                "hasListFilter": false,
                "dataType": "STRING",
                "resizable": true,
                "componentId": null,
                "defaultText": null,
                "translatable": true,
                "titleTranslatable": true,
                "useObjectServiceForFilter": false
            },
            {
                "id": null,
                "name": null,
                "type": null,
                "property": "TypeID",
                "showText": false,
                "showIcon": true,
                "showColumnTitle": true,
                "showColumnIcon": false,
                "size": 50,
                "sortable": true,
                "filterable": true,
                "hasListFilter": true,
                "dataType": "STRING",
                "resizable": true,
                "componentId": null,
                "defaultText": null,
                "translatable": true,
                "titleTranslatable": true,
                "useObjectServiceForFilter": false
            }
        ],
        "tableColumnConfigurations": null,
        "enableSelection": false,
        "toggle": false,
        "toggleOptions": null,
        "sortOrder": null,
        "headerHeight": 1.75,
        "rowHeight": 1.75,
        "emptyResultHint": "Translatable#0 data sets found.",
        "fixedFirstColumn": false,
        "additionalTableObjectsHandler": [],
        "intersection": true,
        "searchId": null
    },
    "headerComponents": null,
```

```
        "showFilter": false,
        "shortTable": false,
        "predefinedTableFilters": null,
        "cache": false,
        "resetFilterOnReload": true
    },
    "minimized": false,
    "minimizable": true,
    "icon": "kix-icon-ticket",
    "contextDependent": false,
    "contextObjectDependent": true,
    "formDependent": false,
    "formDependencyProperties": [
        "DynamicFields.myDynamicFieldName"
     ]
}
```

13.1.2.6   Configuration of Sidebar

The right sidebar (1) is available in all ticket-relevant interfaces. It can be configured separately for each context (see page 330) in which it is located (home dashboard, ticket details, dialogs, etc.). This means that the sidebar can contain other widgets (2) - each with different content - in every context. You can define yourself which widgets with which content are to be displayed in which sidebar. The sidebar and the individual widgets are configured in the "*System> SysConfig*" menu.
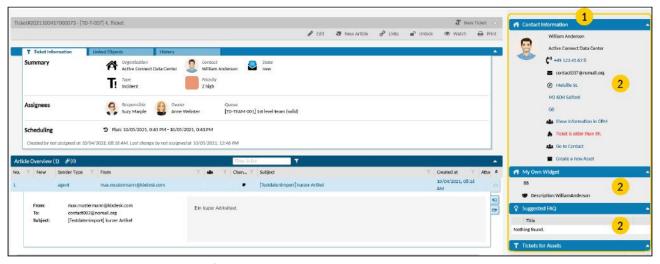
Fig .: The ticket details with the configured sidebar

Overview of the configuration keys

The configuration keys to be used for the sidebar depend on the context in which the sidebar is integrated. The configuration keys of the sidebar widgets also depend on the context of the sidebar.

The following are the most important configuration keys for the sidebar configuration and the widgets initially contained therein. You can use the asterisk wildcard (*) to search for the configuration key, e.g. * card * to search for the object-information-card-widget "contact information".

Configuration key of the sidebar

Sidebar	Configuration key
Ticket details	ticket-details
Dialog "New Ticket"	new-ticket-dialog-context
Dialog "Edit Ticket"	edit-ticket-dialog-context
Dialog "New Article"	new-ticket-article-dialog-context

Configuration key of the widgets

Widget	Configuration key		
	Ticket details	Dialog "New Ticket"	Dialog "Edit Ticket"
Contact informationen	ticket-details-contact-card-widget	ticket-new-contact-card-widget	ticket-edit-contact-card-widget
Recommended FAQ	ticket-details-suggested-faq-widget	ticket-new-dialog-suggested-faq-widget	ticket-edit-dialog-suggested-faq-widget
Tickets to assets	ticket-details-affected-asset-tickets	ticket-new-affected-asset-tickets	ticket-edit-affected-asset-tickets
Tickets to contact	ticket-details-contact-tickets	ticket-new-contact-tickets	n.a.
Open child tickets (KIX Pro)	ticket-details-child-tickets	n.a.	n.a.

Widget	Configuration key		
	Ticket details	Dialog "New Ticket"	Dialog "Edit Ticket"
Time Accounting (KIX Pro)	without The configuration is directly in the sidebar implemented (key: ticket details)	n.a.	n.a.

The basic configuration of the sidebar
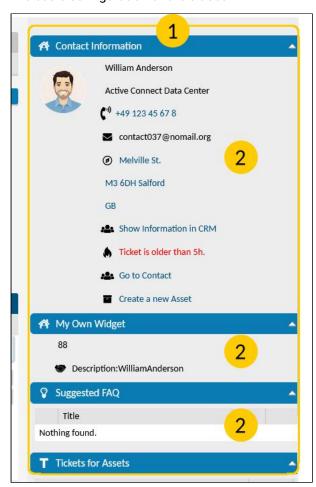
Fig .: The sidebar with widgets

The sidebar consists of the sidebar itself (1) and the widgets contained therein (2). Each context has its own sidebar and contains the code block " `sidbars` ": [...] in its configuration key. A list of widget configurations can be specified in this code block in order to define the content to be displayed in the sidebar.

KIX Start initially contains the following widgets in the ticket details sidebar:

- Contact information
- Recommended FAQ
- Tickets to assets
- Tickets to contact

Example 1: Configuration of sidebar in the ticket zoom view

```
{  //Configuration of the "Ticket
Details" context
   "id": "ticket-details",
   "name": "Ticket details",
   "type": "Context",
   "contextId": "ticket-details",
```

```
    // Sidebar in the ticket details (contains flexibly implementable widgets)
  "sidebars": [
   {   //Instance of the Contact Information widget
       "instanceId": "ticket-details-contact-card-widget",
       "configurationId": "ticket-details-contact-card-widget",
       "permissions": [],
       "size": "large"
   },
   {   //Instance of the Recommended FAQ widget
       "instanceId": "ticket-details-suggested-faq-widget",
       "configurationId": "ticket-details-suggested-faq-widget",
       "permissions": [],
       "size": "large"
   },
   {   //Instance of the Tickets to Assets widget
       "instanceId": "ticket-details-affected-asset-tickets",
       "configurationId": "ticket-details-affected-asset-tickets",
       "permissions": [],
       "size": "large"
   },
   {   //Instance of the widget ticket to contact
       "instanceId": "ticket-details-contact-tickets",
       "configurationId": "ticket-details-contact-tickets",
       "permissions": [],
       "size": "large"
   }
  ],
```

The sidebar of the ticket details in KIX Pro also contains the widgets:

- Time tracking
- Open child tickets

Example 2: Configuration of sidebar in the ticket zoom view (KIX Pro)

```
{   //Configuration of the "Ticket Details" context
    ...
  //Configuration of the sidebar
  "sidebars": [
   { //KIX-Start-Sidebars },
   {
       //Instance of the "Time booking" widget (permanently implemented)
       "instanceId": "ticket-details-simple-time-accounting-widget",
       "configurationId": "ticket-details-simple-time-accounting-widget",
       "configuration": {
           "id": "ticket-details-simple-time-accounting-widget",
           "name": "Simple Time Accounting",
           "type": "Widget",
           "widgetId": "ticket-simple-time-accounting-widget",
```

```
                "title": "Translatable#Simple Time Accounting",
                "actions": [],
                "subConfigurationDefinition": null,
                "configuration": null,
                "minimized": false,
                "minimizable": true,
                "icon": "kix-icon-time",
                "contextDependent": false,
                "contextObjectDependent": false,
                "formDependent": false,
                "formDependencyProperties": []
                },
            "permissions": [],
            "size": "large"
        },
        {
            //Instance of the "Open Child Tickets" widget
            "instanceId": "ticket-details-child-tickets",
            "configurationId": "ticket-details-child-tickets",
            "permissions": [],
            "size": "large"
        }
    ],
}
```

Configuration options

You have the following configuration options:

1. You can reconfigure the configuration of the widgets initially contained in the sidebar, for example by specifying the required permissions for displaying the widget (" `permissions` ": [])
    If necessary, you can change the global configuration directly in the configuration key of the widget (see configuration example: Configuring an object-information-card-widget (see page 473))
    You can find an overview of possible attributes under:
    1. Basics of GUI Configuration (see page 330)
    2. The object-information-card-widget (see page 355)
    3. The table widget. (see page 393)

2. You can remove unwanted widgets by deleting the widget's code block from the sidebar.
    You can reintegrate an instance of a widget into the sidebar at any time. To do this, copy one of the basic configurations under " `sidebars` ": [] and adapt it accordingly.
    You can use example 1 (see page 414) above as a guide.

3. You can add further widgets to the widgets contained in the sidebar whose configuration key already exists in the system (see example: Adding a Widget to the Sidebar (see page 501))
    1. To do this, look for the corresponding widgets in the SysConfig key table and make a note of their exact designation.

2. Duplicate a configuration block (several times if necessary) and replace the values of "`instanceID`" and "`configurationID`" with the respective ID of the widget to be inserted.
    You can use the existing sidebar configuration as a guide.

4. You can integrate your own, individual widgets into the sidebar. However, these must be permanently implemented directly in the sidebar. You can use example 2 as a guide.
    (see also implementation types widgets)

Attributes of the sidebar configuration

The following is a selection of possible attributes for use in the sidebar configuration:

Keyword	Beschreibung	Beispiele und Hinweise
`avatar {...}`	Avatar (see page 355) configuration section (contact photo)	Data types: `string` | `ObjectIcon` • `string`: "kix-icon-gear" oder "fas fa-phone-volume" • `ObjectIcon` : der dem Kontakt/Organisation hinterlegte Avatar (Icon aus /system/objekticons)
`conditions [...]`	Conditions (see page 355) for when a value should be displayed	
`configuration {...}`	Contains the configuration block	
`configurationId`	Configuration ID	`"name-of-my-own-widget"`
`fallbackIcon`	Icon (see page 355) that is displayed if the icon specified under "icon" cannot be displayed, e.g. "kix-icon-man"	Data type: `string`

Keyword	Beschreibung	Beispiele und Hinweise
`icon`	Icon which is placed in front of information (e.g. "kix-icon-men") Also defines the icon next to the widget title and on the button for showing and hiding the widget.	Data types: `string` | `ObjectIcon`
`iconStyle`	CSS style for formatting the icon	`"iconStyle": "color:#ff0000",`
`instanceId`	Creates an instance of the widget.	`"name-of-my-own-widget"`
`linkSrc`	HTML linking (see page 355) of external URLs, KIX URLs, e-mail addresses, telephone numbers, web CRM, etc. You can link to a URL generated by KIX in order to call up a specific page in the KIX (e.g. a search result, a specific ticket or a view). To do this, copy the URL of the page and link it with "linkSrc".	Data type: `string` • für URL: `"linkSrc": "https://www.domainname.de/optionalerPfad"` • für KIX-URL: `"linkSrc": "https://kixURL.de/search?search=%7B%22objectType%22:%[ ... ] name%22:null%7D"` • für E-Mail: `"linkSrc": "mailto:max.mustermann@company.de"` • für Telefon: `"linkSrc":"tel:+49123456789"`
`margin`	Creates a space upwards if `true`	`true` | `false`

Keyword	Beschreibung	Beispiele und Hinweise
`minimizable`	Defines whether the widget can be minimized.	`true | false`
`minimized`	Defines whether the widget is displayed minimized.	`true | false`
`name`	Name of the widget	`"My Own Widget",`
`operator:`	Operator in conditions	• EQ → Equal to • NE → Not equal to • LT → Less than • LE → Less than or equal to • GT → Greater than • GE → Greater than or equal to • AND → logical AND • OR → logical OR • IN → logisches IN • CONTAINS

`permissions:`	Roles / permissions for the display	`"permissions": [`
		`{`
		  `"target": "system/slas",`
		  `"permissions": [`
		    `2`
		  `],`
		  `"OR": false,`
		  `"permissionCheckMaxLayer": "Resource",`
		  `"value": 2`
		`}`
		`],`

`routingConfiguration {...}`	internal linking (see page 355) , e.g. on contact details	

Keyword	Beschreibung	Beispiele und Hinweise
`routingObjectId`	ID of the object to be linked to (can contain placeholders), e.g. <KIX_CONTACT_ID>	Data type: `string`
`rows {...}`	Content block; contains all information that should be displayed in the widget	
`text`	Information text (can contain placeholders)	Data type: `string`
`textStyle`	CSS styles for text formatting	`"textStyle": "color:#ff0000",`
`title`	Title of the widget in the frontend; KIX placeholders can be used. "Translatable #" makes the title translatable. A corresponding pattern must be stored in the *Internationalization* menu.	`Translatable#Simple Time Accounting`

Keyword	Beschreibung	Beispiele und Hinweise
`type`	Type of integrated element	`Widget	TableWidget` • Widget: simple map widget like "contact information" • TableWidget: Table widget like "Tickets to Contact"

`values [...]`	Contains the information values for icon, text, linkSrc, routingConfiguration, routingObjectId, conditions etc.	`"values": [{`
		`   "icon": null,`
		`   "text": "<KIX_CONTACT_ID>",`
		`   "linkSrc": null`
		`}]`

13.1.3 Configuration examples of GUI configuration

In the following chapters you will find a number of configuration examples. These are only intended to serve as suggestions, assistance and orientation and to familiarise you with the configuration of the user interface. If you use the configuration examples and code blocks, you must adapt them to your specific needs.

13.1.3.1 Integrating a Dynamic Field

Dialog	Configuration key	Required object type of the dynamic field
New Ticket	ticket-new-form-group-data	Ticket
Ticket edit	ticket-edit-form-group-data	Ticket
New FAQ Article	faq-article-new-form-group-data	FAQ
Edit FAQ Article	faq-article-edit-form-group-data	FAQ
New Contact	contact-new-form-group-information	Contact
Edit Contact	contact-edit-form-group-information	Contact
New Organisation (Key depends on the information block in the form)	organisation-new-form-group-information organisation-new-form-group-address organisation-new-form-group-other	Organisation
Edit Organisation (Key depends on the information block in the form)	organisation-edit-form-group-information organisation-edit-form-group-address organisation-edit-form-group-other	Organisation

Dynamic fields are individual input and selection fields. If they are integrated into the program interface, additional information can be stored in them.

Dynamic fields can be integrated in various places of the program interface, e.g. in the "New" and "Edit" dialogs of

- Tickets

- FAQ articles
- Contacts and Organizations.

Note: Dynamic fields cannot be deployed to assets. However, you can add custom fields to an asset class definition (see also Editing a Class Definition (see page 130)).

The object type selected in the Dynamic Field configuration determines the object to which the Dynamic Field can be attached. This means: If a dynamic field is assigned to the object type "Ticket", it can only be integrated in ticket-relevant interfaces, but not in contacts or organizations.

The integration of a Dynamic Field is done in the *System > SysConfig* menu by adding a code block to the respective configuration key. The table above contains a selection of the most important configuration keys. The procedure for integration is the same in all keys.

Dynamic fields can only be integrated in system configurations of the type "FormGroup", because only these contain the parameter "formFields". Info about this is provided by the column "Metadata" in the overview of the SysConfig keys and/or the search for " `*form-group*` ".

Integration in the Dialog "New Ticket"

Configuration key	ticket-new-form-group-data

In the following description, we will limit ourselves to the integration of a dynamic field in the "New ticket" dialog as an example. The dynamic field is to be a single-line text field for storing the serial number in the ticket.

The integration of a dynamic field in other dialogs (e.g. "Edit ticket" or "Edit contact") is done in the same way in the corresponding SysConfig keys (see table above).

ⓘ **Info**

The following description is valid for <u>KIX Start</u>!
KIX Pro first uses the "Default - New Ticket Template" template for the ticket creation screen for the "New Ticket" dialog and only uses the (modified) SysConfig key as a fallback. KIX Pro users should therefore integrate dynamic fields directly in the template (*Workflow > Templates* menu).
The dialog "Edit ticket" is a ticket action in KIX Pro (Ticket Edit). This can be opened for editing in the *Workflow > Actions* menu to conveniently include dynamic fields via drop-down selection.
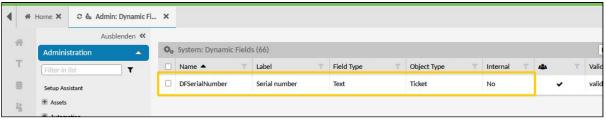
To integrate a dynamic field in the ticket creation dialog, proceed as follows:

1. Create a dynamic field of the field type "Text field" under *System > Dynamic fields* (if not already available).

   Select the object type "Ticket", as the dynamic field is to be integrated into a ticket-relevant interface.

   Make a note of the name of the dynamic field. You will need it to reference it (see step 5).

Alternative:

Select a different field type, if necessary, depending on what type of Dynamic Field you need.

If necessary, select a different object type, depending on the object (contact, organization, asset, etc.) where you want the Dynamic Field to be available.

2. Navigate to *System > SysConfig*. Open key *"ticket-new-form-group-data"*. This key defines the structure of the ticket creation screen.

Alternative:

Select another SysConfig key if necessary (see table above). The selection depends on the interface into which the dynamic field is to be integrated.

3. Open key and copy source code from "Value" field to JSON editor. To make it easier to edit the code, the right-hand column contains the source code in maximized format.

4. Insert the following source code (without comments!) into the code block under `"formFields: [...]"`. Place it before or after an existing code section (e.g. "`ticket-new-form-field-affectedasset"[...]`).

To include several dynamic fields, insert the source text several times. Separate the individual code sections with commas.

Tip: With the code section "`ticket-new-form-field-affectedasset"[...],` the selection field "Affected Assets" is integrated into the ticket creation screen, which is also a Dynamic Field. You can optionally change the parameters for the field or remove the field from the ticket creation screen. You can also use the code block as a template for integrating your own dynamic fields by duplicating the code block and adjusting the configuration.

```
1    {
2        "id": "ticket-new-form-field-dfserialnumber", //unique ID of the
     DF in the key
3        "label": null,
4        "property": "DynamicFields",
5        "inputComponent": null,
6        "required": false,
7        "hint": "Translatable#Seriennummer des Geräts",
8        "options": [
9            {
10               "option": "FIELD_NAME",
11               "value": "DFSerialNumber"   //Name of the DF as created in
     the system
12           }
13       ],
14       "defaultValue": {
15           "value": null,
16           "valid": true,
17           "errorMessages": []
18       },
```

```
19        "fieldConfigurationIds": [],
20        "children": [],
21        "parentInstanceId": null,
22        "countDefault": null,
23        "countMax": null,
24        "countMin": null,
25        "maxLength": null,
26        "regEx": null,
27        "regExErrorMessage": null,
28        "empty": false,
29        "asStructure": false,
30        "readonly": false,
31        "placeholder": null,
32        "existingFieldId": null,
33        "showLabel": true,
34        "name": null,
35        "draggableFields": false,
36        "defaultHint": "Translatable#optionaler_ Hilfetext",
37        "type": "FormField",
38        "visible": true,
39        "translateLabel": true,
40        "instanceId": null
41    },
```

Alternative:

You can also use the source code in the other configuration keys mentioned above.

Always insert the source code between the square brackets of the "formFields" parameter. Depending on the configuration key, this parameter may be initially empty (e.g. in the "Edit Contact" dialog).

```
{
    "id": "contact-edit-form-group-information",
    "name": "Translatable#Contact Information",
    "fieldConfigurationIds": [
        "contact-edit-form-field-title",
        "contact-edit-form-field-firstname",
        "contact-edit-form-field-lastname",
        "contact-edit-form-field-organisation",
        "contact-edit-form-field-primary-organisation",
        "contact-edit-form-field-communication-container",
        "contact-edit-form-field-address-container",
        "contact-edit-form-field-other-container"
    ],
    "separatorString": null,
    "formFields": [ ]
    "draggableFields": false,
    "type": "FormGroup",
    "valid": true,
    "application": "agent-portal"
}
```

5. Give the dynamic field an ID. To do so:

   1. Replace example ID " `ticket-new-formfield-dfserialnumber` " with your own meaningful ID.
      You can use any name you like for the ID, but it must only occur once in the entire form and must not be identical to the value set under "value".
      That is why we recommend you use a meaningful ID.

2. Under "value", replace the example value "DFserialNumber" with the name of the dynamic field to be included (from step 1). The name of the dynamic field defined under *System > Dynamic Fields* must be entered.

6. Add the ID of the dynamic field (from 5a) to the code block " `fieldConfigurationIDs` ". The order of the field IDs determines the order of the form fields in the ticket creation screen.
In the example, the dynamic field will be included in the third place - between "Organisation" and "Affected asset" in the form.
If you do not enter an ID here, the dynamic field will always be included at the last position in the form, in the order in which the fields in the code block " `formFields[...]` " are included.

```
{
    "id": "ticket-new-form-group-data",
    "name": "Translatable#Ticket Data",
    "fieldConfigurationIds": [
        "ticket-new-form-field-contact",
        "ticket-new-form-field-organisation",
        "ticket-new-form-field-serialnumber",
        "ticket-new-form-field-affectedasset",
        "ticket-new-form-field-type",
        "ticket-new-form-field-queue",
        "ticket-new-form-field-channel",
        "ticket-new-form-field-owner",
        "ticket-new-form-field-priority",
        "ticket-new-form-field-state"
    ],
    "separatorString": null,
    "formFields": [
        {
            "id": "ticket-new-form-field-serialnumber", //unique ID of the DF in the key
            "label": null,
            "property": "DynamicFields",
            "inputComponent": null,
            "required": false,
            "hint": "Translatable#Serial number",
            "options": [
                {
                    "option": "FIELD_NAME",
                    "value": "DFSerialNumber"   //Name of the DF (how in KIX created)
                }
            ],
            "defaultValue": {
                "value": null,
                "valid": true,
                "errorMessages": []
            },
            "fieldConfigurationIds": [],
            "children": [],
            "parentInstanceId": null,
            "countDefault": null
```
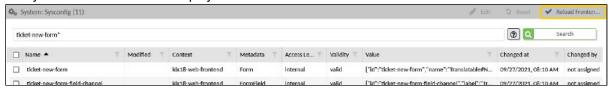
7. **Important:** Minify source code to remove superfluous blanks and line breaks and copy source code to clipboard.

8. Paste source code from clipboard back into "Value" field and click "Save" to apply changes.

9. Click "Reload Frontend Configurations" to update fields of ticket creation screen. If you do not do this, the dynamic field will not be displayed.

10. The dynamic field will be displayed when opening the ticket creation screen. The serial number can now be saved directly in the ticket.

> ✅ **Tip**
>
> In the Admin Manual of KIX Pro you will find a detailed description of how to provide and display checklists.

Configuration examples

Below you will find examples of already modified configuration keys for your information. To do so, expand the respective area.

> ⚠️ Please note that the following are configuration **examples**. We do not assume any liability if you transfer the configurations 1:1 into your system.

Dialog "Neues Ticket" (ticket-new-form-group-data)

```
 1  {
 2      "id": "ticket-new-form-group-data",
 3      "name": "Translatable#Ticket Data",
 4      "fieldConfigurationIds": [
 5          "ticket-new-form-field-contact",
 6          "ticket-new-form-field-organisation",
 7          "ticket-new-form-field-myDynamicField",
 8          "ticket-new-form-field-affectedasset",
 9          "ticket-new-form-field-type",
10          "ticket-new-form-field-queue",
11          "ticket-new-form-field-channel",
12          "ticket-create-form-field-timeunit",
13          "ticket-new-form-field-owner",
14          "ticket-new-form-field-priority",
15          "ticket-new-form-field-state"
16      ],
17      "separatorString": null,
18      "formFields": [
19          {
20              "id": "ticket-new-form-field-myDynamicField",
21              "label": null,
22              "property": "DynamicFields",
23              "inputComponent": null,
24              "required": false,
25              "hint": "Translatabl#Übersetzbarer Hinweistext",
26              "options": [
27                  {
28                      "option": "FIELD_NAME",
29                      "value": "SomeDynamicField"
30                  }
31              ],
32              "defaultValue": {
33                  "value": null,
34                  "valid": true,
35                  "errorMessages": []
36              },
37              "fieldConfigurationIds": [],
38              "children": [],
39              "parentInstanceId": null,
40              "countDefault": null,
41              "countMax": null,
42              "countMin": null,
43              "maxLength": null,
44              "regEx": null,
45              "regExErrorMessage": null,
46              "empty": false,
47              "asStructure": false,
48              "readonly": false,
49              "placeholder": null,
```
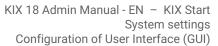
```
50              "existingFieldId": null,
51              "showLabel": true,
52              "name": null,
53              "draggableFields": false,
54              "defaultHint": "Translatable#optionaler_ Hilfetext",
55              "type": "FormField",
56              "visible": true,
57              "translateLabel": true,
58              "instanceId": null
59          },
60          {
61              "id": "ticket-new-form-field-affectedasset",
62              "label": null,
63              "property": "DynamicFields",
64              "inputComponent": null,
65              "required": false,
66              "hint":
    "Translatable#Helptext_Tickets_TicketCreate_AffectedAsset",
67              "options": [
68                  {
69                      "option": "FIELD_NAME",
70                      "value": "AffectedAsset"
71                  }
72              ],
73              "defaultValue": {
74                  "value": null,
75                  "valid": true,
76                  "errorMessages": []
77              },
78              "fieldConfigurationIds": [],
79              "children": [],
80              "parentInstanceId": null,
81              "countDefault": null,
82              "countMax": null,
83              "countMin": null,
84              "maxLength": null,
85              "regEx": null,
86              "regExErrorMessage": null,
87              "empty": false,
88              "asStructure": false,
89              "readonly": false,
90              "placeholder": null,
91              "existingFieldId": null,
92              "showLabel": true,
93              "name": null,
94              "draggableFields": false,
95              "defaultHint":
    "Translatable#Helptext_Tickets_TicketCreate_AffectedAsset",
96              "type": "FormField",
97              "visible": true,
98              "translateLabel": true,
99              "valid": true,
100             "application": "agent-portal",
```

```
101            "instanceId": null
102        },
103        {
104            "id": "ticket-create-form-field-timeunit",
105            "label": "Translatable#Account Time",
106            "property": "TimeUnit",
107            "inputComponent": "number-input",
108            "required": false,
109            "hint": "Translatable#Helptext_Tickets_TicketCreate_TimeUnit",
110            "options": [
111                {
112                    "option": "STEP",
113                    "value": 1
114                },
115                {
116                    "option": "EXCEPTS_EMPTY",
117                    "value": true
118                },
119                {
120                    "option": "UNIT_STRING",
121                    "value": "Translatable#Minutes"
122                }
123            ],
124            "defaultValue": null,
125            "fieldConfigurationIds": null,
126            "children": null,
127            "parentInstanceId": null,
128            "countDefault": null,
129            "countMax": null,
130            "countMin": null,
131            "maxLength": null,
132            "regEx": "^-?\\d+$",
133            "regExErrorMessage": "Translatable#Account time has to be an
     integer.",
134            "empty": false,
135            "asStructure": false,
136            "readonly": false,
137            "placeholder": null,
138            "existingFieldId": null,
139            "showLabel": true,
140            "name": "Translatable#Account Time",
141            "draggableFields": false,
142            "defaultHint":
     "Translatable#Helptext_Tickets_TicketCreate_TimeUnit",
143            "type": "FormField",
144            "visible": true,
145            "translateLabel": true,
146            "valid": true,
147            "application": "agent-portal",
148            "instanceId": null
149        }
150    ],
151    "draggableFields": false,
```

```
152     "type": "FormGroup",
153     "valid": true,
154     "application": "agent-portal"
155   }
```

Dialog "Ticket bearbeiten" (ticket-edit-form-group-data)

```
1   {
2       "id": "ticket-edit-form-group-data",
3       "name": "Translatable#Ticket Data",
4       "fieldConfigurationIds": [
5           "ticket-edit-form-field-title",
6           "ticket-edit-form-field-contact",
7           "ticket-edit-form-field-organisation",
8           "ticket-edit-form-field-myDynamicField",
9           "ticket-edit-form-field-affectedasset",
10          "ticket-edit-form-field-type",
11          "ticket-edit-form-field-queue",
12          "ticket-edit-form-field-channel",
13          "ticket-edit-form-field-owner",
14          "ticket-edit-form-field-responsible",
15          "ticket-edit-form-field-priority",
16          "ticket-edit-form-field-state"
17      ],
18      "separatorString": null,
19      "formFields": [
20          {
21              "id": "ticket-edit-form-field-planbegin",
22              "label": null,
23              "property": "DynamicFields",
24              "inputComponent": null,
25              "required": false,
26              "hint": null,
27              "options": [
28                  {
29                      "option": "FIELD_NAME",
30                      "value": "PlanBegin"
31                  }
32              ],
33              "defaultValue": {
34                  "value": null,
35                  "valid": true,
36                  "errorMessages": []
37              },
38              "fieldConfigurationIds": [],
39              "children": [],
40              "parentInstanceId": null,
41              "countDefault": null,
42              "countMax": null,
43              "countMin": null,
44              "maxLength": null,
45              "regEx": null,
46              "regExErrorMessage": null,
47              "empty": false,
48              "asStructure": false,
```

```
49          "readonly": false,
50          "placeholder": null,
51          "existingFieldId": null,
52          "showLabel": true,
53          "name": null,
54          "draggableFields": false,
55          "defaultHint": null,
56          "type": "FormField",
57          "visible": true,
58          "translateLabel": true,
59          "valid": true,
60          "application": "agent-portal",
61          "instanceId": null
62      },
63      {
64          "id": "contact-edit-form-field-myDynamicField",
65          "label": null,
66          "property": "DynamicFields",
67          "inputComponent": null,
68          "required": false,
69          "hint": "Translatabl#Übersetzbarer Hinweistext",
70          "options": [
71              {
72                  "option": "FIELD_NAME",
73                  "value": "SomeDynamicField"
74              }
75          ],
76          "defaultValue": {
77              "value": null,
78              "valid": true,
79              "errorMessages": []
80          },
81          "fieldConfigurationIds": [],
82          "children": [],
83          "parentInstanceId": null,
84          "countDefault": null,
85          "countMax": null,
86          "countMin": null,
87          "maxLength": null,
88          "regEx": null,
89          "regExErrorMessage": null,
90          "empty": false,
91          "asStructure": false,
92          "readonly": false,
93          "placeholder": null,
94          "existingFieldId": null,
95          "showLabel": true,
96          "name": null,
97          "draggableFields": false,
98          "defaultHint": "Translatable#optionaler_ Hilfetext",
99          "type": "FormField",
100         "visible": true,
101         "translateLabel": true,
```

```
102              "instanceId": null
103          },
104          {
105              "id": "ticket-edit-form-field-planend",
106              "label": null,
107              "property": "DynamicFields",
108              "inputComponent": null,
109              "required": false,
110              "hint": null,
111              "options": [
112                  {
113                      "option": "FIELD_NAME",
114                      "value": "PlanEnd"
115                  }
116              ],
117              "defaultValue": {
118                  "value": null,
119                  "valid": true,
120                  "errorMessages": []
121              },
122              "fieldConfigurationIds": [],
123              "children": [],
124              "parentInstanceId": null,
125              "countDefault": null,
126              "countMax": null,
127              "countMin": null,
128              "maxLength": null,
129              "regEx": null,
130              "regExErrorMessage": null,
131              "empty": false,
132              "asStructure": false,
133              "readonly": false,
134              "placeholder": null,
135              "existingFieldId": null,
136              "showLabel": true,
137              "name": null,
138              "draggableFields": false,
139              "defaultHint": null,
140              "type": "FormField",
141              "visible": true,
142              "translateLabel": true,
143              "valid": true,
144              "application": "agent-portal",
145              "instanceId": null
146          },
147          {
148              "id": "ticket-edit-form-field-affectedasset",
149              "label": null,
150              "property": "DynamicFields",
151              "inputComponent": null,
152              "required": false,
153              "hint": "Translatable#Helptext_Tickets_TicketEdit_AffectedAsset",
154              "options": [
```

```
155                {
156                    "option": "FIELD_NAME",
157                    "value": "AffectedAsset"
158                }
159            ],
160            "defaultValue": {
161                "value": null,
162                "valid": true,
163                "errorMessages": []
164            },
165            "fieldConfigurationIds": [],
166            "children": [],
167            "parentInstanceId": null,
168            "countDefault": null,
169            "countMax": null,
170            "countMin": null,
171            "maxLength": null,
172            "regEx": null,
173            "regExErrorMessage": null,
174            "empty": false,
175            "asStructure": false,
176            "readonly": false,
177            "placeholder": null,
178            "existingFieldId": null,
179            "showLabel": true,
180            "name": null,
181            "draggableFields": false,
182            "defaultHint":
"Translatable#Helptext_Tickets_TicketEdit_AffectedAsset",
183            "type": "FormField",
184            "visible": true,
185            "translateLabel": true,
186            "valid": true,
187            "application": "agent-portal",
188            "instanceId": null
189        }
190    ],
191    "draggableFields": false,
192    "type": "FormGroup",
193    "valid": true,
194    "application": "agent-portal"
195 }
```

Dialogue "Edit contact" (contact-edit-form-group-information)

```
 1   {
 2       "id": "contact-edit-form-group-information",
 3       "name": "Translatable#Contact Information",
 4       "fieldConfigurationIds": [
 5           "contact-edit-form-field-title",
 6           "contact-edit-form-field-firstname",
 7           "contact-edit-form-field-lastname",
 8           "contact-edit-form-field-organisation",
 9           "contact-edit-form-field-primary-organisation",
10           "contact-edit-form-field-communication-container",
11           "contact-edit-form-field-address-container",
12           "contact-edit-form-field-other-container",
13           "contact-edit-form-field-myDynamicField"
14       ],
15       "separatorString": null,
16       "formFields": [
17           {
18               "id": "contact-edit-form-field-myDynamicField",
19               "label": null,
20               "property": "DynamicFields",
21               "inputComponent": null,
22               "required": false,
23               "hint": "Translatabl#Übersetzbarer Hinweistext",
24               "options": [
25                   {
26                       "option": "FIELD_NAME",
27                       "value": "SomeDynamicField"
28                   }
29               ],
30               "defaultValue": {
31                   "value": null,
32                   "valid": true,
33                   "errorMessages": []
34               },
35               "fieldConfigurationIds": [],
36               "children": [],
37               "parentInstanceId": null,
38               "countDefault": null,
39               "countMax": null,
40               "countMin": null,
41               "maxLength": null,
42               "regEx": null,
43               "regExErrorMessage": null,
44               "empty": false,
45               "asStructure": false,
46               "readonly": false,
47               "placeholder": null,
48               "existingFieldId": null,
```

```
49            "showLabel": true,
50            "name": null,
51            "draggableFields": false,
52            "defaultHint": "Translatable#optionaler_ Hilfetext",
53            "type": "FormField",
54            "visible": true,
55            "translateLabel": true,
56            "instanceId": null
57        }
58     ],
59     "draggableFields": false,
60     "type": "FormGroup",
61     "valid": true,
62     "application": "agent-portal"
63  }
```

Info block in the "Edit organisation" dialog (organisation-edit-form-group-information)

```
1    {
2        "id": "organisation-edit-form-group-information",
3        "name": "Translatable#Organisation Information",
4        "fieldConfigurationIds": [
5            "organisation-edit-form-field-cno",
6            "organisation-edit-form-field-name",
7            "organisation-edit-form-field-type",
8            "organisation-edit-form-field-url",
9            "organsiation-edit-form-field-myDynamicField"
10       ],
11       "separatorString": null,
12       "formFields": [
13           {
14               "id": "organisation-edit-form-field-myDynamicField",
15               "label": null,
16               "property": "DynamicFields",
17               "inputComponent": null,
18               "required": false,
19               "hint": "Translatabl#Übersetzbarer Hinweistext",
20               "options": [
21                   {
22                       "option": "FIELD_NAME",
23                       "value": "SomeDynamicField"
24                   }
25               ],
26               "defaultValue": {
27                   "value": null,
28                   "valid": true,
29                   "errorMessages": []
30               },
31               "fieldConfigurationIds": [],
32               "children": [],
33               "parentInstanceId": null,
34               "countDefault": null,
35               "countMax": null,
36               "countMin": null,
37               "maxLength": null,
38               "regEx": null,
39               "regExErrorMessage": null,
40               "empty": false,
41               "asStructure": false,
42               "readonly": false,
43               "placeholder": null,
44               "existingFieldId": null,
45               "showLabel": true,
46               "name": null,
47               "draggableFields": false,
48               "defaultHint": "Translatable#optionaler_ Hilfetext",
```

```
49            "type": "FormField",
50            "visible": true,
51            "translateLabel": true,
52            "instanceId": null
53          }
54        ],
55      "draggableFields": false,
56      "type": "FormGroup",
57      "valid": true,
58      "application": "agent-portal"
59    }
```

13.1.3.2 Displaying Values of Dynamic Fields

Widget	Configuration key	Required object type of the dynamic field
Ticket Information	ticket-details-info-card	Ticket
Contact Information	contact-details-info-widget	Contact
Organisation Information	organisation-details-info-widget	Organisation
FAQ Information	faq-article-info-widget	FAQ

If an agent stores information in a dynamic field, this information is stored directly on the object (ticket, contact, etc.). This makes it possible to read out this content specifically and display it at the desired location.

The description below is a continuation of the example from chapter "Integrating a Dynamic Field (see page 423) ". The serial number is now to be displayed in the "Ticket information". It is assumed that you have already created a dynamic field for the serial number and integrated it into the ticket creation screen, and that the serial number is specified on a ticket (the dynamic field is filled with a value).

The display of a Dynamic Field in other detail views (e.g. Contact Details) is done in the same way in the corresponding SysConfig keys (see table above).
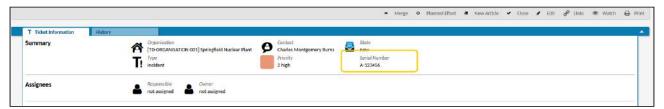
Fig.: Value of a Dynamic Field Displayed in Ticket Information

> ✅ **Tip**
>
> KIX Pro users can conveniently integrate dynamic fields via the integrated JSON editor in the menu *System > GUI Configuration > Agent Portal*. See also: GUI Configuration in the Admin Manual of KIX Pro.

Show value of a dynamic field in ticket details

Configuration key:	ticket-details-info-card

To display the values of a dynamic field in the ticket information, proceed as follows:

1. Navigate to *System > SysConfig* and search for key *"ticket-details-info-card"*.
   This is the key of the object-information-card-widget to display the "Ticket Information".
   Optionally, use another SysConfig key (see table above) to display the value of the dynamic field in another detail view.
2. Open the key and copy the source code in the "Value" field into a JSON editor for easier editing.
3. Copy the following code block into a values block at the desired position.
   Your placement determines where in the widget the value of the dynamic field is displayed.
   To include several dynamic fields, insert the source text several times. Separate the individual code sections with commas.
   1. In the `conditions` code section you can specify conditions under which the Dynamic Field is displayed (optional). In the example, the dynamic field will only be displayed if it also has a value. This means that if no serial number is specified on the ticket, the dynamic field will not be displayed in the ticket details. If no conditions are noted, the field is always displayed.
   2. As an alternative to the `object-avatar-label` component, you can also use `dynamic-field-value,` especially if multi-line texts, checklist or table values are to be displayed (see variant below).

```
{
  "componentId": "object-avatar-label",
  "componentData": {
      "property": "DynamicFields.DFserialNumber"
  },
  "conditions": [
      {
          "property": "DynamicFields.DFserialNumber",
          "operator": "NE",
          "value": null
      }
  ]
},
```

```json
{
    "id": "ticket-details-info-card",
    "name": "Ticket Info Widget",
    "type": "Widget",
    "widgetId": "object-information-card-widget",
    "title": "Translatable#Ticket Information",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
        "id": "1643014545149",
        "name": "1643014545149",
        "type": null,
        "avatar": [],
        "rows": [
            {
                "title": "Translatable#Summary",
                "style": "",
                "separator": true,
                "values": [
                    [🖮],
                    [🖮],
                    [
                        {🖮},
                        {🖮},
                        {
                            "componentId": "object-avatar-label",
                            "componentData": {
                                "property": "DynamicFields.DFSerialNumber"
                            },
                            "conditions": [
                                {
                                    "property": "DynamicFields.DFSerialNumber",
                                    "operator": "NE",
                                    "value": null
                                }
                            ]
                        },
                        {
                            "componentId": "object-avatar-label",
                            "componentData": {
                                "property": "DynamicFields.AnonymiseTicket"
```

4. Replace the Dynamic Field Name (DFSerialNumber) used in the sample code with the name of your own Dynamic Field. The Dynamic Field Name defined under *System > Dynamic Fields* must be entered. Use the following syntax: `DynamicFields.NameDynamicField.`
    If you have included several dynamic fields, replace these names in the same way.

5. Minify source code to remove superfluous blanks and line breaks and copy source code to clipboard.

6. Paste source code from clipboard back into "Value" field and click "Save" to apply changes.

7. Click "Reload Frontend Configurations" to update display of fields in lane. If you do not do this, the dynamic field will not be displayed.

8. The serial number saved in the ticket will now also be displayed in the "Ticket Information" lane.

✅ **Tip**

In the Admin Manual of KIX Pro you will find a detailed description of how to provide and display checklists.

Alternative variant

You can display the value of the dynamic field in a separate content block.
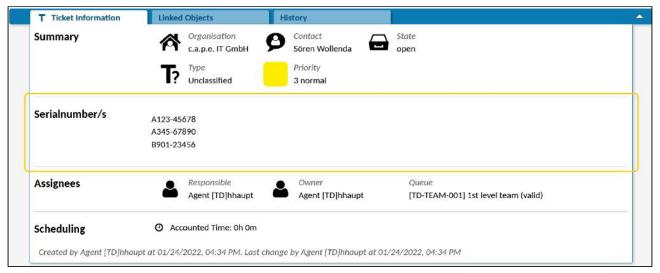
Fig.: Value of a dynamic field (multi-line display)

To do this, insert the following code block under "`rows[...]`". Again, the placement in the code determines the placement in the GUI.

The variant uses `dynamic-field-value` as `componentID`. This component offers individual components for dynamic field values, based on their dynamic field type. It thus enables the output of formatted text, e.g. with line breaks in a multi-line text field. In contrast, the component `object-avatar-label` only provides a single-line value (see also: https://github.com/kix-service-software/kix-frontend/blob/master/doc/components/widgets/object-information-card-widget.md).

The parameters specified under `conditions` are optional.

```
{
   "title": "Translatable#Serialnumber/s",
   "style": "",
   "separator": true,
   "values": [
     {
       "componentId": "dynamic-field-value",
       "componentData": {
          "name": "DFSerialNumber"
        },
       "conditions": [
         {
           "property": "DynamicFields.DFSerialNumber",
           "operator": "NE",
           "value": null
         }
       ]
     }
   ]
},
```

```json
{
    "id": "ticket-details-info-card",
    "name": "Ticket Info Widget",
    "type": "Widget",
    "widgetId": "object-information-card-widget",
    "title": "Translatable#Ticket Information",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
        "id": "1643014545149",
        "name": "1643014545149",
        "type": null,
        "avatar": [],
        "rows": [
            {
                "title": "Translatable#Summary",
                "style": "",
                "separator": true,
                "values": [
                    [⊂▭⊃],
                    [⊂▭⊃],
                    [⊂▭⊃]
                ]
            },
            {
                "title": "Translatable#Serialnumber/s",
                "style": "",
                "separator": true,
                "values": [
                    {
                        "componentId": "dynamic-field-value",
                        "componentData": {
                            "name": "DFSerialNumber"
                        },
                        "conditions": [
                            {
                                "property": "DynamicFields.DFSerialNumber",
                                "operator": "NE",
                                "value": null
                            }
                        ]
                    }
                ]
            },
            {
                "title": "Translatable#Description",
                "style": "",
                "separator": true,
                "values": [
                    [
```

Fig.: Code block for displaying the value of a dynamic field

Configuration examples

Below you will find examples of already modified configuration keys for your information. To do so, open the respective area.

> ⚠ **Please note!**
>
> Please note that the following are configuration **examples**. We do not assume any liability if you transfer the configurations 1:1 into your system.

Zoom view of ticket (Ticket Information)

```
 1  {
 2      "id": "ticket-details-info-card",
 3      "name": "Ticket Info Widget",
 4      "type": "Widget",
 5      "widgetId": "object-information-card-widget",
 6      "title": "Translatable#Ticket Information",
 7      "actions": [],
 8      "subConfigurationDefinition": null,
 9      "configuration": {
10          "id": "1692622169146",
11          "name": "1692622169146",
12          "type": null,
13          "valid": true,
14          "application": "agent-portal",
15          "avatar": [],
16          "rows": [
17              {
18                  "title": "Translatable#Summary",
19                  "style": "",
20                  "separator": true,
21                  "values": [
22                      [
23                          {
24                              "componentId": "object-avatar-label",
25                              "componentData": {
26                                  "property": "OrganisationID"
27                              },
28                              "routingConfiguration": {
29                                  "contextId": "organisation-details",
30                                  "objectType": "Organisation",
31                                  "contextMode": "DETAILS",
32                                  "objectIdProperty": "ID",
33                                  "history": false,
34                                  "externalLink": null,
35                                  "replaceObjectId": null,
36                                  "resetContext": true,
37                                  "params": null,
```

```
38                    "additionalInformation": [],
39                    "contextType": "MAIN"
40                  },
41                  "routingObjectId": "<KIX_TICKET_OrganisationID>"
42              },
43              {
44                  "componentId": "object-avatar-label",
45                  "componentData": {
46                      "property": "TypeID"
47                  }
48              }
49          ],
50          [
51              {
52                  "componentId": "object-avatar-label",
53                  "componentData": {
54                      "property": "ContactID"
55                  },
56                  "routingConfiguration": {
57                    "contextId": "contact-details",
58                    "objectType": "Contact",
59                    "contextMode": "DETAILS",
60                    "objectIdProperty": "ID",
61                    "history": false,
62                    "externalLink": null,
63                    "replaceObjectId": null,
64                    "resetContext": true,
65                    "params": null,
66                    "additionalInformation": [],
67                    "contextType": "MAIN"
68                  },
69                  "routingObjectId": "<KIX_TICKET_ContactID>"
70              },
71              {
72                  "componentId": "object-avatar-label",
73                  "componentData": {
74                      "property": "PriorityID"
75                  }
76              }
77          ],
78          [
79              {
80                  "componentId": "object-avatar-label",
81                  "componentData": {
82                      "property": "StateID"
83                  }
84              },
85              {
86                  "componentId": "object-avatar-label",
87                  "componentData": {
88                      "property": "DynamicFields.DFSerialNumber"
89                  },
90                  "conditions": [
```

```
 91                         {
 92                             "property": "DynamicFields.DFSerialNumber",
 93                             "operator": "NE",
 94                             "value": null
 95                         }
 96                     ]
 97                 },
 98                 {
 99                     "componentId": "object-avatar-label",
100                     "componentData": {
101                         "property": "DynamicFields.CloseCode"
102                     },
103                     "conditions": [
104                         {
105                             "property": "DynamicFields.CloseCode",
106                             "operator": "NE",
107                             "value": null,
108                             "useObjectService": false,
109                             "useDisplayValue": false
110                         }
111                     ]
112                 },
113                 {
114                     "componentId": "object-avatar-label",
115                     "componentData": {
116                         "property": "DynamicFields.AnonymiseTicket"
117                     },
118                     "conditions": [
119                         {
120                             "property": "DynamicFields.AnonymiseTicket",
121                             "operator": "NE",
122                             "value": null,
123                             "useObjectService": false,
124                             "useDisplayValue": false
125                         }
126                     ]
127                 }
128             ]
129         ]
130     },
131     {
132         "title": "Translatable#Description",
133         "style": "",
134         "separator": true,
135         "values": [
136             [
137                 {
138                     "componentId": "dynamic-field-value",
139                     "detailViewWidthFactor": "4",
140                     "componentData": {
141                         "name": "WorkOrder"
142                     },
143                     "conditions": [
```

```
144                                    {
145                                        "property": "DynamicFields.WorkOrder",
146                                        "operator": "NE",
147                                        "value": null,
148                                        "useObjectService": false,
149                                        "useDisplayValue": false
150                                    }
151                                ]
152                            }
153                        ]
154                    ]
155                },
156                {
157                    "title": "Translatable#Assignees",
158                    "style": "",
159                    "separator": true,
160                    "values": [
161                        [
162                            {
163                                "componentId": "object-avatar-label",
164                                "componentData": {
165                                    "property": "QueueID"
166                                }
167                            }
168                        ],
169                        [
170                            {
171                                "componentId": "object-avatar-label",
172                                "componentData": {
173                                    "property": "LockID"
174                                }
175                            }
176                        ],
177                        [
178                            {
179                                "componentId": "object-avatar-label",
180                                "componentData": {
181                                    "property": "OwnerID"
182                                },
183                                "conditions": [
184                                    {
185                                        "property": "OwnerID",
186                                        "operator": "NE",
187                                        "value": 1,
188                                        "useObjectService": false,
189                                        "useDisplayValue": false
190                                    }
191                                ]
192                            }
193                        ],
194                        [
195                            {
196                                "componentId": "object-avatar-label",
```

```
197                         "componentData": {
198                             "property": "ResponsibleID"
199                         },
200                         "conditions": [
201                             {
202                                 "property": "ResponsibleID",
203                                 "operator": "NE",
204                                 "value": 1,
205                                 "useObjectService": false,
206                                 "useDisplayValue": false
207                             }
208                         ]
209                     }
210                 ]
211             ]
212         },
213         {
214             "title": "Translatable#Checklists",
215             "style": "",
216             "separator": true,
217             "values": [
218                 [
219                     {
220                         "text": "Mobile Processing Checklist 010",
221                         "textStyle": "font-weight:bold;margin-bottom:0.5rem",
222                         "icon": "kix-icon-ci",
223                         "componentId": "dynamic-field-value",
224                         "componentData": {
225                             "name": "MobileProcessingChecklist010"
226                         },
227                         "conditions": [
228                             {
229                                 "property":
     "DynamicFields.MobileProcessingChecklist010",
230                                 "operator": "NE",
231                                 "value": null,
232                                 "useObjectService": false,
233                                 "useDisplayValue": false
234                             }
235                         ]
236                     }
237                 ],
238                 [
239                     {
240                         "text": "Mobile Processing Checklist 020",
241                         "textStyle": "font-weight:bold;margin-bottom:0.5rem",
242                         "icon": "kix-icon-ci",
243                         "componentId": "dynamic-field-value",
244                         "componentData": {
245                             "name": "MobileProcessingChecklist020"
246                         },
247                         "conditions": [
248                             {
```

```
249                              "property":
      "DynamicFields.MobileProcessingChecklist020",
250                              "operator": "NE",
251                              "value": null,
252                              "useObjectService": false,
253                              "useDisplayValue": false
254                            }
255                          ]
256                        }
257                      ]
258                    ]
259                  },
260                  {
261                    "title": "Translatable#References",
262                    "style": "",
263                    "separator": false,
264                    "values": [
265                      [
266                        {
267                          "text": "Translatable#Affected Assets",
268                          "textStyle": "font-weight:bold;margin-bottom:0.5rem",
269                          "icon": "kix-icon-ci",
270                          "componentId": "dynamic-field-value",
271                          "componentData": {
272                            "name": "AffectedAsset"
273                          },
274                          "conditions": [
275                            {
276                              "property": "DynamicFields.AffectedAsset",
277                              "operator": "NE",
278                              "value": null,
279                              "useObjectService": false,
280                              "useDisplayValue": false
281                            }
282                          ]
283                        }
284                      ],
285                      [
286                        {
287                          "text": "Translatable#Affected Services",
288                          "textStyle": "font-weight:bold;margin-bottom:0.5rem",
289                          "icon": "kix-icon-ci",
290                          "componentId": "dynamic-field-value",
291                          "componentData": {
292                            "name": "AffectedServices"
293                          },
294                          "conditions": [
295                            {
296                              "property": "DynamicFields.AffectedServices",
297                              "operator": "NE",
298                              "value": null,
299                              "useObjectService": false,
300                              "useDisplayValue": false
```

```
301                                  }
302                             ]
303                         }
304                     ],
305                     [
306                         {
307                             "text": "Translatable#Merged into the Ticket",
308                             "textStyle": "font-weight:bold;margin-bottom:0.5rem",
309                             "icon": "kix-icon-ticket",
310                             "componentId": "dynamic-field-value",
311                             "componentData": {
312                                 "name": "MergeToTicket"
313                             },
314                             "conditions": [
315                                 {
316                                     "property": "DynamicFields.MergeToTicket",
317                                     "operator": "NE",
318                                     "value": null,
319                                     "useObjectService": false,
320                                     "useDisplayValue": false
321                                 }
322                             ]
323                         }
324                     ]
325                 ]
326             },
327             {
328                 "title": " ",
329                 "separator": true,
330                 "values": [
331                     [
332                         {
333                             "text": "Translatable#Related Tickets",
334                             "textStyle": "font-weight:bold;margin-bottom:0.5rem",
335                             "icon": "kix-icon-ci",
336                             "componentId": "dynamic-field-value",
337                             "componentData": {
338                                 "name": "RelatedTickets"
339                             },
340                             "conditions": [
341                                 {
342                                     "property": "DynamicFields.RelatedTickets",
343                                     "operator": "NE",
344                                     "value": null,
345                                     "useObjectService": false,
346                                     "useDisplayValue": false
347                                 }
348                             ]
349                         }
350                     ]
351                 ]
352             },
353             {
```

```
354                "title": "Translatable#Scheduling",
355                "separator": false,
356                "values": [
357                    [
358                        {
359                            "text": "Translatable#Accounted Time: {0}",
360                            "textPlaceholder": [
361                                "<KIX_TICKET_AccountedTime>"
362                            ],
363                            "icon": "kix-icon-time",
364                            "conditions": [
365                                {
366                                    "property": "AccountedTime",
367                                    "operator": "NE",
368                                    "value": null,
369                                    "useObjectService": false,
370                                    "useDisplayValue": false
371                                }
372                            ]
373                        }
374                    ],
375                    [
376                        {
377                            "text": "Translatable#Pending until: {0} ({1})",
378                            "textPlaceholder": [
379                                "<KIX_TICKET_PendingTime>",
380                                "<KIX_TICKET_UntilTime>"
381                            ],
382                            "icon": "kix-icon-time-wait",
383                            "conditions": [
384                                {
385                                    "property": "StateType",
386                                    "operator": "CONTAINS",
387                                    "value": "pending",
388                                    "useObjectService": false,
389                                    "useDisplayValue": false
390                                }
391                            ]
392                        }
393                    ],
394                    [
395                        {
396                            "text": "Translatable#Plan: {0} - {1}",
397                            "textPlaceholder": [
398                                "<KIX_TICKET_DynamicField_PlanBegin>",
399                                "<KIX_TICKET_DynamicField_PlanEnd>"
400                            ],
401                            "icon": "kix-icon-time-back",
402                            "conditions": [
403                                {
404                                    "property": "DynamicFields.PlanBegin",
405                                    "operator": "NE",
406                                    "value": null,
```

```
407                         "useObjectService": false,
408                         "useDisplayValue": false
409                     }
410                 ]
411             }
412         ]
413     ]
414 },
415 {
416     "title": "",
417     "style": "",
418     "separator": false,
419     "values": [
420         [
421             {
422                 "icon": null,
423                 "iconStyle": "",
424                 "text": "Translatable#Created by {0} at {1}. Last
change by {2} at {3}",
425                 "textPlaceholder": [
426                     "<KIX_TICKET_CreateBy>",
427                     "<KIX_TICKET_Created>",
428                     "<KIX_TICKET_ChangeBy>",
429                     "<KIX_TICKET_Changed>"
430                 ],
431                 "textStyle": "color:#5b5b5b;font-style:italic",
432                 "linkSrc": null
433             }
434         ]
435     ]
436 }
437 ]
438 },
439 "minimized": false,
440 "minimizable": false,
441 "icon": "kix-icon-ticket",
442 "contextDependent": false,
443 "contextObjectDependent": false,
444 "formDependent": false,
445 "formDependencyProperties": [],
446 "valid": true,
447 "application": "agent-portal"
448 }
```

Zoom view of Contact (Contact Information)

```
1 {
2     "id": "contact-details-info-widget",
3     "name": "Contact Info Widget",
4     "type": "Widget",
5     "widgetId": "object-information-card-widget",
```

```
 6          "title": "Translatable#Contact Information",
 7          "actions": [],
 8          "subConfigurationDefinition": null,
 9          "configuration": {
10            "id": "1691394743446",
11            "name": "1691394743446",
12            "type": null,
13            "valid": true,
14            "application": "agent-portal",
15            "avatar": [],
16            "rows": [
17              {
18                "values": [
19                  [
20                    {
21                      "componentId": "icon",
22                      "componentData": {
23                        "icon": {
24                          "propertyBindings": [],
25                          "Links": [],
26                          "LinkTypeName": null,
27                          "ChangeBy": null,
28                          "ChangeTime": null,
29                          "CreateBy": null,
30                          "CreateTime": null,
31                          "ValidID": 1,
32                          "Comment": null,
33                          "DynamicFields": [],
34                          "LinkCount": 0,
35                          "displayValues": [],
36                          "displayIcons": [],
37                          "KIXObjectType": "OBJECT_ICON",
38                          "ObjectID": "<KIX_CONTACT_ID>",
39                          "ObjectId": "<KIX_CONTACT_ID>",
40                          "Object": "Contact",
41                          "ContentType": null,
42                          "Content": null,
43                          "fallbackIcon": "kix-icon-man-bubble"
44                        },
45                        "style": "width: 5rem;height:5rem;font-size:5rem;"
46                      }
47                    }
48                  ],
49                  [
50                    {
51                      "componentId": "object-avatar-label",
52                      "componentData": {
53                        "property": "Title"
54                      }
55                    }
56                  ],
57                  [
58                    {
```

```
 59                          "componentId": "object-avatar-label",
 60                          "componentData": {
 61                              "property": "Firstname"
 62                          }
 63                      },
 64                      {
 65                          "componentId": "object-avatar-label",
 66                          "componentData": {
 67                              "property": "Lastname"
 68                          }
 69                      }
 70                  ],
 71                  [
 72                      {
 73                          "componentId": "object-avatar-label",
 74                          "componentData": {
 75                              "property": "PrimaryOrganisationID"
 76                          }
 77                      },
 78                      {
 79                          "componentId": "object-avatar-label",
 80                          "componentData": {
 81                              "property": "ValidID"
 82                          }
 83                      }
 84                  ]
 85              ],
 86              "title": null,
 87              "style": "",
 88              "separator": true
 89          },
 90          {
 91              "values": [
 92                  [
 93                      {
 94                          "componentId": "object-avatar-label",
 95                          "componentData": {
 96                              "property": "UserLogin"
 97                          }
 98                      }
 99                  ],
100                  [
101                      {
102                          "componentId": "object-avatar-label",
103                          "componentData": {
104                              "property": "USER_ACCESS"
105                          }
106                      }
107                  ],
108                  [
109                      {
110                          "componentId": "object-avatar-label",
111                          "componentData": {
```

```
112                    "property": "UserLanguage"
113                }
114            }
115        ]
116    ],
117    "title": null,
118    "style": null,
119    "separator": true
120    },
121    {
122        "values": [
123            [
124                {
125                    "componentId": "object-avatar-label",
126                    "componentData": {
127                        "property": "Phone"
128                    }
129                }
130            ],
131            [
132                {
133                    "componentId": "object-avatar-label",
134                    "componentData": {
135                        "property": "Mobile"
136                    }
137                }
138            ],
139            [
140                {
141                    "componentId": "object-avatar-label",
142                    "componentData": {
143                        "property": "Fax"
144                    }
145                }
146            ],
147            [
148                {
149                    "componentId": "object-avatar-label",
150                    "componentData": {
151                        "property": "Email"
152                    },
153                    "conditions": null,
154                    "icon": null,
155                    "iconStyle": "",
156                    "text": "",
157                    "textPlaceholder": null,
158                    "textStyle": "",
159                    "detailViewWidthFactor": "2"
160                }
161            ],
162            [
163                {
164                    "componentId": "object-avatar-label",
```

```
165             "componentData": {
166                 "property": "Email1"
167             },
168             "conditions": [
169                 {
170                     "property": "Email1",
171                     "operator": "NE",
172                     "value": "",
173                     "useObjectService": false,
174                     "useDisplayValue": false
175                 }
176             ],
177             "icon": null,
178             "iconStyle": "",
179             "text": "",
180             "textPlaceholder": null,
181             "textStyle": "",
182             "detailViewWidthFactor": "2"
183         }
184     ],
185     [
186         {
187             "componentId": "object-avatar-label",
188             "componentData": {
189                 "property": "Email2"
190             },
191             "conditions": [
192                 {
193                     "property": "Email2",
194                     "operator": "NE",
195                     "value": "",
196                     "useObjectService": false,
197                     "useDisplayValue": false
198                 }
199             ],
200             "icon": null,
201             "iconStyle": "",
202             "text": "",
203             "textPlaceholder": null,
204             "textStyle": "",
205             "detailViewWidthFactor": "2"
206         }
207     ],
208     [
209         {
210             "componentId": "object-avatar-label",
211             "componentData": {
212                 "property": "Email3"
213             },
214             "conditions": [
215                 {
216                     "property": "Email3",
217                     "operator": "NE",
```

```
218                        "value": "",
219                        "useObjectService": false,
220                        "useDisplayValue": false
221                      }
222                   ],
223                   "icon": null,
224                   "iconStyle": "",
225                   "text": "",
226                   "textPlaceholder": null,
227                   "textStyle": "",
228                   "detailViewWidthFactor": "2"
229                 }
230               ],
231               [
232                 {
233                   "componentId": "object-avatar-label",
234                   "componentData": {
235                     "property": "Email4"
236                   },
237                   "conditions": [
238                     {
239                        "property": "Email4",
240                        "operator": "NE",
241                        "value": "",
242                        "useObjectService": false,
243                        "useDisplayValue": false
244                      }
245                   ],
246                   "icon": null,
247                   "iconStyle": "",
248                   "text": "",
249                   "textPlaceholder": null,
250                   "textStyle": "",
251                   "detailViewWidthFactor": "2"
252                 }
253               ],
254               [
255                 {
256                   "componentId": "object-avatar-label",
257                   "componentData": {
258                     "property": "Email5"
259                   },
260                   "conditions": [
261                     {
262                        "property": "Email5",
263                        "operator": "NE",
264                        "value": "",
265                        "useObjectService": false,
266                        "useDisplayValue": false
267                      }
268                   ],
269                   "icon": null,
270                   "iconStyle": "",
```

```
271             "text": "",
272             "textPlaceholder": null,
273             "textStyle": "",
274             "detailViewWidthFactor": "2"
275           }
276         ]
277       ],
278       "title": null,
279       "style": null,
280       "separator": true
281     },
282     {
283       "title": "Info",
284       "style": "",
285       "separator": true,
286       "values": [
287         [
288           {
289             "componentId": "object-avatar-label",
290             "componentData": {
291               "property": "DynamicFields.MyDynamicField"
292             }
293           }
294         ]
295       ]
296     },
297     {
298       "values": [
299         [
300           {
301             "componentId": "object-avatar-label",
302             "componentData": {
303               "property": "Street"
304             }
305           }
306         ],
307         [
308           {
309             "componentId": "object-avatar-label",
310             "componentData": {
311               "property": "Zip"
312             }
313           }
314         ],
315         [
316           {
317             "componentId": "object-avatar-label",
318             "componentData": {
319               "property": "City"
320             }
321           }
322         ],
323         [
```

```
324                        {
325                            "componentId": "object-avatar-label",
326                            "componentData": {
327                                "property": "Country"
328                            }
329                        }
330                    ]
331                ],
332                "title": null,
333                "style": null,
334                "separator": true
335            },
336            {
337                "values": [
338                    [
339                        {
340                            "componentId": "object-avatar-label",
341                            "componentData": {
342                                "property": "CreateTime"
343                            }
344                        }
345                    ],
346                    [
347                        {
348                            "componentId": "object-avatar-label",
349                            "componentData": {
350                                "property": "CreateBy"
351                            }
352                        }
353                    ],
354                    [
355                        {
356                            "componentId": "object-avatar-label",
357                            "componentData": {
358                                "property": "ChangeTime"
359                            }
360                        }
361                    ],
362                    [
363                        {
364                            "componentId": "object-avatar-label",
365                            "componentData": {
366                                "property": "ChangeBy"
367                            }
368                        }
369                    ],
370                    [
371                        {
372                            "componentId": "object-avatar-label",
373                            "componentData": {
374                                "property": "DynamicFields.Source"
375                            },
376                            "conditions": [
```

```
377                          {
378                              "property": "DynamicFields.Source",
379                              "operator": "NE",
380                              "value": null,
381                              "useObjectService": false,
382                              "useDisplayValue": false
383                          }
384                      ]
385                  }
386              ]
387          ],
388          "title": null,
389          "style": null,
390          "separator": true
391      },
392      {
393          "values": [
394              [
395                  {
396                      "componentId": "object-avatar-label",
397                      "componentData": {
398                          "property": "Comment"
399                      }
400                  }
401              ]
402          ]
403      }
404      ]
405   },
406   "minimized": false,
407   "minimizable": true,
408   "icon": null,
409   "contextDependent": false,
410   "contextObjectDependent": false,
411   "formDependent": false,
412   "formDependencyProperties": [],
413   "valid": true,
414   "application": "agent-portal"
415  }
```

Zoom view of organisation (Organisation Information)

```
1  {
2     "id": "organisation-details-info-widget",
3     "name": "Organisation Info Widget",
4     "type": "Widget",
5     "widgetId": "object-information-card-widget",
6     "title": "Translatable#Organisation Information",
7     "actions": [],
8     "subConfigurationDefinition": null,
9     "configuration": {
```

```
10            "id": "1692622104699",
11            "name": "1692622104699",
12            "type": null,
13            "valid": true,
14            "application": "agent-portal",
15            "avatar": [],
16            "rows": [
17                {
18                    "style": "",
19                    "separator": true,
20                    "values": [
21                        [
22                            {
23                                "componentId": "icon",
24                                "componentData": {
25                                    "icon": {
26                                        "propertyBindings": [],
27                                        "Links": [],
28                                        "LinkTypeName": null,
29                                        "ChangeBy": null,
30                                        "ChangeTime": null,
31                                        "CreateBy": null,
32                                        "CreateTime": null,
33                                        "ValidID": 1,
34                                        "Comment": null,
35                                        "DynamicFields": [],
36                                        "LinkCount": 0,
37                                        "displayValues": [],
38                                        "displayIcons": [],
39                                        "KIXObjectType": "OBJECT_ICON",
40                                        "ObjectID": "<KIX_ORGANISATION_ID>",
41                                        "ObjectId": "<KIX_ORGANISATION_ID>",
42                                        "Object": "Organisation",
43                                        "ContentType": null,
44                                        "Content": null,
45                                        "fallbackIcon": "kix-icon-man-house"
46                                    },
47                                    "style": "width: 5rem;height:5rem;font-size:5rem;"
48                                }
49                        }
50                    ],
51                    [
52                        {
53                            "componentId": "object-avatar-label",
54                            "componentData": {
55                                "property": "Number"
56                            }
57                        }
58                    ],
59                    [
60                        {
61                            "componentId": "object-avatar-label",
62                            "componentData": {
```

```
 63                             "property": "Name"
 64                         }
 65                     }
 66                 ],
 67                 [
 68                     {
 69                         "componentId": "object-avatar-label",
 70                         "componentData": {
 71                             "property": "DynamicFields.Type"
 72                         }
 73                     }
 74                 ],
 75                 [
 76                     {
 77                         "componentId": "object-avatar-label",
 78                         "componentData": {
 79                             "property": "Url"
 80                         }
 81                     },
 82                     {
 83                         "componentId": "object-avatar-label",
 84                         "componentData": {
 85                             "property": "ValidID"
 86                         }
 87                     }
 88                 ]
 89             ]
 90         },
 91         {
 92             "style": "",
 93             "separator": true,
 94             "values": [
 95                 [
 96                     {
 97                         "componentId": "object-avatar-label",
 98                         "componentData": {
 99                             "property": "Street"
100                         }
101                     }
102                 ],
103                 [
104                     {
105                         "componentId": "object-avatar-label",
106                         "componentData": {
107                             "property": "Zip"
108                         }
109                     }
110                 ],
111                 [
112                     {
113                         "componentId": "object-avatar-label",
114                         "componentData": {
115                             "property": "City"
```

```
116                              }
117                          }
118                      ],
119                      [
120                          {
121                              "componentId": "object-avatar-label",
122                              "componentData": {
123                                  "property": "Country"
124                              }
125                          }
126                      ]
127                  ]
128              },
129              {
130                  "style": "",
131                  "title": "Info",
132                  "separator": true,
133                  "values": [
134                      [
135                          {
136                              "componentId": "object-avatar-label",
137                              "componentData": {
138                                  "property": "DynamicFields.MyDynamicField"
139                              }
140                          }
141                      ]
142                  ]
143              },
144              {
145                  "style": "",
146                  "separator": true,
147                  "values": [
148                      [
149                          {
150                              "componentId": "object-avatar-label",
151                              "componentData": {
152                                  "property": "CreateTime"
153                              }
154                          }
155                      ],
156                      [
157                          {
158                              "componentId": "object-avatar-label",
159                              "componentData": {
160                                  "property": "CreateBy"
161                              }
162                          }
163                      ],
164                      [
165                          {
166                              "componentId": "object-avatar-label",
167                              "componentData": {
168                                  "property": "ChangeTime"
```

```
169                    }
170                }
171            ],
172            [
173                {
174                    "componentId": "object-avatar-label",
175                    "componentData": {
176                        "property": "ChangeBy"
177                    }
178                }
179            ]
180        ]
181    },
182    {
183        "style": "",
184        "separator": false,
185        "values": [
186            [
187                {
188                    "componentId": "object-avatar-label",
189                    "componentData": {
190                        "property": "Comment"
191                    }
192                }
193            ]
194        ]
195    }
196    ]
197 },
198 "minimized": false,
199 "minimizable": true,
200 "icon": null,
201 "contextDependent": false,
202 "contextObjectDependent": false,
203 "formDependent": false,
204 "formDependencyProperties": [],
205 "valid": true,
206 "application": "agent-portal"
207 }
```

Zoom view of FAQ (FAQ Information)

```
1 {
2    "id": "faq-article-info-widget",
3    "name": "FAQ Article Info",
4    "type": "Widget",
5    "widgetId": "object-information-card-widget",
6    "title": "Translatable#FAQ Information",
7    "actions": [],
8    "subConfigurationDefinition": null,
9    "configuration": {
```

```
10          "id": "1692622173540",
11          "name": "1692622173540",
12          "type": null,
13          "valid": true,
14          "application": "agent-portal",
15          "avatar": [],
16          "rows": [
17              {
18                  "style": "",
19                  "separator": true,
20                  "values": [
21                      [
22                          {
23                              "componentId": "object-avatar-label",
24                              "componentData": {
25                                  "property": "CategoryID"
26                              }
27                          }
28                      ],
29                      [
30                          {
31                              "componentId": "object-avatar-label",
32                              "componentData": {
33                                  "property": "CustomerVisible"
34                              }
35                          },
36                          {
37                              "componentId": "object-avatar-label",
38                              "componentData": {
39                                  "property": "DynamicFields.MyDynamicField"
40                              },
41                              "conditions": [
42                                  {
43                                      "property": "DynamicFields.MyDynamicField",
44                                      "operator": "NE",
45                                      "value": null
46                                  }
47                              ]
48                          }
49                      ],
50                      [
51                          {
52                              "componentId": "object-avatar-label",
53                              "componentData": {
54                                  "property": "ValidID"
55                              }
56                          }
57                      ]
58                  ]
59              },
60              {
61                  "style": "",
62                  "separator": true,
```

```
 63                  "values": [
 64                      [
 65                          {
 66                              "componentId": "object-avatar-label",
 67                              "componentData": {
 68                                  "property": "Created"
 69                              }
 70                          }
 71                      ],
 72                      [
 73                          {
 74                              "componentId": "object-avatar-label",
 75                              "componentData": {
 76                                  "property": "CreatedBy"
 77                              }
 78                          }
 79                      ],
 80                      [
 81                          {
 82                              "componentId": "object-avatar-label",
 83                              "componentData": {
 84                                  "property": "Changed"
 85                              }
 86                          }
 87                      ],
 88                      [
 89                          {
 90                              "componentId": "object-avatar-label",
 91                              "componentData": {
 92                                  "property": "ChangedBy"
 93                              }
 94                          }
 95                      ]
 96                  ]
 97              },
 98              {
 99                  "title": "Translatable#References",
100                  "style": "",
101                  "separator": false,
102                  "values": [
103                      [
104                          {
105                              "text": "Translatable#Related Assets",
106                              "textStyle": "font-weight:bold;margin-bottom:0.5rem",
107                              "detailViewWidthFactor": "1",
108                              "icon": "kix-icon-ci",
109                              "componentId": "dynamic-field-value",
110                              "componentData": {
111                                  "name": "RelatedAssets"
112                              },
113                              "conditions": [
114                                  {
115                                      "property": "DynamicFields.RelatedAssets",
```

```
116                         "operator": "NE",
117                         "value": null,
118                         "useObjectService": false,
119                         "useDisplayValue": false
120                   }
121               ]
122           }
123         ]
124       ]
125     }
126   ]
127 },
128 "minimized": false,
129 "minimizable": true,
130 "icon": null,
131 "contextDependent": false,
132 "contextObjectDependent": false,
133 "formDependent": false,
134 "formDependencyProperties": [],
135 "valid": true,
136 "application": "agent-portal"
137 }
```

13.1.3.3   Configuring an object-information-card-widget

Configuration key:	ticket-details-contact-card-widget

The right sidebar in the ticket details contains the widget "Contact information". You can reconfigure its content individually. The following example shows changes to the configuration of this widget of the type object-information-card-widget. You can use this example as a guide if you want to configure other widgets of this type, for example, the ticket information in the ticket zoom view.

The basics of this widget type can be found in the chapter: The object-information-card-widget.
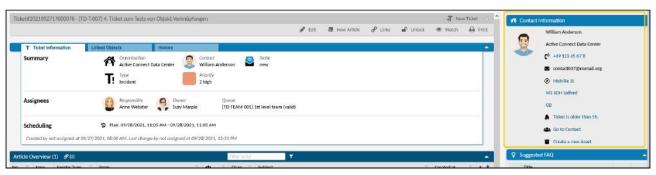
Fig.: The configured widget "Contact information" in the sidebar of the ticket details

The default configuration of the "Contact Information" widget generates the following content:

- Avatar
- First name, last name
- Primary organisation with icon
- Phone number (as URL) with icon
- Email address with icon
- Address as text with link to online map service
- Link to open tickets of the contact and the organisation

Added in the example:

- external linking to a web CRM
- Display of tickets older than 5 hours
- internal link to open the contact
- internal link to create a new asset

Not included in the example:

- Link to open tickets of the contact and the organisation.

To configure the widget "Contact Information" in the ticket details, proceed as follows:

1. In explorer, navigate to *System > SysConfig*.
2. Search for key "*ticket-details-contact-card-widget*". This key defines the structure of the widget "Contact Information" for the ticket zoom view.

3. Open key and copy source code from "Value" field to JSON editor (e.g. https://www.jsonformatter.io).To make it easier to edit the code, the right-hand column in the editor contains the source code in maximized format.

4. Edit source code as required (see example configuration below).

5. Minify edited source code to remove superfluous blanks and line breaks and copy source code to clipboard.

6. Paste source code from clipboard back into "Value" field and click "Save" to apply changes.

7. Click "Reload Frontend Configurations" to update KIX user interface.

Edit the source code

The source code in the configuration key contains a block " `configuration":{...}.` Inside this block is the configuration of the widget content.

The widget content defines the avatar (`"avatar":{...}`) and the content to be displayed row by row (`"rows":[{...}]`).

A row consists of the defined distance to the previous element (("margin": true | false) and the values to be displayed in the row ("values": [{...}]). By specifying further attributes, the display of the value can be influenced. An overview of possible attributes can be found in the basics of sidebar configuration.

Example configuration:

In the example configuration below, the contact address has been linked and various items of additional information have been entered and linked:

Figure of "Contact Information" Widget	Image Number	Description and line number in source code (see below)
	1	Avatar (lines: 10 to 17)
	2	Last name and first name of contact (lines 19 to 28) Organization of the contact (lines: 29 to 45)
	3	Linked telephone number of contact (lines 46 to 55) Email address of contact (lines 56 to 65)
	4	Opens the address for searching in an online card service. (lines 66 to 75 and 76 to 85 as well as 86 to 95)

Figure of "Contact Information" Widget	Image Number	Description and line number in source code (see below)
	5	Opens an external website in a new browser tab. Is displayed if there is a value in the comment for a contact. (lines 96 to 114).
	6	Is displayed if the ticket is older than 5 hours. (lines 115 to 134)
	7	Internal link to details page for contact. (lines 135 to 154)
	8	Internal link; opens a dialog for creating a new asset. (lines 155 to 176)
	without	Widget behavior and icon in widget title (lines: 177 to 184)

Source code of example configuration

```
1   {
2       "id": "ticket-details-contact-card-widget",
3       "name": "Contact Info Widget",
4       "type": "Widget",
5       "widgetId": "object-information-card-widget",
6       "title": "Translatable#Contact Information",
7       "actions": [],
8       "subConfigurationDefinition": null,
9       "configuration": {
10          "avatar": {
11              "displayValues": [],
12              "KIXObjectType": "OBJECT_ICON",
13              "ObjectId": "<KIX_CONTACT_ID>",
14              "Object": "Contact",
15              "ContentType": null,
16              "Content": null
17          },
18          "rows": [
19              {
20                  "margin": false,
21                  "values": [
22                      {
23                          "icon": null,
24                          "text": "<KIX_CONTACT_Firstname> <KIX_CONTACT_Lastname>",
25                          "linkSrc": null
26                      }
27                  ]
28              },
29              {
30                  "margin": false,
31                  "values": [
32                      {
33                          "icon": {
34                              "displayValues": [],
35                              "KIXObjectType": "OBJECT_ICON",
36                              "ObjectId": "<KIX_ORG_ID>",
37                              "Object": "Organisation",
38                              "ContentType": null,
39                              "Content": null
40                          },
41                          "text": "<KIX_ORG_Name>",
42                          "linkSrc": null
43                      }
44                  ]
45              },
46              {
47                  "margin": true,
48                  "values": [
```

```
 49                        {
 50                            "icon": "fas fa-phone-volume",
 51                            "text": "<KIX_CONTACT_Phone>",
 52                            "linkSrc": "tel:<KIX_CONTACT_Phone>"
 53                        }
 54                    ]
 55                },
 56                {
 57                    "margin": false,
 58                    "values": [
 59                        {
 60                            "icon": "kix-icon-mail",
 61                            "text": "<KIX_CONTACT_Email>",
 62                            "linkSrc": null
 63                        }
 64                    ]
 65                },
 66                {
 67                    "margin": true,
 68                    "values": [
 69                        {
 70                            "icon": "kix-icon-compass",
 71                            "text": "<KIX_CONTACT_Street>",
 72                            "linkSrc": "https://www.google.de/maps/place/
    <KIX_CONTACT_Street>+<KIX_CONTACT_Zip>+<KIX_CONTACT_City>"
 73                        }
 74                    ]
 75                },
 76                {
 77                    "margin": false,
 78                    "values": [
 79                        {
 80                            "icon": null,
 81                            "text": "<KIX_CONTACT_Zip> <KIX_CONTACT_City>",
 82                            "linkSrc": "https://www.google.de/maps/place/
    <KIX_CONTACT_Street>+<KIX_CONTACT_Zip>+<KIX_CONTACT_City>"
 83                        }
 84                    ]
 85                },
 86                {
 87                    "margin": false,
 88                    "values": [
 89                        {
 90                            "icon": null,
 91                            "text": "<KIX_CONTACT_Country>",
 92                            "linkSrc": "https://www.google.de/maps/place/
    <KIX_CONTACT_Street>+<KIX_CONTACT_Zip>+<KIX_CONTACT_City>"
 93                        }
 94                    ]
 95                },
 96                {
 97                    "margin": true,
 98                    "values": [
```

```
 99                    {
100                        "icon": "kix-icon-men",
101                        "text": "Show Information in CRM",
102                        "linkSrc": "https://your.crm.example?
       param0=<KIX_CONTACT_Firstname>;param1=<KIX_CONTACT_Laststname>;param2=<KIX
       _CONTACT_Email>",
103                        "conditions": [
104                            {
105                                "property": "<KIX_CONTACT_Comment>",
106                                "operator": "NE",
107                                "value": "",
108                                "useObjectService": null,
109                                "useDisplayValue": null
110                            }
111                        ]
112                    }
113                ]
114            },
115            {
116                "margin": true,
117                "values": [
118                    {
119                        "icon": "kix-icon-fire",
120                        "text": "Ticket is older than 5h.",
121                        "linkSrc": null,
122                        "textStyle": "color:#ff0000",
123                        "conditions": [
124                            {
125                                "property": "Age",
126                                "operator": "GT",
127                                "value": 300,
128                                "useObjectService": false,
129                                "useDisplayValue": false
130                            }
131                        ]
132                    }
133                ]
134            },
135            {
136                "margin": true,
137                "values": [
138                    {
139                        "icon": "kix-icon-men",
140                        "text": "Go to Contact",
141                        "linkSrc": null,
142                        "routingConfiguration": {
143                            "contextId": "contact-details",
144                            "objectType": "Contact",
145                            "contextMode": "DETAILS",
146                            "objectIdProperty": "ID",
147                            "history": false,
148                            "resetContext": true,
149                            "contextType": "MAIN"
```
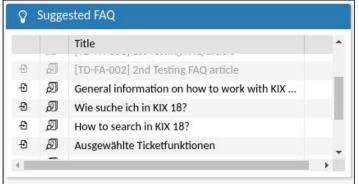
```
150                   },
151                   "routingObjectId": "<KIX_CONTACT_ID>"
152               }
153             ]
154         },
155         {
156             "margin": true,
157             "values": [
158                 {
159                     "icon": "kix-icon-ci",
160                     "text": "Create a new Asset",
161                     "linkSrc": null,
162                     "routingConfiguration": {
163                         "contextId": "new-config-item-dialog-context",
164                         "objectType": "ConfigItem",
165                         "contextMode": "CREATE",
166                         "objectIdProperty": null,
167                         "history": false,
168                         "resetContext": false,
169                         "contextType": "DIALOG",
170                         "resetForm": false
171                     }
172                 }
173             ]
174         }
175     ]
176 },
177 "minimized": false,
178 "minimizable": true,
179 "icon": "kix-icon-man-house",
180 "contextDependent": false,
181 "contextObjectDependent": false,
182 "formDependent": false,
183 "formDependencyProperties": []
184 }
```

13.1.3.4   Configuration of table widget "Suggested FAQ"

Configuration key:	Dialog "New ticket"	ticket-new-dialog-suggested-faq-widget
	Dialog "Edit ticket"	ticket-edit-dialog-suggested-faq-widget
	Ticket details	ticket-details-suggested-faq-widget

Contents on this page:

- Search basics (see page 480)
- Combined or simple search (see page 481)
- Configure the search for the affected assets (see page 482)
- Configure search for keywords (see page 483)

The widget "Suggested FAQ" displays relevant FAQ articles for a ticket. It is an initial component of the sidebars in the ticket details and in the dialogues "New ticket" and "Edit ticket". Agents can view the FAQ articles displayed in the widget in a short preview and transfer them to an article with one click.

The widget "Suggested FAQ" is a table widget. The configuration differs only slightly in the attribute values of `formDependent, formDependencyProperties, contextObjectDependent` in the individual contexts (ticket details, dialogs "New ticket" and "Edit ticket"). An overview of the widget attributes used in the configuration as well as information on the basic configuration of a table widget can be found in the chapter: The table widget (see page 393) .

In the following sections you will find specific configuration options for the widget "Suggested FAQ".

Search basics

The FAQ articles displayed in the table widget are based on a search for the keywords entered in the ticket subject/title and/or the affected assets selected in the ticket. With each change to the ticket subject/title or in the "Affected assets" field, the FAQ will be searched and the table content of the widget will be updated based on the search result.

To search for keywords, a ticket subject/title must be specified in the ticket. A full text search is performed in all FAQ information (keywords, title, number, symptom, cause, solution, etc.). Irrelevant words are filtered out.

> ✅ **Tip**
>
> You can edit the list of irrelevant words in the SysConfig key "*Ticket::SearchIndex::StopWords**"
> separately for each language created in KIX.

To search for the affected assets, at least one affected asset must be selected in the ticket and assets must
be stored at FAQ entries.
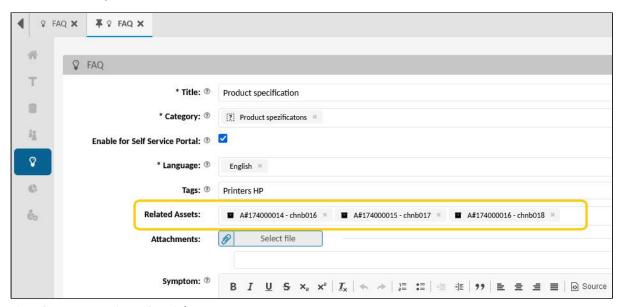
Fig .: Store assets in FAQ articles

Combined or simple search

You can control the search behaviour and thus the result of the search by specifying how the search is
performed and how the results are combined for display in the table. This is done in the respective
SysConfig keys (see table above) by specifying "`intersection : true|false`" in the section
"`Configuration{...}`".

- **true** = intersection
    - Search for keywords in the ticket subject / title **AND** for the assets concerned. Both must be
      specified, otherwise the table is empty.
    - Only FAQ articles are displayed that match at least one of the assets concerned and one of the
      keywords specified in the subject / title.
    - **Important**: Make sure that the search returns at least 1 result (1 FAQ) when you configure the
      search. Otherwise no intersection can be formed and displayed.
- **false** (default) = combine = all results (without duplications)
    - Search for keywords in the ticket subject / title **OR** for the assets concerned. One of the two
      must be specified.

- The FAQ articles are displayed that are linked to at least one of the assets concerned or that contain one of the keywords specified in the ticket subject / title.

```
    cache : true,
    "resetFilterOnReload": true,
    "configuration": {
        "id": "ticket-details-suggested-faq-table-config",
        "name": "Suggested FAQ",
        "type": "Table",
        "objectType": "FAQArticle",
        "loadingOptions": {[ ]},
        "displayLimit": 10,
        "tableColumns": [[ ]],
        "tableColumnConfigurations": null,
        "enableSelection": false,
        "toggle": false,
        "toggleOptions": null,
        "sortOrder": null,
        "headerHeight": 1.75,
        "rowHeight": 1.75,
        "emptyResultHint": "Translatable#0 data sets found.",
        "routingConfiguration": null,
        "fixedFirstColumn": null,
        "additionalTableObjectsHandler": [[ ]],
        "intersection": false,
        "searchId": null,
        "valid": true,
        "application": "agent-portal"
    },
    "valid": true
```

Fig .: Control of the display in the widget with the "intersection" attribute

Configure the search for the affected assets

In the code block " `filter":[{Filter 1}, {Filter 2}, ...]` " it is defined which asset information is taken into account in the search. In the standard configuration, all valid FAQs to which the asset selected in the ticket is assigned as a related asset are loaded into the table.

You can adjust the filter so that different or additional asset information is included in the search. You can enter FAQ attributes and your own dynamic fields. Note that these are not filters that further restrict the result, but rather necessary search criteria.

If no filters are specified, all FAQs are displayed.

Make sure with your configuration that at least 1 FAQ is returned if you have set `"intersection":true` .

Default configuration of asset filter

```
"loadingOptions":{
  "filter": [
    {
      "property": "DynamicFields.RelatedAssets",      //Name of the Dynamic Field
(here: RelatedAssets)
      "operator": "IN",
```

```
        "type": "NUMERIC",
        "filterType": "AND",
        "value": "<KIX_TICKET_DynamicField_AffectedAsset_ObjectValue>"    //
 Placeholder; contains the value of the dynamic field
      },
      {
        "property": "ValidID",
        "operator": "EQ",
        "type": "NUMERIC",
        "filterType": "AND",
        "value": 1
      }
    ],
    "sortOrder": null,
    "limit": 100,      //Limitation of the search result
    "includes": [
      "Votes"
    ]
  },
```

Configure search for keywords

You can adjust the configuration of the object handlers to control the search for the keywords specified in the ticket subject / title.

The code section `additionalTableObjectsHandler:[...]` contains a list of the handler configurations that determine further objects for the table. The determination depends on the handler. The following attributes are possible:

Attribute	Description
id	Unique identifier for internal processing.
name	Name (description)
handlerId	Specifies which handler should be used. Currently only "SuggestedFAQHandler"
handlerConfiguration	specific options of the handler. The SuggesteFAQHandler currently supports: • **minLenght**: Indicates how long the words in the title / subject must be at least (separated by spaces) so that they can be used for the search (default: 3). • **onlyValid**: Indicates whether only valid objects are determined (default: false).

Attribute	Description
dependency Properties	List of object properties (attributes) which are taken into account for the search. The information is relevant for the handler. The SuggestedFAQHandler uses it to determine the properties of the words. The list also determines whether this configuration makes it necessary to reload the table in the event of changes in the dialog / form. It is used to avoid unnecessary searches for the handlers involved. If this list is not available, the table is reloaded with each change.

Default configuration of object handler

```
"additionalTableObjectsHandler": [
  {
    "id": "ticket-new-suggested-faq-handler-config",
    "name": "FAQs by ticket title or subject",
    "handlerId": "SuggestedFAQHandler",
    "handlerConfiguration": {
    "minLength": 3,
    "onlyValid": true
    },
    "dependencyProperties": [
      "Title",
      "Subject"
    ],
    "type": "AdditionalTableObjectsHandler"
  }
],
```

ⓘ **Info**

Optionally, the configuration of the object handler can be removed if the search should only be for the affected assets (only useful if `"intersection":false`).

13.1.3.5   Adding a widget to the dashboard

You can add custom widgets to the home dashboard. By personalizing the dashboard, agents can define which widgets should be displayed in their personal dashboard.

Fig .: Selection of the (table) widgets to be displayed in the home dashboard

How to integrate another widget into the home dashboard:

1. Navigate to the menu *System> SysConfig* open the key "home"
2. For better editing, copy the content of the "Value" field into a JSON editor (e.g. https://www.jsonformatter.io)
3. Paste one of the following code blocks (without comments) into the editor. Place it, for example, between 2 existing widgets. The insertion position defines where in the dashboard the widget is displayed by default. The position can be changed by the agent in the course of personalizing the dashboard.
4. Adapt the examples to your needs by for example
   1. change the ID and name of the Object-information-card-widget according to the widget to be displayed
   2. adapt the ID, name and parameters of the table configuration according to your needs.
5. Minimize the source code and copy it back into the "Value" field of the configuration key and save your changes.
6. In the configuration key overview, click on "Reload frontend configuration".

The widget is now included in the dashboard. Agents may need to refresh their browser display.

Fig.: A table widget integrated into the dashboard

Example code block for inserting an object card widget

```
{
  "instanceId": "name-of-info-widget",
  "configurationId": "name-of-info-widget",
  "configuration": null,
  "permissions": [
    {
      "target": "tickets",
      "permissions": [
        2
      ],
      "OR": false,
      "value": 2
    }
  ],
  "size": "small"
},
```

Sample code block for inserting a table widgets

```
{ //Create an instance of your own table widget
  "instanceId": "home-dashboard-custom-ticket-list",
  "configurationId": null,
  "configuration": {
    "id": "home-dashboard-custom-ticket-list",
    "name": "Custom Ticket List",
    "type": "Widget",
    "widgetId": "table-widget",
```

```
    "title": "Custom Ticket List",
    "actions": [
        "bulk-action",
        "csv-export-action"
    ],
    "subConfigurationDefinition": null,
    "configuration": {
        "id": "home-dashboard-ticket-custom-ticket-table-widget",
        "name": "Custom Tickets Table Widget",
        "type": "TableWidget",
        "objectType": "Ticket",
        "sort": [
            "TicketNumber",
            "Up"
        ],
        "subConfigurationDefinition": null,
        //tableConfiguration
        "tableConfiguration": {
            "id": "home-dashboard-ticket-table-new",
            "name": "New Tickets Table",
            "type": "Table",
            "objectType": "Ticket",
            //Definition of the content to be loaded
            "loadingOptions": {
                "filter": [
                    {
                        "property": "StateType",
                        "operator": "EQ",
                        "type": "String",
                        "filterType": "OR",
                        "value": "Open"
                    }
                ],
                "limit": 30,
                "includes": [
                    "Watchers",
                    "DynamicFields"
                ]
            },
            "displayLimit": 10,
            //Definition of the table columns
            "tableColumns": [
                {
                    "id": null,
                    "name": null,
                    "type": null,
                    "property": "PriorityID",
                    "showText": false,
                    "showIcon": true,
                    "showColumnTitle": true,
                    "showColumnIcon": false,
                    "size": 65,
```

```
        "sortable": true,
        "filterable": true,
        "hasListFilter": true,
        "dataType": "STRING",
        "resizable": true,
        "componentId": null,
        "defaultText": null,
        "translatable": true,
        "titleTranslatable": true,
        "useObjectServiceForFilter": false
    },
    {
        "id": null,
        "name": null,
        "type": null,
        "property": "TicketNumber",
        "showText": true,
        "showIcon": false,
        "showColumnTitle": true,
        "showColumnIcon": true,
        "size": 135,
        "sortable": true,
        "filterable": true,
        "hasListFilter": false,
        "dataType": "STRING",
        "resizable": true,
        "componentId": null,
        "defaultText": null,
        "translatable": true,
        "titleTranslatable": true,
        "useObjectServiceForFilter": false
    },
    {
        "id": null,
        "name": null,
        "type": null,
        "property": "Title",
        "showText": true,
        "showIcon": false,
        "showColumnTitle": true,
        "showColumnIcon": true,
        "size": 463,
        "sortable": true,
        "filterable": true,
        "hasListFilter": false,
        "dataType": "STRING",
        "resizable": true,
        "componentId": null,
        "defaultText": null,
        "translatable": true,
        "titleTranslatable": true,
        "useObjectServiceForFilter": false
```

```
            },
            {
                "id": null,
                "name": null,
                "type": null,
                "property": "QueueID",
                "showText": true,
                "showIcon": false,
                "showColumnTitle": true,
                "showColumnIcon": true,
                "size": 175,
                "sortable": true,
                "filterable": true,
                "hasListFilter": true,
                "dataType": "STRING",
                "resizable": true,
                "componentId": null,
                "defaultText": null,
                "translatable": true,
                "titleTranslatable": true,
                "useObjectServiceForFilter": false
            },
            {
                "id": null,
                "name": null,
                "type": null,
                "property": "DynamicFields.AffectedAsset",
                "showText": true,
                "showIcon": false,
                "showColumnTitle": true,
                "showColumnIcon": false,
                "size": 200,
                "sortable": true,
                "filterable": true,
                "hasListFilter": true,
                "dataType": "STRING",
                "resizable": true,
                "componentId": "label-list-cell-content",
                "defaultText": null,
                "translatable": true,
                "titleTranslatable": true,
                "useObjectServiceForFilter": false
            }
        ],
        "tableColumnConfigurations": null,
        "enableSelection": true,
        "toggle": true,
        "toggleOptions": {
            "componentId": "ticket-article-details",
            "inputPropertyName": "article",
            "actions": [],
            "toggleFirst": true,
```

```
                "data": {},
                "toggleAll": false
            },
            "headerHeight": 2.25,
            "rowHeight": 1.75,
            "emptyResultHint": "Translatable#No tickets available.",
            "fixedFirstColumn": false
        },
        "headerComponents": null,
        "showFilter": true,
        "shortTable": false,
        "predefinedTableFilters": [],
        "cache": false,
        "resetFilterOnReload": true
    },
    "minimized": false,
    "minimizable": true,
    "icon": "kix-icon-ticket",
    "contextDependent": false
},
"permissions": [
    {
        "target": "tickets",
        "permissions": [
            2
        ],
        "OR": false,
        "value": 2
    }
],
"size": "large"
}
```

Complete code example: Home dashboard with additional table widget "home-dashboard-custom-ticket-list"

```
{
    "id": "home",
    "name": "home",
    "type": "Context",
    "contextId": "home",
    "sidebars": [
        {
            "instanceId": "home-dashboard-notes-widget",
            "configurationId": "home-dashboard-notes-widget",
            "permissions": [],
            "size": "large"
        }
```

```
        ],
    "explorer": [],
    "lanes": [],
    "content": [
        {
            "instanceId": "home-dashboard-ticket-chart-widget-priorities",
            "configurationId": "home-dashboard-ticket-chart-widget-priorities",
            "configuration": null,
            "permissions": [
                {
                    "target": "tickets",
                    "permissions": [
                        2
                    ],
                    "OR": false,
                    "value": 2
                }
            ],
            "size": "small"
        },
        {
            "instanceId": "home-dashboard-ticket-chart-widget-states",
            "configurationId": "home-dashboard-ticket-chart-widget-states",
            "configuration": null,
            "permissions": [
                {
                    "target": "tickets",
                    "permissions": [
                        2
                    ],
                    "OR": false,
                    "value": 2
                }
            ],
            "size": "small"
        },
        {
            "instanceId": "home-dashboard-ticket-chart-widget-new",
            "configurationId": "home-dashboard-ticket-chart-widget-new",
            "configuration": null,
            "permissions": [
                {
                    "target": "tickets",
                    "permissions": [
                        2
                    ],
                    "OR": false,
                    "value": 2
                }
            ],
            "size": "small"
        },
```

```
{  //eigenes Tabellenwidget
    "instanceId": "home-dashboard-custom-ticket-list",
    "configurationId": null,
    "configuration": {
        "id": "home-dashboard-custom-ticket-list",
        "name": "Custom Ticket List",
        "type": "Widget",
        "widgetId": "table-widget",
        "title": "Custom Ticket List",
        "actions": [
            "bulk-action",
            "csv-export-action"
        ],
        "subConfigurationDefinition": null,
        "configuration": {
            "id": "home-dashboard-ticket-custom-ticket-table-widget",
            "name": "Custom Tickets Table Widget",
            "type": "TableWidget",
            "objectType": "Ticket",
            "sort": [
                "TicketNumber",
                "Up"
            ],
            "subConfigurationDefinition": null,
            "tableConfiguration": {
                "id": "home-dashboard-ticket-table-new",
                "name": "New Tickets Table",
                "type": "Table",
                "objectType": "Ticket",
                "loadingOptions": {
                    "filter": [
                        {
                            "property": "StateType",
                            "operator": "EQ",
                            "type": "String",
                            "filterType": "OR",
                            "value": "Open"
                        }
                    ],
                    "limit": 30,
                    "includes": [
                        "Watchers",
                        "DynamicFields"
                    ]
                },
                "displayLimit": 10,
                "tableColumns": [
                    {
                        "id": null,
                        "name": null,
                        "type": null,
                        "property": "PriorityID",
```

```
                        "showText": false,
                        "showIcon": true,
                        "showColumnTitle": true,
                        "showColumnIcon": false,
                        "size": 65,
                        "sortable": true,
                        "filterable": true,
                        "hasListFilter": true,
                        "dataType": "STRING",
                        "resizable": true,
                        "componentId": null,
                        "defaultText": null,
                        "translatable": true,
                        "titleTranslatable": true,
                        "useObjectServiceForFilter": false
                    },
                    {
                        "id": null,
                        "name": null,
                        "type": null,
                        "property": "TicketNumber",
                        "showText": true,
                        "showIcon": false,
                        "showColumnTitle": true,
                        "showColumnIcon": true,
                        "size": 135,
                        "sortable": true,
                        "filterable": true,
                        "hasListFilter": false,
                        "dataType": "STRING",
                        "resizable": true,
                        "componentId": null,
                        "defaultText": null,
                        "translatable": true,
                        "titleTranslatable": true,
                        "useObjectServiceForFilter": false
                    },
                    {
                        "id": null,
                        "name": null,
                        "type": null,
                        "property": "Title",
                        "showText": true,
                        "showIcon": false,
                        "showColumnTitle": true,
                        "showColumnIcon": true,
                        "size": 463,
                        "sortable": true,
                        "filterable": true,
                        "hasListFilter": false,
                        "dataType": "STRING",
                        "resizable": true,
```

```
                    "componentId": null,
                    "defaultText": null,
                    "translatable": true,
                    "titleTranslatable": true,
                    "useObjectServiceForFilter": false
                },
                {
                    "id": null,
                    "name": null,
                    "type": null,
                    "property": "QueueID",
                    "showText": true,
                    "showIcon": false,
                    "showColumnTitle": true,
                    "showColumnIcon": true,
                    "size": 175,
                    "sortable": true,
                    "filterable": true,
                    "hasListFilter": true,
                    "dataType": "STRING",
                    "resizable": true,
                    "componentId": null,
                    "defaultText": null,
                    "translatable": true,
                    "titleTranslatable": true,
                    "useObjectServiceForFilter": false
                },
                {
                    "id": null,
                    "name": null,
                    "type": null,
                    "property": "DynamicFields.AffectedAsset",
                    "showText": true,
                    "showIcon": false,
                    "showColumnTitle": true,
                    "showColumnIcon": false,
                    "size": 200,
                    "sortable": true,
                    "filterable": true,
                    "hasListFilter": true,
                    "dataType": "STRING",
                    "resizable": true,
                    "componentId": "label-list-cell-content",
                    "defaultText": null,
                    "translatable": true,
                    "titleTranslatable": true,
                    "useObjectServiceForFilter": false
                }
            ],
            "tableColumnConfigurations": null,
            "enableSelection": true,
            "toggle": true,
```

```
                "toggleOptions": {
                    "componentId": "ticket-article-details",
                    "inputPropertyName": "article",
                    "actions": [],
                    "toggleFirst": true,
                    "data": {},
                    "toggleAll": false
                },
                "headerHeight": 2.25,
                "rowHeight": 1.75,
                "emptyResultHint": "Translatable#No tickets available.",
                "fixedFirstColumn": false
            },
            "headerComponents": null,
            "showFilter": true,
            "shortTable": false,
            "predefinedTableFilters": [],
            "cache": false,
            "resetFilterOnReload": true
        },
        "minimized": false,
        "minimizable": true,
        "icon": "kix-icon-ticket",
        "contextDependent": false
    },
    "permissions": [
        {
            "target": "tickets",
            "permissions": [
                2
            ],
            "OR": false,
            "value": 2
        }
    ],
    "size": "large"
},
{
    "instanceId": "home-dashboard-myOpenTickets-widget",
    "configurationId": "home-dashboard-myOpenTickets-widget",
    "configuration": null,
    "permissions": [
        {
            "target": "tickets",
            "permissions": [
                2
            ],
            "OR": false,
            "value": 2
        }
    ],
    "size": "large"
```

```
        },
        {
            "instanceId": "home-dashboard-new-tickets-widget",
            "configurationId": "home-dashboard-new-tickets-widget",
            "configuration": null,
            "permissions": [
                {
                    "target": "tickets",
                    "permissions": [
                        2
                    ],
                    "OR": false,
                    "value": 2
                }
            ],
            "size": "large"
        }
    ],
    "generalActions": [],
    "actions": [],
    "overlays": [],
    "others": [],
    "dialogs": [],
    "customizable": true
}
```

13.1.3.6 Configuration of dashboard tables

Configuration key:	home-dashboard-ticket-table-new

You can customise the tables in the home or ticket dashboard as required. It is possible to insert additional table columns or to remove unneeded table columns. The adjustment of the tables is done via the configuration keys in the menu *System > SysConfig*. In the following example, the table "New tickets" in the home dashboard is extended by the table column "Contact", in which the name of the customer contact is displayed. For this, the configuration key "home-dashboard-ticket-table-new" is adapted accordingly. If you want to configure a different table, please search for the corresponding key in the SysConfig menu and change it.

This is how you add a table column:

1. Navigate to *System > SysConfig*. Locate the key "home-dashboard-ticket-table-new". This key defines the structure of the table.

2. Open the key and copy the source code from the "Value" field into a JSON editor. The right column contains the source code in maximised view for easier editing.

3. Insert the following code block (without comments!) into the source code on the right-hand side of the editor. Place it at the desired position between two already existing table columns. A table column usually starts and ends with `{"id":[...] "useObjectServiceForFilter": true | false}`.

   Configure the inserted code block as required:
   - Enter the ticket attribute to be displayed in the table column under "property". Here the attribute "ContactID" for the customer contact. You can find an overview of the ticket attributes in the appendix (see page 875) .
   - If you want to include a Dynamic Field, use the syntax: "`DynamicFields.FieldName`". The name defined under *System > Dynamic Fields* must be specified.
   - If you want to display the progress bar of a dynamic field of the type "Checklist" of the table column, enter the value "`dynamic-field-checklist-cell`" under "`componentID`".
   - If you do not know the data type of the attribute to be included, note the value "null" under "`dataType`".

- For an overview of the attributes used, see: The table widge t.

```
{
    "id": null,
    "name": null,
    "type": null,
    "property": "ContactID",    //ticket attribute to be displayed in the
column
    "showText": true,
    "showIcon": false,
    "showColumnTitle": true,
    "showColumnIcon": true,
    "size": 225,
    "sortable": true,
    "filterable": true,
    "hasListFilter": false,
    "dataType": "STRING",
    "resizable": true,
    "componentId": null,        //or value for a progress bar: "dynamic-
field-checklist-cell
    "defaultText": null,
    "translatable": true,
    "titleTranslatable": true,
    "useObjectServiceForFilter": false
},
```
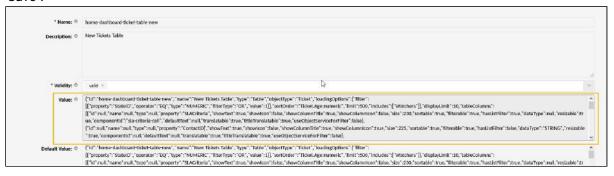

4. Minimise the source code to remove unnecessary spaces and line breaks and copy the source code to the clipboard.

5. Paste the source code from the clipboard back into the "Value" field and apply the changes with "Save".

6. Click on "Load Frontend Configuration" to update the table in the Home Dashboard. The agents must also reload their frontend.

7. The name of the customer contact is now displayed in the dashboard table.

This is how you remove a table column:

1. Navigate to *System > SysConfig*.
2. Search for the configuration key of the table you want to edit, for example, "home-dashboard-ticket-table-new".
    You can use the asterisk (*) as a wildcard for the search.
3. Open the key and copy the source code from the Value field into a JSON editor. The right column contains the source code in maximized view for easier editing.

4. Remove the unneeded table column from the source code. A table column usually starts and ends with `{"id":[...] "useObjectServiceForFilter": true | false}.`

5. Minimize the source code to remove unnecessary spaces and line breaks and copy the source code to the clipboard.

6. Paste the source code from the clipboard back into the "Value" field and apply the changes by clicking "Save".

7. Click "Load Frontend Configuration" to update the table in the home dashboard. Agents will need to reload their frontend as well.

8. The column is now removed from the dashboard table.

13.1.3.7 Adding a Widget to the Sidebar
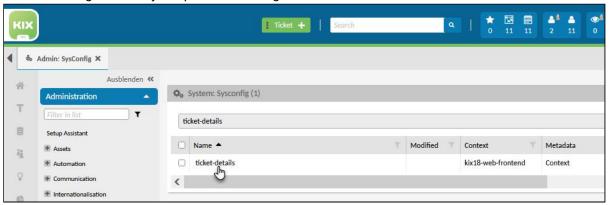
The following description relates to the configuration of the ticket details sidebar. You can do the same for all other sidebars.

Your own widgets must be implemented directly in the sidebar. Proceed as follows:

1. Navigate to the *System > SysConfig* menu and look for the desired configuration key (here: ticket-details)
   You can use the asterisk (*) as a wildcard for your search.

2. Click the configuration key to open it for editing.

3. For better editing, copy the content of the "Value" field into a JSON editor (e.g. https://www.jsonformatter.io)

4. Add the following source code (without comments) within the `"sidebars":[...]` section. It forms the basic framework for a card widget. To integrate a table widget, you can use the example configuration in the chapter Configuration of table widget "Suggested FAQ" (see page 480) as a guide. Place the source code between 2 existing widgets or at the beginning or end of the section. You use this to determine where the widget is displayed in the sidebar.

```
{    //Create an instance of your own widget
    "instanceId": "ticket-details-my-own-widget",
    //Configuration of your own widget
    "configuration": {
        "id": "ticket-details-my-own-widget",
        "name": "My Own Widget",
        "type": "Widget",
        "widgetId": "object-information-card-widget",
        "title": "Translatable#My Own Widget",
        "actions": [],
        "subConfigurationDefinition": null,
        //Configuration of the widget content
        "configuration": {
            //Configuration of the avatar
            "avatar": {
                "displayValues": [],
                "KIXObjectType": "OBJECT_ICON",
                "ObjectId": "<KIX_CONTACT_ID>",
                "Object": "CONTACT",
                "ContentType": null,
                "Content": null
            },
            //Content to be displayed line by line in the widget
            "rows": [
                //line 1
                {
                    "margin": false,
```

```
                    "values": [
                        {
                            "icon": null,
                            "text": "<KIX_CONTACT_ID>",
                            "linkSrc": null
                        }
                    ]
                },
                //line 2
                {
                    "margin": true,
                    "values": [
                        {
                            "icon": "kix-icon-handshake",
                            "text":
"Description:<KIX_CONTACT_Firstname><KIX_CONTACT_Lastname>",
                            "linkSrc": null
                        }
                    ]
                }
            ]
        },
        //Behavior of the widget and widget title
        "minimized": false,
        "minimizable": true,
        "icon": "kix-icon-man-house",
        "contextDependent": false
    },
    "permissions": [],
    "size": "large"
},
```

```
{
    "id": "ticket-details",
    "name": "Ticket Details",
    "type": "Context",
    "contextId": "ticket-details",
    "sidebars": [
        { //Configuration of the widget "Contact Information"
            "instanceId": "ticket-details-contact-card-widget",
            "configurationId": "ticket-details-contact-card-widget",
            "permissions": [],
            "size": "large"
        },
        { //Instance of your own widget
            "instanceId": "ticket-details-my-own-widget",
            //Configuration of your own widget
            "configuration": {
                "id": "ticket-details-my-own-widget",
                "name": "My Own Widget",
                "type": "Widget",
                "widgetId": "object-information-card-widget",
                "title": "Translatable#My Own Widget",
                "actions": [],
                "subConfigurationDefinition": null,
                //Configuration of the widget content
                "configuration": {
                    //Configuration of the avatar
                    "avatar": {
                        "displayValues": [],
                        "KIXObjectType": "OBJECT_ICON",
                        "ObjectId": "<KIX_CONTACT_ID>",
                        "Object": "CONTACT",
                        "ContentType": null,
                        "Content": null
                    },
                    //Content to be displayed line by line in the widget
                    "rows": [
                        //line 1
                        {
                            "margin": false,
                            "values": [
                                {
                                    "icon": null,
                                    "text": "<KIX_CONTACT_ID>",
                                    "linkSrc": null
                                }
                            ]
                        },
                        //line 2
                        {
                            "margin": true,
                            "values": [
                                {
                                    "icon": "kix-icon-handshake",
                                    "text": "Description:<KIX_CONTACT_Firstname><KIX_CONTACT_Lastname>",
                                    "linkSrc": null
                                }
                            ]
                        }
                    ]
                },
                //Behavior of the widget and widget title
                "minimized": false,
                "minimizable": true,
                "icon": "kix-icon-man-house",
                "contextDependent": false
            },
            "permissions": [],
            "size": "large"
        },
        {
            "instanceId": "ticket-details-suggested-faq-widget"
```

5. Remove the comments in the source code (// comment text) and configure the widget content. You can also read about this in the chapter Configuring an object information card widget (see page 473) . You can find an overview of the attributes that can be used in the chapter Configuration of Sidebar (see page 412) .

6. Minimize the source code and copy it back into the "Value" field of the configuration key and save your changes.
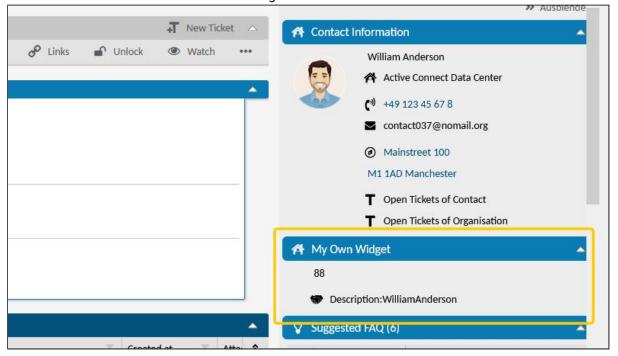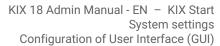
7. In the configuration key overview, click on "Reload frontend configuration".

8. The sidebar now contains the additional widget.

> ⓘ **Information**
>
> You may have to update the translation for the widget title or for individual elements in the widget under *Internationalization> Translations*.

13.2   Settings for tickets

The following chapters contain advanced configuration options for the views in the ticket and the ticket dashboard.

- Adapting the Kanban Board (see page 508)
- Calendar configuration (see page 513)
- Changing the display of the article list (see page 518)
- Configuration PDF printing (see page 523)
- Display asset attributes in ticket zoom view (see page 559)
- Generate ticket numbers (see page 563)

13.2.1 Adapting the Kanban Board

Configuration key:	kanban-main-widget

The Kanban Board gives agents a structured overview of their upcoming tasks. Agents can call up the Board via the toolbar.

The tickets can be dragged and dropped within the Kanban Board, and doing so automatically changes the owner and state:

Column	Contains tickets...
Team Backlog	that have not been assigned to any owner and have the state "New".
Personal Backlog	where the logged-in user is the agent, and that have the state "New".
Work in Progress	where the logged-in user is the owner, and that have the state "Open".
Pending	where the logged-in user is the owner, and that have the state "Pending".
Recently Closed	where the logged-in user is the owner, and that have the state"Closed". This column only shows closed tickets from the last 24 hours.

You can use the GUI configuration to tailor the display of cards in the Kanban Board to suit your individual preferences. This may be required if, for example, the values of dynamic fields should be displayed in the details, or if tickets are moved and should therefore receive a different (user-defined) state than their initial one.

The description below is a continuation of the example from chapter "Integrating a Dynamic Field ". In this example, we now need to display the serial number in the Kanban Board. To do this, you must already have created a dynamic field for the serial number and integrated it into the ticket creation screen so that agents can save the serial number in the ticket.
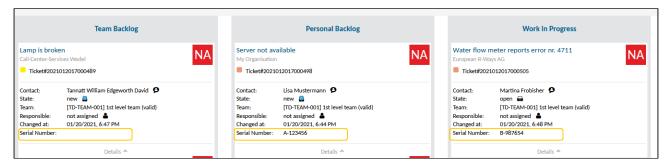
Fig.: Value of a Dynamic Field in Kanban Board

To display the values of a dynamic field in the Kanban Board, proceed as follows:

1. Navigate to *System > SysConfig* and search for key *"kanban-main-widget".* This key defines the structure of the ticket details for the Kanban Board.

2. Open key and copy source code from "Value" field to JSON editor. To make it easier to edit the code, the right-hand column contains the source code in maximized format.

3. In block of code "cardProperties", enter reference and name of dynamic field at appropriate point. The sequence in the block of code determines the sequence in which the fields are displayed in the card. Use the following syntax: " `DynamicFields.NameOfYourDynamicField` ".

4. Optionally, change state. The IDs in the block of code "columns" define the individual columns shown in the Kanban Board. The line "dropState" defines which state is assigned to a ticket when it is moved into the column concerned. You can also change the names of the IDs. However, to do this you may need to maintain the translation (menu *Internationalisation > Translations*).

5. Minify source code to remove superfluous blanks and line breaks and copy source code to clipboard.

6. Paste source code from clipboard back into "Value" field and click "Save" to apply changes.

7. Click "Reload Frontend Configurations" to update display in Kanban Board. If you do not do this, the dynamic field will not be displayed.

8. The serial number saved in the ticket will now also be displayed in the details.

13.2.2 Calendar configuration

SysConfig key	*calendar*

Calendars are used in KIX 18 to calculate working times and to determine Begin of Business Day (BOB) and End of Business Day (EOB). In KIX Pro, they are also used to determine SLA service and escalation times.

Initially, 9 calendars per calendar type are delivered. Each of these calendars can be activated as required and the times and dates individually adapted. For example, for internationally active companies with several subsidiaries, a different calendar can apply depending on the localisation.

The calendars are configured in the System > SysConfig menu. Initially, 9 calendars per calendar type are delivered (designation: Calendar1-9). The SysConfig keys for each calendar can be found in the following table.

Note for KIX Pro: If no calendar is stored at the SLA, the default values of the system are used. The SysConfig keys for this can be found in the table below.

13.2.2.1 Calendar overview

Calendar type/ SysConfig key	Description	Notes	Sample value

TimeVacationDays:: <CalendarNr>	Defines recurring free days for the corresponding calendar (Public holidays)	Initial active
		
		Initially, all calendars are delivered with the uniform public holidays valid in Germany and Austria:
		
		• New Years Eve
		• New Year
		• Labour Day
		• Christmas Eve
		• Christmas Day
		• St. Stephans Day
			[{
			  "Day":"1",
			  "Month":"1",
			  "Translatable":"1",
			  "content":"New Year's Day"
			},
			{
			  "Day":"24",
			  "Month":"12",
			  "Translatable":"1",
			
			"content":"Christmas Eve"
			},
			{
			  "Day":"25",
			  "Month":"12",
			  "Translatable":"1",
			  "content":"First Christmas Day"
			}]

Calendar type/ SysConfig key	Description	Notes	Sample value
TimeVacationDays Modules::<Calendar Nr>	Floating holiday	Floating holidays are automatically determined for each calendar year and considered according to the working time configured for that day. Usually no working time for the whole day. An activated public holiday is taken into account for the respective calendar in all relevant calculations, especially escalations. That means, tickets do not escalate on holidays, SLA times are suspended). The following floating holidays are active (valid) • Easter Monday • Ascension Day • Whit Monday The following floating holidays are initially inactive (invalid). They can be activated if required. • Corpus Christi • Maundy Thursday • Good Friday • Day of Repentance and Prayer	**Please do not change the value!**

Calendar type/ SysConfig key	Description	Notes	Sample value
TimeVacationDaysOneTime::<CalendarNr>	Defines one-off days off for the calendar.	Can be used if a shutdown is set for a certain day (e.g. bridge day).	[{ "Day":"1", "Month":"1", "Year":"2004", "content":"test" }]

TimeWorkingHours:: <CalendarNr>	Specifies the daily working hours for the calendar.	Days set as holidays can also be referenced here using the designation entered under "content". Entry is possible to the minute. Several time intervals per day can be specified, separated by commas, in order to store fixed break times.	Example for Calendar 1:
			{
			"Fri":"8:00-21:00",
			
			"Mon":"8:00-21:00",
			"Thu":"8:00-21:00",
			"Tue":"8:00-21:00",
			"Wed":"8:00-21:00"
			}

TimeZone::<CalendarNr>	Specifies the time zone for the calendar.		local
TimeZone::<CalendarNr>Name	Specifies the name for the calendar.		Calendar Name <Nr.>

13.2.2.2 Default values (system-wide)

Calendar type / SysConfig key	Description
TimeVacation Days	Defines recurring days off system-wide. (Public holidays)

Calendar type / SysConfig key	Description
TimeVacation DaysOneTime	Defines one-time free days system-wide.
TimeWorkingH ours	Defines the days and times that count as business hours. Specifies the daily working hours for the calendar. Days set as holidays can also be referenced here using the designation entered under "content". Entry is possible to the minute. Several time intervals per day can be specified, separated by commas, in order to store fixed break times. **Note:** The keys for business time are considered in the following order: TimeVacationDaysOneTime > TimeVacationDays > Day of Week If a day is stored as a "public holiday" and there is no time entry for this day, no business hours are considered for this day.
TimeZone	Specifies the time zone system-wide.

13.2.3 Changing the display of the article list

13.2.3.1 Changing the colour display in the thread

Configuration key	article-colors-configuration

The overview of the articles in the ticket details is displayed as a thread. The colour representation of the articles depends on the channel used and the channel type. You can change the colours:

- In KIX Start, navigate to *System > SysConfig*.
- Open the SysConfig key "*article-colors-configuration*".
- Replace the hexadecimal values with your own hexadecimal values and save your changes.

- Click on "Reload frontend configurations" to update the frontend.

Initially, the articles are colour-coded as follows:

Color	Hex	Channel	Sender type
light yellow	#fbf7e2	Notice	all
light blue	#e1eaeb	Email	internal (Agent and System)
white	Not configurable	Email	external

> ✅ **Tipp**
>
> The order of items can be set by the agent in their Personal Settings (oldest or newest first).

13.2.3.2 Display of the articles as a list

Configuration key	ticket-details

If you prefer the display of the items as a list with expandable list elements, you can change this in the SysConfig key "*ticket-details*". This will restore the view of the item list before the v23 release.

Proceed as follows:

- In KIX Start, navigate to *System > SysConfig.*
- Open the SysConfig key "*ticket-details*" and copy the value of the key into an external JSON editor.
- Navigate to the section `content{...}`

- Replace the `content{...}` section with code block 1 or 2:
  Note: Codeblock 1 is only supported by older systems that still contain the SysConfig key "*ticket-details-article-list-widget*".
  New installations do not have this configuration. However, you can configure your own table widget for articles. Use code block 2 as a basis.

Code block 1 - Artikel list (old)

```
{
  "instanceId": "ticket-details-article-list-widget",
  "configurationId": "ticket-details-article-list-widget",
  "configuration": null,
```

```
    "permissions": [],
    "size": "large"
}
```

Code block 2 - Table widget article list

```
"content": [
  {
  "instanceId": "ticket-details-article-list-widget",
  "configurationId": null,
  "configuration": {
    "id": "ticket-details-article-list-widget",
    "name": "Article List Widget",
    "type": "Widget",
    "widgetId": "table-widget",
    "title": "Translatable#Article Overview",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
      "id": "ticket-details-article-list-table-config",
      "name": "Article Table",
      "type": "TableWidget",
      "objectType": "Article",
      "tableConfiguration": {
        "id": "ticket-details-article-list-table",
        "name": "Article Table",
        "type": "Table",
        "objectType": "Article",
        "loadingOptions": null,
        "displayLimit": 10,
        "tableColumns": null,
        "tableColumnConfigurations": null,
        "enableSelection": null,
        "toggle": true,
        "toggleOptions": {
          "componentId": "ticket-article-details",
          "inputPropertyName": "article",
          "actions": [
            "article-reply-action",
            "article-forward-action",
            "article-get-plain-action"
          ],
          "toggleFirst": true,
          "data": null,
          "rowObjectProperty": null,
          "toggleAll": true
        },
        "headerHeight": 2.25,
        "rowHeight": 1.75,
```

```
        "emptyResultHint": "Translatable#0 data sets found.",
        "fixedFirstColumn": false,
        "additionalTableObjectsHandler": [],
        "intersection": true,
        "searchId": null
      },
      "headerComponents": [
        "article-attachment-count"
      ],
      "showFilter": true,
      "shortTable": false,
      "predefinedTableFilters": [],
      "cache": false,
      "resetFilterOnReload": true,
      "configuration": {
        "id": "ticket-details-article-list-table",
        "name": "Article Table",
        "type": "Table",
        "objectType": "Article",
        "loadingOptions": null,
        "displayLimit": 10,
        "tableColumns": null,
        "tableColumnConfigurations": null,
        "enableSelection": null,
        "toggle": true,
        "toggleOptions": {
          "componentId": "ticket-article-details",
          "inputPropertyName": "article",
          "actions": [
            "article-reply-action",
            "article-forward-action",
            "article-get-plain-action"
          ],
          "toggleFirst": true,
          "data": null,
          "rowObjectProperty": null,
          "toggleAll": true
        },
        "headerHeight": 2.25,
        "rowHeight": 1.75,
        "emptyResultHint": "Translatable#0 data sets found.",
        "fixedFirstColumn": false,
        "additionalTableObjectsHandler": [],
        "intersection": true,
        "searchId": null
      }
    },
    "minimized": false,
    "minimizable": true,
    "icon": null,
    "contextDependent": true,
    "contextObjectDependent": false,
```

```
        "formDependent": false,
        "formDependencyProperties": []
    },
    "permissions": [],
    "size": "large"
  }
],
```

- Minimise the code in the external JSON editor and copy it back to the SysConfig key value.
- Save your changes.
- Click on "Reload frontend configurations" to update the frontend.

> ✅ **Tip**
>
> KIX Pro has an integrated JSON editor (menu *System > GUI Configuration > Agent Portal*). This can be used to conveniently change the key. Copying the value into an external editor can thus be omitted.

13.2.4 Configuration PDF printing

You have the option to output ticket and article information in a PDF file. For example, to store the PDF file in the company's internal file or document management system or to forward it by e-mail.

The creation of a PDF can be done via the actions "Print" available at the ticket and the article, the macro action "Convert to PDF" or via the console command of the KIX Console. You can give the macro action and the console commands various parameters and thus modify the PDF output. The following description is intended to illustrate the functional principle of PDF printing and to provide assistance in configuring the macro action "Convert to PDF".

13.2.4.1 HTML to PDF

HTMLToPDF is a converter that can convert HTML to PDF. It enables the output of ticket and article information in a PDF file. The converter is already included in KIX.

To create the PDF, KIX needs a specific template in JSON format. KIX uses this JSON template to generate the HTML on the basis of which the converter creates the PDF.

KIX initially contains 2 templates: one for ticket printing and one for article printing. The structure of the respective PDF is defined in the templates. Here it is specified how the data of the connected object (ticket, article) is to be prepared within an HTML structure. In addition to the prepared data of the object, placeholders can also be applied that are related to the object.

In addition, data from other objects can be used, as long as the main object provides the necessary basic data. This means, for example, that the object must provide IDs so that the other object can prepare its data. Without this data, no HTML can be generated for the respective object.

> ⓘ **Note**
>
> Currently, no further objects or own templates can be integrated into the system. Changes to the default templates are currently only possible in the database (on-premises).

13.2.4.2   The structure of the template

The structure defines how the HTML is structured and filled with the data of the object. It forms the basis for what information is contained in the PDF and where in the PDF it is output. The structure must be in JSON format so that KIX can generate the HTML from it.

In general, the structure consists of 3 main containers: Header (H), Content (C) and Footer (F). The containers can be structured separately in each template and filled with data from the object.

Header and Footer (blue) can be set optionally. They form the headers and footers of a page. If the header and footer are set, they are present on every page. The content (green) contains the page content. All object information is listed here in the PDF - as defined in the structure.

Variants of the container structure:

H+C+F	C+F	H+C	C

Example of a container structure with header, footer and content

```
[...]
'Definition' => {
  'Content' => [... ],
  'Expands' => [...],
  'Footer' => [...],
  'Header' => [...],
  'Page' => {...}
},
```

```
[...]
```

The content of a container is formed by the structure types specified in it (see below). The number of structure types contained in the container is unlimited, but should be reasonable. The header and footer in particular should only contain a few structure types, as these only contain simple information.

Page setting

In order for the structure to be displayed correctly on the page, a page format including margins must be set. You have the following options for this:

- In the SysConfig key "*HTMLtoPDF::wkhtmltopdf*" (menu *System > SysConfig*).
  Contains the configuration of the page setting for the converter, e.g. the page format.

- In the template structure (menu *System > Console > Console command*
  "`Console::Command::Admin::HTMLtoPDF::Inspect`")
    - see also section Console below.
    - In the section "`Page`" the margins are set:

Parameter	Description
Top	Top margin in millimetres
Bottom	Top margin in millimetres
Left	Left margin in millimetres
Right	Right margin in millimetres
SpacingHeader	Header spacing in millimetres The smaller the value, the further the header extends into the PDF.
SpacingFooter	Footer spacing in millimetres The smaller the value, the further the footer extends into the PDF.

Note: Please use only millimetres as the unit of measurement

13.2.4.3   Structure types

Structure types are independent HTML elements that create specific representations and are placed in the container. Some can be filled with specific data and others can be combined. Each structure type generally always consists of one line, but can also contain several lines.
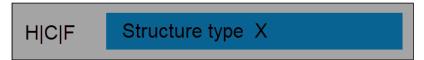
Fig.: Structure below a container

Blocks

The structure type "Blocks" is a container that combines other structure types. It enables:

- The combination of repeating structures without having to define them more than once.
- The combination of two adjacent structure types without influencing the following structure.
- The combination of related structures that are dependent on a "data". Advantage: If data is missing, the entire structure is displayed in the PDF.

Repeating structures

In order for such a structure to be mapped, please note:

1. "Blocks" must be of the type "List". This ensures that the underlying structure is repeated.
2. The following must be specified:
    1. An object that prepares the new data.
    2. The data, which provides the necessary ID, name, number.
3. Filters or expands (data expansion) are optional.
    1. Filter checks whether the fetched data is permissible.
4. If further data of the object is to be worked with within the block, this must be specified.
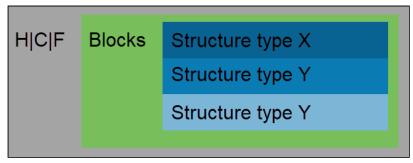
Fig.: Structure combination for an object

Parameter

Parameter*	Mandatory field	Hints
Object	✅	Object to be used in the underlying structures.
Expands		Details of additional data extension.
Filters		Restriction of the data list of Data.
Data	✅	Data from the data extension which are to be used in blocks. Attention: Only array data should be used here.
Type	✅	Specification of the type "List.
ID		Identifier of the entire block (default is Blocks).

see below: The parameters

Example: Structure code

```
{
    "Type": "List",
    "Object": "Article",
    "Expand": "DynamicField",
    "Data": "Article",
    "Blocks": [
        {
            "ID": "ArticleHeader",
            "Type": "Text",
            "Value": [
                "<Font_Bold>Article",
                "#<Count>"
            ],
            "Join": " ",
            "Break": true,
            "Translate": true,
            "Style": {
                "Size": "1.1em",
                "Color": "gray"
            }
        },
        ….
    ]
}
```

Simple double structure

Here it is only a matter of placing two structure types next to each other. An object, filter, expands or type are not needed.

Make sure, however, that the structure types below each have a style width of 48% - 50% and a float.

Fig.: Structure combination placed next to each other

Parameter

Parameter*	Mandatory field	Hints
Data		Data from the data expansion, which should be applied in blocks.
		If no data is given, only the main data of the object is used.
		Attention: In this case, the data should come from an expand, which is not an array.
ID		Identifier of the entire block (default is Blocks)

see below: The parameters

Example: Structure combination that are set next to each other as code

```
{
   "Data": "LinkObject",
   "Blocks":
   [
      {
         "ID": "LinkedHeader",
         "Type": "Text",
         "Value": "<Font_Bold>Linked Objects",
         "Translate": true,
         "Break": true,
         "Style":
         {
            "Size": "1.1em",
            "Color": "gray"
         }
      },
```

```
    {
        "ID": "LinkedTable",
        "Type": "Table",
        "Columns":
        [
            "<Font_Bold>Key",
            "Value"
        ],
        "Translate": true,
        "Join": "<br>"
    }
    ]
},
```

Text

The structure type "Text" is for the display of simple texts. The text can be deposited as a hyperlink. The URL for this must be specified in the "Value".

Parameter

Parameter*	Mandatory field	Hints
Type	✅	Must be "text
ID	✅	Identifier
AsLink		
Value	✅	Array or String Can resolve placeholders
Translate		
Style		Requires an ID Permitted styles: • Color • Height • Width • Float • BGColor • Size

Parameter*	Mandatory field	Hints
Break		Activates the dividing line below the structure type
Join		Required when Value is used as an array.

see below: The parameters

Example: Structure code for type text

```
{
    "ID": "ArticleHeader",
    "Type": "Text",
    "Value":
    [
        "<Font_Bold>Article",
        "#<Count>"
    ],
    "Join": " ",
    "Break": true,
    "Translate": true,
    "Style":
    {
        "Size": "1.1em",
        "Color": "gray"
    }
},
```

Richtext

Richtext is the HTML variant of "Text". Here, a complete HTML code can be inserted and displayed.

Parameter

Parameter*	Mandatory field	Hints
Type	✅	Must be "richtext"
ID	✅	Identifier

Parameter*	Mandatory field	Hints
Value	✅	Array or String Can resolve placeholders
Translate		
Style		Requires an ID Permitted styles: • Height • Width • Float • BGColor
Join		Required when Value is used as an array.

see below: The parameters

Example: Structure code for type richtext

```
{
    "ID": "ArticleBody",
    "Type": "Richtext",
    "Value": "<KIX_ARTICLE_BodyRichtext>"
}
```

Page

With "Page" you can specify the page number. Use this preferably only in the header or footer. The page number can be displayed as "Page x" or as "Page x of y".

Parameter

Parameter*	Mandatory field	Hints
Type	✅	Must be "page"
ID	✅	Identifier

Parameter*	Mandatory field	Hints
PageOf	✅	Sets the display type. • 0: "page x" • 1: "page x of y"
Translate		Activates the translation
Style		Requires an ID Permitted styles: • Height • Width • Float • Size

see below: The parameters

Example: Structure code for type Page

```
{
    "ID": "Paging",
    "Type": "Page",
    "PageOf": 0,
    "Translate": true,
    "Style":
    {
        "Float": "right"
    }
}
```

Image

With "Image" you can insert individual images, such as a header logo. There are three ways in which the image can enter the document:

1. From the KIX database table (ObjectIcon).
2. As a file path from the server (where the KIX backend is located).
3. As passed Base64 content.

Note that the ContentType must always be "Image".

Parameter

Parameter*	Mandatory field	Hints
Type	✅	Must be "image"
ID	✅	Identifier
Value	✅	Depending on types: • **DB**: ObjectID of the object of ObjectIcon • **Path:** System path to the image (placeholder possible) • **Base64:** Direct Base64 code in HTML image src style
TypeOf	✅	Permitted types: • DB • Path • Base64
Style		Requires an ID Permitted styles: • Height • Width • Float

see below: The parameters

Example: Structure code for type Image

```
{
    "ID": "PageLogo",
    "Type": "Image",
    "Value": "agent-portal-logo",
    "TypeOf": "DB",
    "Style":
    {
        "Width": "2.5rem",
```

```
        "Height": "2.5rem",
        "Float": "left"
    }
},
```

Table

Tables can represent large amounts of data in different forms. Currently, there are three structure variants to represent this data:

- KeyValue (standard)
- DataSet
- Custom

The selected structure variant is given to the table as a sub-type.

Code structure for the table display

```
{
  "Columns": [
      "Key",
      "Value"
  ],
  "Type": "Table",
  "SubType": "KeyValue | DataSet | Custom",
  ...
},
```

ⓘ **Hint**

Since space is limited on an A4 sheet in portrait format, only necessary attributes should be displayed in the tables.

KeyValue

This table type is the default type. It is used to display attributes of an object. If no SubType is specified in the configuration, "KeyValue" is set automatically.

Table structure

Basically, "KeyValue" is a two-column table, which contrasts the data of the object as attribute name (Key) and attribute value (Value). For this purpose, the values "Key" and "Value" are stored in the parameter "Columns" (array). By omitting one of these values, a single-column table can be created in which only the attribute names or only the attribute values are displayed.

By additionally specifying "Count", you can add a numbering column to the table.

Which content the table displays can be restricted with the parameters "Allow" and "Ignore".

Standard representation

Key	Value
Attribute name	Attribute value

Code structure for the table display (KeyValue without Count)

```
{
   "Columns": [
      "Key",
      "Value"
   ],
   "Type": "Table",
   "SubType": "KeyValue",
   ...
},
```

Standard representation with Count

Count	Key	Value
1	Attribute name	Attribute value

Code structure for the table display (KeyValue with Count)

```
{
   "Columns": [
      "Count",
      "Key",
      "Value"
   ],
   "Type": "Table",
   "SubType": "KeyValue",
   ...
},
```

Allow/Ignore

With the parameters "Allow" and "Ignore" you can restrict the attributes:

- **Allow:** defines a whitelist. Only the named attributes are displayed.
- **Ignore:** defines a blacklist and denies attributes.
  The attributes that have made it through the whitelist are removed.
  If no Ignore is set, the final result is the same as the Allow application.

The structure of both parameters is a hash in JSON format. Key is the attribute name. The associated value can be used to determine what is to be done with the attribute.

- If the value "KEY" is set to the attribute, then the entire attribute is either permitted (Allow) or ignored (Ignore).
- If a regular expression is set to the attribute instead of the "KEY", then this regular expression is applied to the value of the attribute.

All other attributes not specified in the "Allow" will not be displayed in the table.

Allow - Allows the following attributes (object is ticket)

```
[...],
"Allow": {
   "State": "KEY",      // allows attribute – no matter what it says, because "KEY"
is set
   "Type": "^Prob.*"    // allows attribute only if the expression is contained in
the attribute value
}
```

"Ignore" can be used in addition to "Allow". The structure is the same as "Allow", except that the specified attributes are removed from the table.

To be noted:

- If an "Allow" is set for the table, the "Ignore" is applied to the attributes that were filtered by "Allow".
- If no "Allow" is set, "Ignore" goes over all attributes and restricts them.

Ignore - discards the following attributes (object is ticket)

```
[...],
"Ignore": {
   "State": "KEY",      // discards the attribute – no matter what it says, because
"KEY" is set
   "Type": "^Prob.*"    // discards the attribute only if the expression is contained
in the attribute value
}
```

Example: Structure code tablet type KeyValue

```
{
  "ID": "InfoTableLeft",
  "Type": "Table",
  "Include": "DynamicField",
  "SubType": "KeyValue",
  "Columns":
  [
    "<Font_Bold>Key",
    "Value"
  ],
  "Allow":
  {
    "State": "KEY",
    "Queue": "KEY",
    "Lock": "KEY",
    "CustomerID": "KEY",
    "Owner": "KEY",
    "Responsoble": "KEY",
    "Type": "KEY",
    "Priority": "KEY"
  },
},
```

DataSet

With "DataSet" data sets can be displayed as a table. Each data set is a row in the table. You can determine which data (attributes) of the data set are to be displayed.

Table structure

With this table type, you can define the table structure as you wish by specifying the desired attribute names under "Columns". The specified attribute names form the table columns. The respective data set fills the cells with the corresponding values based on the set column.

You can use "Allow" and "Ignore" to restrict the data in the table. However, this does not affect the individual attributes, but the entire data set.

Standard representation

Attribute name	[Attribute name...]
Attribute value	[Attribute value...]

Code structure for the table display (DataSet)

```
{
  "Columns": [
    "AttributeX",
    "AttributeY",
    ...
  ],
  "Type": "Table",
  "SubType": "DataSet",
  ...
},
```

Allow/Ignore

With "DataSet", the restriction affects the entire data set and not individual columns. That is:

- **Allow**: The record is only admitted if all values of the attributes to be checked match.
- **Ignore**: The record is only ignored if one of the values of the attributes to be checked matches.

> ⓘ **Note**
>
> For "DataSet", avoid using "Key" for Allow/Ignore. Check only with regular expressions.

Allow - Allows the following record (object is ticket)

```
[...],
"Allow": {
   "Type": "^Prob.*"    // allows the record only if the expression for the attribute
is contained in the attribute value.
}
```

Ignore - discards the following record (object is ticket)

```
[...],
"Ignore": {
   "Type": "^Prob.*"   // discards the record if the expression for the attribute is
contained in the attribute value.
},
```

Example structure code tablet type DataSet

```
{
  "ID": "Example",
  "Type": "Table",
  "Include": "DynamicField",
  "SubType": "DataSet",
  "Columns":
  [
    "AttributeX",
    "AttributeY"
    "AttributeZ"
  ],
  "Ignore":
  {
    "AttributeX": "Prob.*",
  },
},
```

Custom

Compared to the other two table types, "Custom" is more flexible with regard to the presentation of content:

- Free text can be used without reference to an object attribute.
- Several object attributes can be entered in one cell.
- Any number of columns and rows can be defined.
    - **Note**: Too many columns will be cut off during printing because they protrude beyond the margin.
- Object-related KIX placeholders can be used.
- A restriction with Allow/Ignore is not possible because this table type has freely selectable content. Thus, it cannot be determined what is permissible or should be ignored.

Tip: In the structure template delivered by KIX, this table type is used for the display of data in the block "Customer Information".

Table structure

With this table type you can freely choose the structure. The parameter "Rows" supports you in this. It defines the contents of the individual rows in the respective columns. The parameter "Rows" is an ArrayOfArrays, where the first index is the row and the second index is the respective column. "Rows" does not contain the column headings, but only the table contents.

Structure Rows

Code structure for table display (custom) Rows only

```
[...],
"Rows": [
    [ // Row 1
        "Row1Column1"     // Column 1
        "Row1Column2"     // Column 2
    ],
    [...]   // Row 2
]
```

Structure Columns and Rows

Code structure for the table display (custom) Columns and Rows

```
[...],
"Columns": [
    "",
    "Column2"
],
"Rows": [
    [ // Row 1
        "Row1Column1"  // Column 1
        "Row1Column2"     // Column 2
    ],
    [...]   // Column 2
]
```

Example structure code tablet type custom

```
{
  "Type": "List",
  "Object": "Contact",
  "Data": "Contact",
  "Expand": "DynamicField",
  "Blocks":
  [
    {
      "ID": "Contact",
      "Type": "Table",
      "SubType": "Custom",
      "Include": "DynamicField",
      "Columns":
      [
        "<Class_Col1><Font_Bold>",
        ""
```

```
    ],
    "Rows":
    [
      [
        "Contact",
        "Contact.Fullname.Value"
      ],
      [
        "Phone",
        "Contact.Phone.Value"
      ],
      [
        "Mobile",
        "Contact.Mobile.Value"
      ]
    ],
    "Translate": true,
    "Style":
    {
      "Class":
      [
        {
          "Selector": " .Col1",
          "CSS": "width: 20%;"
        }
      ]
    }
  }
]
}
```

Parameter

Parameter*	Mandatory field	Hints
Type	✅	Must be "Table"
ID	✅	Identifier
Allow		Returns a hash with attributes which, depending on the variant, represents the attribute or the data set in the table. (Whitelist)
Ignore		Returns a hash with attributes which, depending on the variant, does not represent the attribute or the data set in the table. (Blacklist)

Parameter*	Mandatory field	Hints
Columns	✅	Details of the columns per variant
Rows	✅	Only for table type "Custom Defines the table content by row and column
Include		Adds an expand to the main data for this table only. **Attention:** The expand should have been fetched from the object beforehand.
Translate		Translates column contents incl. column header if activated.
Style		Requires an ID Permitted styles: • Height • Width • Float
Headline		Activates the column header • false: inactive • true: active
SubType		Defines the table type Permitted types: • KeyValue (Standard) • DataSet • Custom

see below: The parameters

13.2.4.4   Console

In the menu *System > Console* 2 console commands are available for HTMLToPDF:

- Inspect an existing template: `Console::Command::Admin::HTMLToPDF::`**`Inspect`**

- Create a PDF at console level: `Console::Command::Admin::HTMLToPDF::`**`Convert`**

Inspect

With the console command `Console::Command::Admin::HTMLToPDF::Inspect` , an existing template of an object (article or ticket) can be viewed. The structure of the template is displayed in PEARL format.

Based on the displayed structure, you can see how the template is structured and which parameters are initially set. You can use this information, for example, to set filters, expands, allows or ignores in the configuration of the macro action "Convert to PDF" (KIX Pro) or when executing the convert command.

Users of on-premises installations can optionally use the following command in the console (terminal):

```
webuser@system:/<KIX_HOME>$ ./bin/kix.Console.pl Admin::HTMLToPDF::Inspect --
allow-root
```

Options

You can add additional options to the console command to specify which templates you want to view:

Option	Mandatory field	Description
`--name`	✅	Name of the template to be viewed. The option "--id" is then not necessary. Examples: • `--name Ticket` • `--name Article`
`--id`	✅	ID of the template to be viewed (if known). The option "--name" is then not necessary. Examples: • `--id 1` (Ticket) • `--id 2` (Article) **Note:** Depending on the installation, the ID may be different. We recommend using the name.

Convert

The console command `Console::Command:: Admin::HTMLToPDF::Convert` initiates the conversion. This means that the HTML is generated from the selected template and then converted to a PDF.

Users of on-premises installations can optionally use the following command in the console (terminal):

```
webuser@system:/<KIX_HOME>$ ./bin/kix.Console.pl Admin::HTMLToPDF::Convert --allow-root
```

The Convert command is the counterpart to the macro action "Convert to PDF (see page 819) ".

The Convert command places the PDF where the system is located. If no file name or storage location is specified, the default is used. The default storage directory is currently "/tmp". The default file name is formed from the template name, the supplied ID or number and the current time.

```
<Template Name>_<ID or Number>_<Timestamp> → Ticket_1_20221008090052
```

Options

You can add additional options to the console command to manipulate the PDF output:

Option	Mandatory field	Description
`--name`	✅	Name of the template to be converted, e.g. ticket
`--user_id`		UserID of the creator. The name of the user in the document is given as "<Current_User>" if this is set.
`--filename`		Specific file name of the PDF (specification without file extension)
`--directory`		Storage location of the created PDF
`--filter`		(as JSON string) Filters the objects set in the template (see "Filter" below).
`--allow`		(as JSON string) Overwrites the Allows set in the table (see "Allow" below).

Option	Mandatory field	Description
`--ignore`		(as JSON string) Overwrites in the ignores set in the table (see "Ignore" below).
`--id`	✅	Necessary ID so that the object can prepare the data (e.g. TicketID or ArticleID). What the object expects can be found out with "`--help`".
`--number`	✅	Necessary number so that the object can prepare the data. What the object expects can be found out with "`--help`".
`--help`		Provides information about a command.

Filter

Filters can be used to filter objects set in the template. The main object should not be included, otherwise the PDF could be empty. This must be stored as a JSON string in the command.

Structure Filter Format

```
{
   "Object":
   {
      "AND|OR":
      [
         {
            "Field": "Attribute",
            "Type": "EQ|CONTAINS",
            "Value": "some values"
         }
      ]
   }
}
```

Example Filter Format

```
{
    "Article":
    {
        "AND":
        [
            {
                "Field": "ArticleType",
                "Type": "EQ",
                "Value": "external"
            }
        ]
    }
}
```

Allow

With Allow, Allows set by tables can be overwritten. The ID of the respective structure type is necessary for this. In addition, this must be constructed as a JSON string.

Structure of the Allow Format

```
{
    "ID-Structurtype":
    {
        "Attribute": "KEY or some value",
        ...
    }
}
```

Example Allow Format

```
{
    "InfoTableLeft":
    {
        "Type": "KEY",
        "Lock": "lock",
        "Owner": "KEY"
    }
}
```

Ignore

Ignore can be used to overwrite ignores set by tables. The ID of the respective structure type is necessary for this. In addition, this must be constructed as a JSON string.

Structure Ignore Format

```
{
    "ID-Structurtype":
    {
        "Attribute": "KEY or some value",
        ...
    }
}
```

Example Ignore Format

```
{
    "ArticleMeta":
    {
        "Channel": "KEY"
    }
}
```

Examples

Example 1: Ticket with one article

Only one specific article is to be printed with additional ticket information. In principle, the template "Article" could be used for this. However, this does not contain any ticket information, which is why the template "Ticket" is to be used. The template "Ticket", however, prints all articles of the ticket. Therefore, a filter must be added to the command, which selects the desired article (see also section "Filter" above). In this example, the filter is limited by the article number.

> ⓘ **Note**
>
> The article number is not a parameter of the article, but an external counter. This can only be applied with the "Ticket" template.

A filter set via the console command must be a JSON string. If necessary, use a JSON editor (e.g. jsonformatter.io[14]) for this. For this example, the filter looks like this:

---

14 http://jsonformatter.io

```
{
    "Article":
    {
        "AND":
        [
            {
                "Field": "ArticleNumber",
                "Type": "EQ",
                "Value": "4"
            }
        ]
    }
}
```

Minimize the JSON string and then paste it into the console command (click on "minify" in the JSON editor). After sending the console command, the PDF is created. It contains the ticket information and the selected article.

```
webuser@system:/<KIX_HOME>$ ./bin/kix.Console.pl Admin::HTMLToPDF::Convert --allow-
root --id 1 --name Ticket
--filter '{"Article":{"AND":[{"Field":"ArticleNumber","Type":"EQ","Value":"4"}]}}'
```

The macro action "Convert to PDF" also expects the filter as a JSON string. Here it is not necessary to minimize the JSON string.

Example 2: Change article information

In the delivered templates, the information contained in the PDF is predefined. Since this information may be sufficient or too extensive, it can be temporarily influenced with --allow (see section "Allow") or --ignore (see section "Ignore"). This influence only applies to the current execution of the console command or only when the macro action "Convert to PDF" is executed.

It must be weighed up beforehand what is easier to map. Sometimes it is easier to use a whitelist (allow) than a blacklist (ignore) - or vice versa. It always depends on what is to be displayed.

> ⚠ **Attention**
>
> If both options (allow and ignore) are given and both options contain an ID, the ID from "allow" is used.

In this example, "ignore" is used. The structure of the JSON is the same as for "allow". The difference is that "ignore" is a blacklist and "allow" is a whitelist. Since "ignore" is used, all information about the article is also displayed. This also applies to the dynamic fields, provided that an "include" is set for the structure type. This is not the case in the standard system.

First, the ID of the structure type "Table" is required. This specifies the table to which the restriction of the information to be displayed applies. In both templates, the ID "ArticleMeta" contains the article information.

In the example, all article information is to be displayed except for the CustomerVisible, MessageID, TicketID, ChannelID, Body, Cc and Bcc. However, the channel should only be displayed if it is not "note":
(**Note**: "KEY" is a trigger for the entire attribute to be either added or omitted).

```json
{
   "ArticleMeta":
   {
      "CustomerVisible": "KEY",
      "MessageID": "KEY",
      "TicketID": "KEY",
      "ChannelID": "KEY",
      "Body": "KEY",
      "Cc": "KEY",
      "Bcc": "KEY",
      "channel": "note"
   }
}
```

Minimize the JSON string and then paste it into the console command (click on "minify" in the JSON editor). After sending the console command, the PDF is created.

```
webuser@system:/<KIX_HOME>$ ./bin/kix.Console.pl Admin::HTMLToPDF::Convert --allow-root --id 1 --name Ticket
--ignore '{"ArticleMeta":
{"CustomerVisible":"KEY","MessageID":"KEY","TicketID":"KEY","ChannelID":"KEY","Body":
"KEY","Cc":"KEY","Bcc":"KEY","Channel":"note"}}'
```

The macro action "Convert to PDF" also expects the information in the parameter "Allows" or "Ignores" as a JSON string. Here it is not necessary to minimize the JSON string.

13.2.4.5 The parameters (significance)

Parameter	Expected Data type	Meaning
Object	String	Is the object that prepares and provides the data. Depending on the object, an ID, name or number is required for the preparation.

Parameter	Expected Data type	Meaning
Expands	Array | String	With this parameter, the data set of the object can be expanded by further data. Different expands are possible for each object. Depending on the expand, either a data expansion or an ID list is provided.
Data	String	(only applicable with blocks) Only the data that was requested via "Expands" can be used explicitly. Please note that this is the data of the super-ordinate object expand and not of the object that is being built.
Allow	Hash	Whitelist for the structure type Table. Depending on the table structure, the data used in it is also filtered accordingly. For example, if only a KeyValue type is used, only certain attributes can be admitted. If instead only the attribute type is used, only certain data records can be admitted - as long as the values of the attributes to be checked are correct. There are two value checks that can be applied: • **KEY**: Allow the record without directly checking the attribute. • **Check value**: A value to be cross-checked with the value of the parameter. Regular expressions are possible.

Parameter	Expected Data type	Meaning
Ignore	Hash	Is the counterpart to "Allow" and thus a blacklist for the structure type Table.
		Depending on the table structure, the data used in it is also filtered accordingly. For example, if only a KeyValue type is used, certain attributes can be ignored. If the attribute type is used instead, certain data records can be omitted - as long as the values of the attributes to be checked match.
		There are two value checks that can be applied:
		• **KEY:** Omit the record without directly checking the attribute.
		• **Check value**: A value to be checked against the value of the parameter. Regular expressions are possible.
Columns	Array	A parameter for the structure type "Table".
		This parameter can be used to determine how the table is to be structured. A distinction is made between the types:
		• **KeyValue**: Specification of the parameter as a key-value pair. This means: In the table, the key name is specified as "Key" and the value as "Value" for each column.
		``` 'Columns' => ['<Class_Key><Font_Bold>Key', 'Value'], ```
		• **Attritbute**: Defines which attribute should be displayed in which column.
		Info: "KeyValue" and "Attribute" are here only indirect designations for clarification.

Parameter	Expected Data type	Meaning
Rows	ArrayOfArrays	A parameter for the structure type "Table" and the table type "Custom". This parameter can be used to define the content of the table. This parameter is in combination with Columns. ```'Rows' => [`   ` [`   ` 'Title',`   ` 'Contact.Title.Value',`   ` 'Phone',`   ` 'Contact.Phone.Value'`   `],```
Headline	Boolean	Activates the header line of the table and displays the entries stored under "Columns".
ID	String	Is the identifier of the respective structure type. Ensures that the set "Style" only affects the structure type in question. In tables with the specified ID, the use of "Allow" or "Ignore" can override the "Allow" or "Ignore" defined inital in the respective table.
Translate	Boolean	Activates the translation.

Parameter	Expected Data type	Meaning
Style	Hash	Defines the representation of the respective structure type. Not every style type is used in every structure type. Attention: For a good display, only the dimensions %, em or rem should be specified in the style. • Width (rem, em, %): Influences the width of the structure type. If possible, only specify percentage values (e.g. 50%). • Height (rem, em): Height of a structure type. • Size (rem, em): Font size • Float (right,left): Alignment of the structure type • Color (hex, colour name): Font colour • BGColor (Hex, colour name): Background colour
Join	String	With Join, text passages can be combined if an array is used as the value for the structure type "Text".
Break	Boolean	Sets a separator line below the structure type "Text".
Value	Array | String	Values to be displayed in the respective structure type (placeholder substitution possible). Not suitable for structure type "Table". ```\n'Value' => 'Ticket.Title.Value'\n\n'Value' => [\n '<Font_Bold>Customer Information'\n]\n``` This parameter can be used as an array or string. The background for the array is that complete texts are not always translated or, in combination with content from placeholders, this content does not exist. This means that the individual passages can be combined with "Join" and translated individually beforehand (Translate).

Parameter	Expected Data type	Meaning
Type	String	Is the structure type (Table, Page, Image, Text or Richtext)
SubType	String	Is the table type (Custom, KeyValue, DataSet)
PageOf	Boolean	Is the structure type (Table, Page, Image, Text or Richtext)
TypeOf	String	Is part of the structure type "Image". Determines what is to happen with the value from the "Value" parameter. • **DB:** Searches for an image using the "ObjectID" in the database table "ObjectIcon". • **Path:** Searches locally on the server for an image. Then adds it to the structure as Base64. • **Base64:** Passes a Base64 string (e.g. an image) to the structure type. **Note**: The HTML notation should be used (data:\<ContentType\>; base64,\<base64Content\>).
AsLink	Boolean	Determines whether the structure type is to be displayed as a link. **Attention**: Therefore, a URL should also be stored as a "Value".

13.2.4.6   Additional hints

Specific placeholders

In some places there are placeholders such as \<Font_Bold\>. These are not KIX placeholders in the conventional sense, but style placeholders for tables and texts. They can be used, for example, to display certain parts of the text in bold. In addition to the style placeholders, there are three more:

Placeholder	Description
<Current_Time>	Returns the current date incl. time
<Current_User>	Returns the name of the user who performed the conversion.
<Count> | Count	Returns a prefixed number (used for tables to mark records).

Placeholder for Object Attributes

In addition, object-specific placeholders can be applied.

Syntax:	<Object>.<Attribute>.<Key|Value>[.<ValueIndex>]

Attribute	Description
<Object>	Is the object that uses the template. If a block has a "Data", the object changes to the object of the "Data" because other data is used for the block at that time.
<Attribute>	The attribute that is part of the object. With the parameter "Inlcude" at the block, "Expands" from the object can be used directly as an attribute.
<Key|Value>	Determines what is to be displayed. "Key" shows the attribute name and "Value" shows the attribute value.
[.<ValueIndex>]	If the attribute is an array, the index can be specified after the value to determine the value to be displayed. If no index is specified, all values are displayed comma-separated.

Example placeholder (object is ticket)

```
// TicketNumber
// <Object>.<Attribut>.<Key>
Ticket.TicketNumber.Key
```

```
// Dynamic Field (only if Include DynamicField is set)
// <Object>.<Attribut>.<Value>
Ticket.DynamicField_XYZ.Value

// Multiselect  Dynamic Field  (only if Include DynamicField is set //
<Object>.<Attribut>.<Value>.<Index>
Ticket.DynamicField_XXY.Value.1
```

Class specific styles (CSS)

With the placeholder <Class_...>, a column (under Column) or a text/richtext (under Value) can be given a style class. This class is added to the respective HTML tag of the block type.

The respective CSS must be set for a class. This is done in the parameter "Style" with the key "Class" (see example below). "Class" is an ArrayOfHashes, where each hash is a class including CSS. There are two keys in the hash:

Key	Description
Selector	CSS class, which is set to the BlockID. Attention must be paid to the CSS rules of the selectors. `.<BlockID><Selector> {}`
CSS	CSS Contains the CSS specifications for the selector. The information is given as a string and is therefore written without {}.

Example

```
{
  "Type": "List",
  "Object": "Organisation",
  "Data": "Organisation",
  "Expand": "DynamicField",
  "Blocks":
  [
    {
      "ID": "Organisation",
      "Type": "Table",
      "SubType": "Custom",
      "Include": "DynamicField",
      "Columns":
      [
```

```
                "<Class_Col1><Font_Bold>",
                "<Class_Col2>",
                "<Class_Col1><Font_Bold>",
                "<Class_Col2>"
            ],
            "Rows":
            [
                [
                    "Organisation",
                    "Organisation.Name.Value",
                    "Customer No. / Type",
                    "Organisation.Number.Value / Organisation.DynamicField_Type.Value.0"
                ],
                [
                    "Street",
                    "Organisation.Street.Value",
                    "ZIP, City, Country",
                    "Organisation.Zip.Value, Organisation.City.Value,
Organisation.Country.Value"
                ]
            ],
            "Translate": true,
            "Style":
            {
                "Class":
                [
                    {
                        "Selector": " .Col1",
                        "CSS": "width: 20%;"
                    },
                    {
                        "Selector": " .Col2",
                        "CSS": "width: 30%;"
                    }
                ]
            }
        }
    ]
},
```

13.2.4.7 Macro Action "Convert to PDF"

The macro action "Convert to PDF" is the counterpart to the console command `Console::Command::Admin::HTMLToPDF::Convert` . It can be used to output ticket or article information in a PDF file by means of a job or action.

The macro action uses the initial structure templates as a basis for ticket or article printing. In the configuration of the Macro Action, you can specify which of the ticket/article information defined in the template is to be taken over into the PDF. The specifications of the templates are only temporarily changed for the PDF output.

As a return value, the macro action provides a file attachment in JSON format containing "Content", "Filename" and "ContentType". This return value can be referenced in the macro action "Article Attachment Add", for example, to set the generated PDF as an attachment to an article.

For more information on the Macro Action, see the Overview of Macro Actions for KIX Pro (see page 819) .

13.2.5 Display asset attributes in ticket zoom view

Starting point:

If an "Affected Asset" is selected on a ticket, the manufacturer and model designation of the affected device should be automatically copied to the ticket. This information can be used for problem analyses, evaluations or workflow controls.
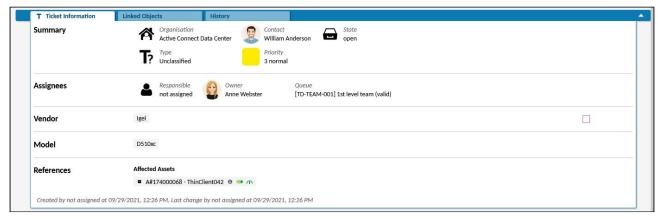
Fig.: Details of the manufacturer and model of an affected device in the ticket details

Prerequisites:

- The affected device is already recorded.
- The device is assigned to an asset class (e.g. "Computer")
- The vendor and model name details are specified.

Procedure:

For the display of the asset information, 2 dynamic fields are required, which are integrated into the ticket interfaces (here: ticket zoom view). The dynamic fields are filled with the asset information by a job. The job uses the action "Fetch Asset Attributes" and references the affected asset.

1. Create dynamic fields:

- Navigate to *System > Dynamic Fields*. Create 2 Dynamic Fields (e.g. "DFVendor" and "DFModel").
  - Field type "Text
  - Object type "Ticket".

2. Include the Dynamic Fields for display in the ticket details:

- Navigate to *System > SysConfig* and open the key "*ticket-details-info-card*".
- Integrate the Dynamic Fields "DFVendor" and "DFModel" into the value of the configuration key (s. Displaying Values of Dynamic Fields (see page 442)).
- Reload the frontend.

Fig.: Include the manufacturer and model details in the ticket details.

You can use the following code block as a basis:

```
{
"title": "Translatable#Vendor",
"separator": true,
"values": [
  [
    {
      "componentId": "dynamic-field-value",
      "componentData": {
        "name": "DFVendor"
      },
      "conditions": [
        {
          "property": "DynamicFields.DFVendor",
          "operator": "NE",
          "value": null
        }
      ]
    }
  ]
```

```
    ]
  },
  {
    "title": "Translatable#Model",
    "separator": true,
    "values": [
      [
        {
          "componentId": "dynamic-field-value",
          "componentData": {
              "name": "DFModel"
          },
          "conditions": [
            {
              "property": "DynamicFields.DFModel",
              "operator": "NE",
              "value": null
            }
          ]
        }
      ]
    ]
  },
```

Option: Additionally, you can integrate the dynamic fields into the "Edit ticket" dialogue (see Integrating a Dynamic Field (see page 423)).

3. Configure the job:

- Navigate to *Automation > Jobs* and create a new job. Use the following configuration:

1. Job Information:
    - Job Typ: Ticket
    - Name: Name of the job (e.g. "Affected Asset Details")
    - Validity: valid
2. Execution plan:
    - Event: TicketCreate
3. Filter: none
4. Actions:
    - 1. Action: Fetch Asset Attributes
    - Asset Reference Dynamic Field: AffectedAsset
    - Attribute - DynamicField Mapping:
        - Attribut: "Vendor" | DynamicField: "DFVendor"

- Attribut: "Model" | DynamicField: "DFModel"

Fig.: The linking of asset attributes with the dynamic fields

13.2.6 Generate ticket numbers

When creating a new ticket, KIX automatically generates a new ticket number. You can define the formatting of the ticket numbers by editing the corresponding SysConfig keys in the menu System > SysConfig.

> ⓘ **Info**
>
> The SysConfig keys cannot be set to (temporarily) invalid.

SysConfig key	Description	Parameters
Ticket::NumberGenerator	Defines which module is used for generating the ticket numbers.	• **DateChecksum:** Inserts the counter as a checksum after the date and the SystemID. The checksum rotates daily. Format: "Jahr.Monat.Tag.SystemID.Zähler.Checksum" (z. B. 2022113017000011, 2022113017000029) • **Date:** Generates the ticket number from the current date, the SystemID and the counter. Format: "Jahr.Monat.Tag.SystemID.Zähler" (z. B. 20221130176, 20221130177) • **AutoIncrement:** Increments the counter continuously. The SystemID and the counter are displayed in the format "SystemID.Zähler" (z. B. 1700004, 1700005) • **Random**: Generates random ticket numbers in the format "SystemID.Random" (z. B. 179936114563, 175558963640)

SysConfig key	Description	Parameters
Ticket::NumberGenerator::CheckSystemID	Checks the SystemID when recognising ticket numbers in queries.	Default: 1 Use "0" if the SystemID was changed after the system was used.
Ticket::NumberGenerator::Date::UseFormattedCounter	Activates MinCounterSize for the ticket counter if "Date" or "DateChecksum" is selected as TicketNumberGenerator.	Activate (default): 1 Deaktivate: 0
Ticket::NumberGenerator::MinCounterSize	Sets the minimum ticket counter size when "AutoIncrement", "Date" or "DateChecksum" is selected as TicketNumberGenerator.	Default: 5 (the counter starts at 00001, z. B. 2022113017**000011**) Example: 3 ((the counter starts at 001, z. B. 20221130017**0011**)

13.3   Settings for organisations and contacts

The following chapters contain advanced configuration options for contacts and organisations in the Organisations Dashboard.

- Allow contacts without or without a unique email address (see page 566)
- Automatic assignment of the organisation to a contact (see page 567)
- Provide dynamic fields to organisations (see page 571)
- Control the search for organisations and contacts (see page 588)

13.3.1 Allow contacts without or without a unique email address

SysConfig key	Description	Possible Values
ContactEmailUniqueCheck	Enables/disables the uniqueness check for contact email addresses.	0 | 1

A unique email address must be assigned to each contact in KIX. This is ensured by a validity check when creating the contact, which issues a warning in the event of an error.

To cover various B2C use cases, it may be necessary to deactivate the validity check and thus store contacts without or without a unique email address in the CMDB. For example, if you contact your customers exclusively via SMS or if you generate contacts without a valid email address automatically via an interface.

To deactivate the validity check, set the value in the above key to 1.

> ❗ **Attention!**
>
> **Once a validity check has been deactivated, it MUST NOT be reactivated!**
> Should this nevertheless be necessary in exceptional cases, you **MUST** manually **ensure** that **ALL** email addresses exist only once in the system and that a unique email address is set at each contact.

> ⚠ **Important!**
>
> If the validity check is deactivated, notifications can be read by contacts who should not receive personal information about the current contact. This can be a violation of the GDPR regulation! Therefore, make sure that no notification contains relevant information.
> Furthermore, any email-based contact query or identification can potentially yield ambiguous results.

If the validity check is deactivated, incoming emails are assigned to the first contact - ordered by last name - with the first name assigned to the email address.

Some functions such as

- email-based authentication
- the email-based updating of contact data from LDAP
- or the macro action "CreateOrUpdateContact"

are disabled if no lookup value other than email is specified.

No name changes are possible during a data import, regardless of whether the validity check is activated or deactivated. The identifier is: Email address + surname + first name. Only when the data has been created in the system can it be edited.

13.3.2 Automatic assignment of the organisation to a contact

Menu	KIX > System > SysConfig
Configuration key	Contact::EventModulePost###800-AutoAssignOrganisation

KIX can automatically assign an organisation to a new contact, e.g. when:

- automatic retrieval of emails: If an email is retrieved from a previously unknown contact, a new contact is created and the organisation is assigned.
- manual creation of a new ticket: If a previously unknown email address is entered as a contact, a new contact is created and the organisation is assigned.
- manual creation of a new contact: If no organisation is specified, this can be set automatically.

The organisation is assigned (depending on the configuration) based on the domain of the email sender. Up to 25 domain patterns can be stored for each organisation. KIX checks the domain of the email sender or the specified email address and searches for the corresponding pattern in the organisations. If the domain is found in the domain pattern of an organisation, this organisation is assigned to the new contact.

Example: The domains "*kixdesk.com" and "*.capeIT.de" are stored in the "capeIT" organisation. The previously unknown email senders "info@capeIT.de" or "mail@kixdesk.com" are automatically assigned to the "capeIT" organisation.

13.3.2.1 Assignment methods

The automatic assignment of the organisation to a contact is an event. It is triggered if <u>no</u> organisation is specified when a contact is created or updated. If an organisation was specified (manually), this applies. The automatic process does not take effect then.

The event comprises 3 assignment methods that you can deactivate/activate as required. KIX processes the methods in the following order. If a method is deactivated or a method could not be applied successfully (error/no result), the next method is checked and applied. **If all methods are deactivated or if there were errors or no result, no organisation is set on the contact.**

Konfigurationsschlüssel: Contact::EventModulePost###800-AutoAssignOrganisation

```
1   {
2      "Event": "(ContactAdd|ContactUpdate)",
3      "MappingMethods": [
4         {
5            "Active": "1",
6            "Method": "MailDomain"
7         },
8         {
```

```
 9              "Active": "0",
10              "DefaultOrganisation": "MY_ORGA",
11              "Method": "DefaultOrganisation"
12          },
13          {
14              "Active": "0",
15              "Method": "PersonalOrganisation"
16          }
17      ],
18      "Module": "Kernel::System::Contact::Event::AutoAssignOrganisation"
19  }
```

Sequence	Method	Description

1	MailDomain	• This method checks the mail domain of the contact (e-mail) against the domain patterns that are stored in the organisations.
		    • The method is initially active and is the default behaviour.
		    • All organisations for which the sender's mail domain is stored are assigned to the contact.
		    • The organisation found first is set as the primary organisation (PrimaryOrganisationID) on the contact. All other organisations are set as secondary organisations (OrganisationID) on the contact (sorted based on customer number).
		    • If there is an error, no result or inactive, the next method is used (→ DefaultOrganisation).
		• The "Mail domains" area exists for this method at each organisation. Here you can specify what the "MailDomain" method should look for. Up to 25 search patterns can be stored.
		    • The field for the patterns is a dynamic field "AddressDomainPattern", which is of type "Text" and object "Organisation". It cannot be deleted.
		    • You can use wildcards. The use of wildcards always applies to the entire part of a section of the domain. Sections are separated with ".".
		        • Example domain:
		        `my.company.com`
		        • Possible patterns:
		            • `my.company.com`
		            • `*.company.com`
		            • `*.company.*`
		            • `*.*.de`
		            • `my.*.com`

Sequence	Method	Description
		• `my.company.* *.*` • `my.company.* *.*.*` • It is not possible to use a hyphen, e.g: `*-company` `.de`

2	DefaultOrganisation	• This method can be used to assign a fixed organisation to the contact, e.g:
		`"DefaultOrganisation" : "MY_ORGA"`
		  • The ID, name or number of the organisation can be specified.
		  • The method can be used as the 1st fallback to the "MailDomain" method.
		  • The method is initially inactive and can be activated if required. To do this, set the parameter `"Active" : "1"`.
		  • If there is an error, no result or inactive, the next method is used (→ PersonalOrganisation).
		
		Note: If this mapping method is activated, the SSP must be prevented from viewing the "Tickets by others" (data protection). Therefore, change the authorisations for access in the SysConfig key *AssignedObjectsMapping* (see also Control visibilities in the SSP).

3	PersonalOrganisation	• This method is used to create a separate organisation for the contact, which consists of the contact's email address. • If the organisation could not be created, an error occurred or this method is inactive, the next method (→ last fallback) is used. • In the last fallback, no organisation is set for the contact.

13.3.3 Provide dynamic fields to organisations

The following use case shows how an additional input field is provided in the "Edit organisation" dialogue and how the value stored in the field is displayed in the overview of organisations. In the example, the short name of the organisation is to be entered and displayed.

The following steps are required:

1. Create dynamic field.
2. Provide dynamic field in the dialogue "Edit organisation.
3. Provide the value of the dynamic field in the organisations details.
4. Provide the value of the dynamic field in the organisations dashboard.

Content on this page:

- Create dynamic field (see page 571)
- Provide dynamic field (see page 573)
  - Provision in the "Edit organisation" dialogue (see page 573)
  - Provision in the organisation zoom view (see page 578)
  - Deployment in the organisations dashboard (see page 580)
- Translation of the Dynamic Field (see page 587)

13.3.3.1 Create dynamic field

Create a dynamic field of the type "Text" with the following configuration. The values given are based on the application example above. You can adapt the values to your specific needs. You can read about the basics of creating dynamic fields under Dynamic fields (see page 274) or Object and field types of dynamic fields (see page 281) .

Fig.: The configuration of the dynamic field

Procedure:

1. In the Admin Module, navigate to *System > Dynamic Fields* and click on "New Field".
2. In the dialogue that opens, enter the following configuration:
    1. Name: ShortName
       Tip: Make a note of the name. You will need it for the integration into the interfaces.
    2. Label: Short name
    3. Field Type: Text
    4. Object Type: Organisation
    5. Show in Customer Portal: no/disabled
    6. Validity: valid
    7. Configuration:
        a. Number (min): 0

    b.   Number (max): 1

    c.   Number (default): 0

    d.   Single Entry Separator: Empty

    e.   Default value: empty

    f.   RegEx list: empty (or as required)

13.3.3.2   Provide dynamic field

After creating the dynamic field, it can be made available in the dialogue "Edit organisation". This is done in KIX Start by configuring the SysConfig key. Each context in which the field is included has its own key.

- **For KIX Start users:** For easier editing, we recommend using an external JSON editor (e.g. www.jsonformatter.io[15]). Copy the value of the key into the editor and edit it there. Minimise the edited value in the editor and copy it back to KIX.
- **For KIX Pro users:** You can use the integrated JSON editor in the menu *System > GUI Configuration > Agent Portal*. This eliminates the need to use the external JSON editor, including the steps to copy the value back and forth.

Further information can also be found under: Integrating a Dynamic Field (see page 423) .

Provision in the "Edit organisation" dialogue

Configuration key:	organisation-edit-form-group-information

In order for an agent to be able to store the short name at the organisation, the dynamic field is integrated into the "Edit organisation" dialogue. The initial configuration of this dialogue is firmly anchored in the backend. Changes to the configuration therefore require the addition of the complete set of form fields.

---

15 http://www.jsonformatter.io

Fig.: The Dynamic Field in the "Edit Organisation" Dialogue

Procedure:

1. Navigate to *System > SysConfig*. Open the key "*organisation-edit-form-group-information*".
   The configuration of this key defines the structure and content of the "Edit organisation" dialogue.
2. Replace the value of the key as follows:
   1. Copy the following source code into a JSON editor.
   2. In the maximised view of the editor, remove the comments.
   3. Make further changes to the code as required, e.g. change the labels or add more Dynamic Fields.
   4. **Info**: The order of the `fieldConfigurationIDs` determines the order of the form fields in the dialogue.

```
 1    {
 2      "id": "organisation-edit-form-group-information",
 3      "name": "Translatable#Organisation Information",
 4      "fieldConfigurationIds": [
 5        "organisation-edit-form-field-cno",
 6        "organisation-edit-form-field-name",
 7        "organisation-edit-form-field-Shortname", // ID of the
      dynamic field in the key
 8        "organisation-edit-form-field-type",
 9        "organisation-edit-form-field-url"
10      ],
11      "separatorString": null,
12      "formFields": [
13
14        // Integration of the dynamic field ShortName
```

```
15        {
16          "id": "organisation-edit-form-field-Shortname",
17          "property": "DynamicFields",
18          "required": false,
19          "options": [
20            {
21              "option": "FIELD_NAME",
22              "value": "ShortName"  //Name of the dynamic field
   (s. Configuration)
23            }
24          ],
25          "showLabel": true,
26          "type": "FormField",
27          "visible": true,
28          "translateLabel": true,
29          "instanceId": null
30        }, //End of of the dynamic field
31        //The following are all other form fields analogous to the
   initial configuration
32        {
33          "id": "organisation-new-form-field-MailDomain",
34          "property": "DynamicFields",
35          "required": false,
36          "options": [
37            {
38              "option": "FIELD_NAME",
39              "value": "MailDomain"
40            }
41          ],
42          "showLabel": true,
43          "type": "FormField",
44          "visible": true,
45          "translateLabel": true,
46          "instanceId": null
47        },
48        {
49          "id": "organisation-new-form-field-IfsVersion",
50          "property": "DynamicFields",
51          "required": false,
52          "options": [
53            {
54              "option": "FIELD_NAME",
55              "value": "IfsVersion"
56            }
57          ],
58          "showLabel": true,
59          "type": "FormField",
60          "visible": true,
61          "translateLabel": true,
62          "instanceId": null
63        },
64        {
65          "id": "organisation-new-form-field-IfsCloudYN",
```

```
 66          "property": "DynamicFields",
 67          "required": false,
 68          "options": [
 69            {
 70              "option": "FIELD_NAME",
 71              "value": "IfsCloudYN"
 72            }
 73          ],
 74          "showLabel": true,
 75          "type": "FormField",
 76          "visible": true,
 77          "translateLabel": true,
 78          "instanceId": null
 79        },
 80        {
 81          "id": "organisation-new-form-field-KeyAccountEmail",
 82          "property": "DynamicFields",
 83          "required": false,
 84          "options": [
 85            {
 86              "option": "FIELD_NAME",
 87              "value": "KeyAccountEmail"
 88            }
 89          ],
 90          "showLabel": true,
 91          "type": "FormField",
 92          "visible": true,
 93          "translateLabel": true,
 94          "instanceId": null
 95        },
 96        {
 97          "id": "organisation-new-form-field-ProjectManagerEmail",
 98          "property": "DynamicFields",
 99          "required": false,
100          "options": [
101            {
102              "option": "FIELD_NAME",
103              "value": "ProjectManagerEmail"
104            }
105          ],
106          "showLabel": true,
107          "type": "FormField",
108          "visible": true,
109          "translateLabel": true,
110          "instanceId": null
111        },
112        {
113          "id": "organisation-new-form-field-CustomerContactEmail",
114          "property": "DynamicFields",
115          "required": false,
116          "options": [
117            {
118              "option": "FIELD_NAME",
```

```
119              "value": "CustomerContactEmail"
120            }
121          ],
122          "showLabel": true,
123          "type": "FormField",
124          "visible": true,
125          "translateLabel": true,
126          "instanceId": null
127        },
128        {
129          "id": "organisation-new-form-field-CustomerContactPhone",
130          "property": "DynamicFields",
131          "required": false,
132          "options": [
133            {
134              "option": "FIELD_NAME",
135              "value": "CustomerContactPhone"
136            }
137          ],
138          "showLabel": true,
139          "type": "FormField",
140          "visible": true,
141          "translateLabel": true,
142          "instanceId": null
143        }
144      ],
145      "draggableFields": false,
146      "type": "FormGroup"
147    }
```

3. Minimise the source code in the editor to remove unnecessary spaces and line breaks and copy the source code to the clipboard.

4. Paste the source code from the clipboard back into KIX into the field "Value" and apply the changes with "Save".

5. Click on "Reload Frontend Configuration" so that the dynamic field is displayed in the "Edit Organisation" dialogue.

Provision in the organisation zoom view

Configuration key:	organisation-details-info-widget

If the short name is stored for an organisation, it can be displayed in the zoom view of the organisation. For this purpose, the dynamic field is added to the configuration of the widget "Organisation Information".

In the example, the short name is placed below the name of the organisation. You can also place the dynamic field elsewhere in the widget.

My Organisation (MY_ORGA)

Organisation Information

CNO	Name	Type	Validity
MY_ORGA	My Organisation	service provider	valid
	Short Name		
	MyOrg		

Created at	Created by	Changed at	Changed by
04/03/2023, 04:42 PM	not assigned	04/04/2023, 12:25 PM	Agent [TD]hhaupt
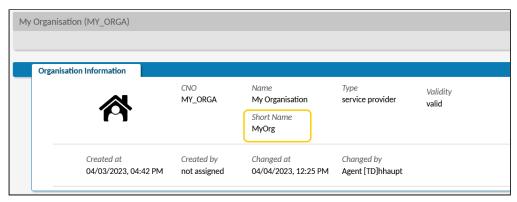
Fig.: The value of the dynamic field in the detailed view of the organisation

Procedure:

1. Navigate to *System > SysConfig*. Open the key "*organisation-details-info-widget*".
   The configuration of this key defines the structure and content of the "organisation-info" dialogue.

2. Copy the value of the key into a JSON editor.

3. In the maximised view of the editor, paste the following code block. Place it there within the `values[...]` block at the desired position.
   Info: Individual content blocks are enclosed in square brackets **[...]**). The elements to be displayed in them are enclosed in curly brackets (**{...}**). Several elements are separated by commas. For further information, see also The object-information-card-widget (see page 355) .

```
{
   "componentId": "object-avatar-label",
   "componentData": {
      "property": "DynamicFields.ShortName"
      }
}
```

```
{
    "id": "organisation-details-info-widget",
    "name": "Organisation Info Widget",
    "type": "Widget",
    "widgetId": "object-information-card-widget",
    "title": "Translatable#Organisation Information",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
        "id": "1649937794814",
        "name": "1649937794814",
        "type": null,
        "avatar": [],
        "rows": [
            {
                "style": "",
                "separator": true,
                "values": [
                    [⊂⊃],
                    [⊂⊃],
                    [
                        {
                            "componentId": "object-avatar-label",
                            "componentData": {
                                "property": "Name"
                            }
                        },
                        {
                            "componentId": "object-avatar-label",
                            "componentData": {
                                "property": "DynamicFields.ShortName"
                            }
                        }
                    ],
                    [
                        {
                            "componentId": "object-avatar-label",
                            "componentData": {
                                "property": "DynamicFields.Type"
                            }
                        }
                    ],
```

4. Minimise the source code in the editor to remove unnecessary spaces and line breaks and copy the source code to the clipboard.

5. Paste the source code from the clipboard back into KIX into the field "Value" and apply the changes with "Save".

6. Click on "Reload Frontend Configuration" so that the value of the dynamic field is displayed in the detailed view.

Deployment in the organisations dashboard

Configuration key:	customer-dashboard-table
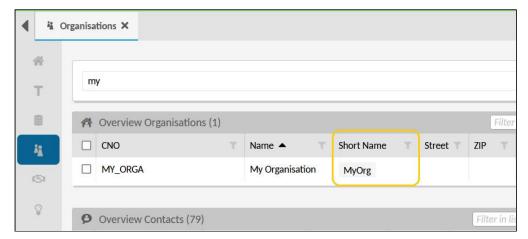
Fig.: The dynamic field in the overview of organisations

In order for the short name stored at the organisation to be displayed in the overview of organisations, an additional column must be added to the table. For this purpose, the entire set of table columns is added to the initial table configuration in the SysConfig key.

- **For KIX Start users:** For easier editing, we recommend using an external JSON editor (e.g. www.jsonformatter.io[16]). Copy the value of the key into the editor and edit it there. Minimise the edited value in the editor and copy it back to KIX.
- **For KIX Pro users:** You can use the integrated JSON editor in the menu *System > GUI Configuration > Agent Portal*. This eliminates the need to use the external JSON editor, including the steps to copy the value back and forth.

Further information can also be found under:Displaying Values of Dynamic Fields (see page 442) .

Procedure:

1. Navigate to *System > SysConfig* and open the key "*customer-dashboard-table*".
   In this key, the structure of the overview of the organisations in the organisations dashboard is configured.

2. Copy the following source code into a JSON editor. In the maximised view, you can remove the comments and edit the code further if necessary, e.g. to define additional table columns.
   Info: Each table column is enclosed in curly brackets. Several table columns are separated by commas.

```
{
  "id": "customer-dashboard-table",
  "name": "Organisations Table",
```

---

16 http://www.jsonformatter.io

```
"type": "Table",
"objectType": "Organisation",
"loadingOptions": null,
"displayLimit": 10,
//Definition of table columns
 "tableColumns": [
  {
    "id": null,
    "name": null,
    "type": null,
    "property": "Number",
    "showText": true,
    "showIcon": false,
    "showColumnTitle": true,
    "showColumnIcon": false,
    "sortable": true,
    "filterable": true,
    "hasListFilter": true,
    "resizable": true,
    "defaultText": null,
    "translatable": true,
    "titleTranslatable": true
  },
  {
    "id": null,
    "name": null,
    "type": null,
    "property": "Name",
    "showText": true,
    "showIcon": false,
    "showColumnTitle": true,
    "showColumnIcon": false,
    "sortable": true,
    "filterable": true,
    "hasListFilter": true,
    "resizable": true,
    "defaultText": null,
    "translatable": true,
    "titleTranslatable": true
  },
  // Additional table column for displaying the "ShortName"
  {
    "id": null,
    "name": null,
    "type": null,
    "property": "DynamicFields.ShortName",
    "showText": true,
    "showIcon": false,
    "showColumnTitle": true,
    "showColumnIcon": false,
    "size": 50,
    "sortable": true,
```

```
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "label-list-cell-content",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    },
    // End of tableColumn
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "Street",
      "showText": true,
      "showIcon": false,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 150,
      "sortable": true,
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "label-list-cell-content",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    },
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "Zip",
      "showText": true,
      "showIcon": false,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 30,
      "sortable": true,
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "label-list-cell-content",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
```

```
    },
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "City",
      "showText": true,
      "showIcon": false,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 100,
      "sortable": true,
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "label-list-cell-content",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    },
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "Country",
      "showText": true,
      "showIcon": false,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 50,
      "sortable": true,
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "label-list-cell-content",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    },
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "ValidID",
      "showText": true,
      "showIcon": false,
      "showColumnTitle": true,
      "showColumnIcon": false,
```

```
        "size": 50,
        "sortable": true,
        "filterable": true,
        "hasListFilter": true,
        "dataType": "STRING",
        "resizable": true,
        "componentId": "label-list-cell-content",
        "defaultText": null,
        "translatable": true,
        "titleTranslatable": true,
        "useObjectServiceForFilter": false
    }
  ],
  "tableColumnConfigurations": null,
  "enableSelection": true,
  "toggle": false,
  "headerHeight": 2.25,
  "rowHeight": 1.75,
  "emptyResultHint": "Translatable#0 data sets found.",
  "fixedFirstColumn": false,
  "additionalTableObjectsHandler": [],
  "intersection": true,
  "searchId": null
}
```

```
{
    "id": "customer-dashboard-table",
    "name": "Organisations Table",
    "type": "Table",
    "objectType": "Organisation",
    "loadingOptions": null,
    "displayLimit": 10,
    "tableColumns": [
        {⌷},
        {⌷},
        {
            "id": null,
            "name": null,
            "type": null,
            "property": "DynamicFields.ShortName",
            "showText": true,
            "showIcon": false,
            "showColumnTitle": true,
            "showColumnIcon": false,
            "size": 50,
            "sortable": true,
            "filterable": true,
            "hasListFilter": true,
            "dataType": "STRING",
            "resizable": true,
            "componentId": "label-list-cell-content",
            "defaultText": null,
            "translatable": true,
            "titleTranslatable": true,
            "useObjectServiceForFilter": false
        },
        {
            "id": null,
            "name": null,
            "type": null,
            "property": "Street",
            "showText": true,
```

3. Minimise the source code in the editor to remove unnecessary spaces and line breaks and copy the source code to the clipboard.

4. Paste the source code from the clipboard back into the "Value" field and apply the changes with "Save".

5. Click on "Load Frontend Configuration" to update the display in the organisation dashboard.

13.3.3.3   Translation of the Dynamic Field

When creating the dynamic field, a name was specified (ShortName). This name is used in the interfaces as a label in front of the input field. You can have the label translated by KIX by creating a pattern for it. For more information on patterns, see: Translations (see page 264) .
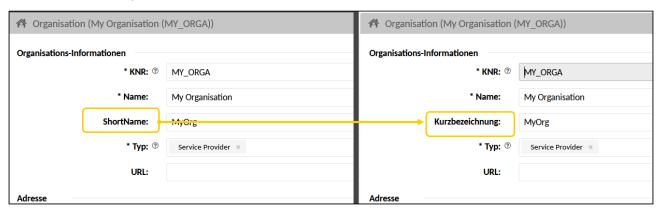
Fig.: The label before and after adding the pattern

Procedure:

1. Navigate to *Internationalisation > Translations* and click on "New".

2. Enter the following parameters:
    1. Pattern: ShortName (or the label you use).
        The pattern defines the word to be translated
    2. English: Short Name (or the English label)
        The English translation of the pattern must be entered
    3. German: Short Name (or the label you use)
        Enter the German translation of the pattern.

3. Save the pattern and, if necessary, refresh the browser with F5.

13.3.4 Control the search for organisations and contacts

13.3.4.1 Using of wildcards

SysConfig key	Description	Possible Values
ContactSearch::UseWildcardPrefix	Enables/disables the wildcard prefix for contact search	0 | 1
OrganisationSearch::UseWildcardPrefix	Enables/disables the wildcard prefix for organisation search	0 | 1

For a better user experience when searching for organizations and contacts, it is not necessary to use wildcards (*). It is sufficient to just enter the search term in the search field. Internally, KIX automatically places an asterisk after the search term (wildcard suffix). The same applies to the autocomplete fields.

The asterisk in front of the search term (wildcard prefix) is not set automatically for performance reasons. It can be set manually when entering the search term.

For on-premises (locally) used KIX instances based on PostgreSQL databases or environments with a manageable number of organizations and contacts, KIX can be set up in such a way that the wildcard prefix is set automatically:

1. Navigate to *System > Sysconfig.*
2. Look for the keys one by one:
   "ContactSearch::UseWildcardPrefix" (setting for contact search)
   "OrganizationSearch::UseWildcardPrefix" (setting for customers / organizations)
   You can define the settings separately for organizations and contacts as required.
3. Change the value to "1" and save this change.
4. In the SysConfig table, click on "Reload frontend configuration" to transfer the changed search behavior to the agent portal. If necessary, the agents must also update their front end.

If the wildcard prefix is active (value: 1), the search is carried out using the search pattern: <Wildcard prefix><Search term><Wildcard suffix>

Search term	Search pattern	Result
musterma	*musterma*	Max **Musterma**nn, Eva **Musterma**ier

Search term	Search pattern	Result
man	*man*	Martin Hof**man**, Her**man** Munster

If the wildcard prefix is not active (value: 0), the search is carried out using the search pattern: <Search term><Wildcard suffix>

Search term	Search pattern	Result
musterma	musterma*	**Musterma**nn AG, **Musterma**ier Ltd.
man	man*	**Man**fred Lobster, **Man**dy McDonald

13.3.4.2   Combined search for telephone numbers

When using the complex search or when searching for contacts in a ticket, operators (&, +, |, *) can be used to search for several parameters. This creates a search string that is processed by KIX, e.g. "Max+Musterman".

However, the + prefixed to a telephone number for the country code can lead to an incorrect search result in a combined search of e.g. name and telephone number. KIX considers the + in front of the telephone number as an operator and not as part of the telephone number. Thus, a search for "Max" and "+49123456" will result in the search string "Max+49123456" - with the wrong phone number. You can work around this by enabling the use of wildcards in the SysConfig key "ContactSearch::UseWildcardPrefix" (see above). This places a wildcard prefix (* or % at DB level) as a placeholder in front of the searched parameters. The resulting search string "*Max+*49123456" provides a correct result, as all characters preceding the digits are taken into account.

Searching for the telephone number alone, even with a preceding +, on the other hand, is unproblematic.

13.4   Settings for FAQ

The following chapters contain extended configuration options for FAQ:

13.4.1 Linking FAQs and Tickets

Configuration key	• LinkObject::PossibleLink###0203 • LinkObject::PossibleLink###8304

Initially, FAQs and tickets are linked with the "Normal" link type. The "ParentChild" link type is inactive and is not available for selection.

You can activate this type of link if required. To do this, set the validity of the following SysConfig keys to "valid".

- **FAQ object**: *LinkObject::PossibleLink###0203*
- **FAQ article**: *LinkObject::PossibleLink###8304*

After activating, click on "Reload frontend configuration". FAQs can then be linked to tickets with the link types "Parent" or "Child".

13.5 Cross-object settings

The following chapters contain global configuration options that do not only refer to a concrete object.

13.5.1 Individualize display values for objects

Configuration key:	Agent portal	display-value-configuration
	Self Service Portal	ssp-display-value-configuration
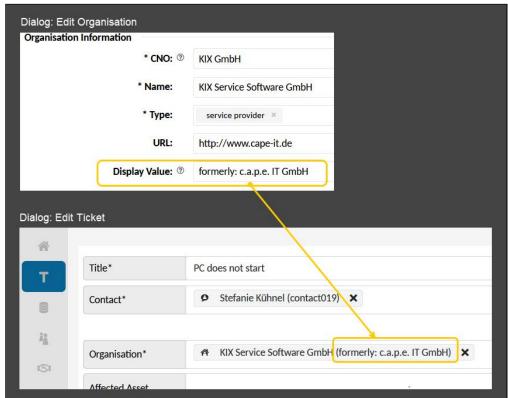
You can configure the display values of the following objects:

- Organisation
- Ticket
- Contact
- Asset (ConfigItem)
- FAQ Item
- User

This allows you to define what information makes up the label representations of the items. This information can consist of a combination of alphanumeric characters and KIX placeholders for attribute values and dynamic fields.

Thus you can:

- adjust the selection value of an organisation so that the value of a dynamic field is displayed instead of a technical customer number.
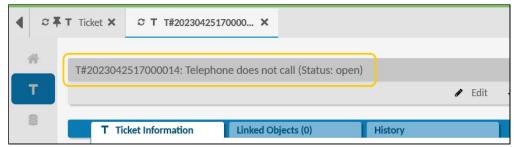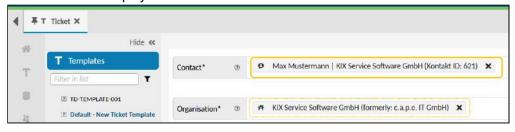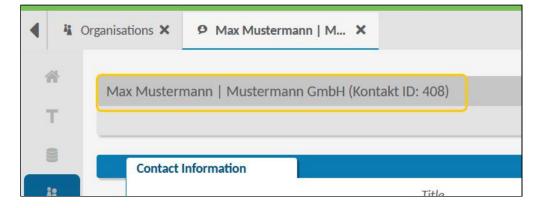
Prerequisites for this example:

- A dynamic field with object type "Organisation" must exist.
- The dynamic field must be integrated into the interface of the organisation.
  - SysConfig key: organisation-edit-form-group-information
  - see also: Integrating a Dynamic Field
- The dynamic field should be assigned a value so that it is displayed.

- Individualise the label of tickets:

- Individualise the display value of a contact:

13.5.1.1 Configuration

The configuration can be done separately for the agent portal and the SSP. There are 2 separate SysConfig keys for this purpose:

- Agent portal: *display-value-configuration*
- SSP: *ssp-display-value-configuration*

Proceed as follows:

- **KIX Start:**
    - In the Admin module, navigate to the *System > SysConfig* menu.
    - Find and open the SysConfig key "*display-value-configuration*" or "*ssp-display-value-configuration*".
        - The pattern for the object "Organisation" is already stored in the key. You can adapt it.
        - If necessary, add further objects to the key and configure their pattern (see also configuration example below).
            - For easier editing, you can copy the value of the key into an external JSON editor (e.g. jsonformatter.io[17]).
            - Copy the changed key back into the "Value" field and save the changes.
    - After saving, click "Reload frontend configurations" to refresh the view in the frontend.

- **KIX Pro:**
    - In the Admin module, navigate to the menu: *System > GUI Configuration > Agent Portal* and/or *Self Service Portal.*
    - In the SysConfig explorer, open the branch "Not Referenced Configurations".
    - Open the SysConfig key "*display-value-configuration*" or "*ssp-display-value-configuration*".
        - The pattern for the object "organisation" is already stored in the key. You can adapt it.
        - If necessary, add further objects to the key and configure the pattern (see also configuration example below).
    - After saving, click on "Reload frontend configurations" to update the view in the frontend.

Parameters

The following parameters must be stored in the configuration:

Parameter	Description
objectType	The object in question (e.g. Ticket, Organisation, Contact, ConfigItem, ConfigItem::Service etc.)

---

17 http://jsonformatter.io

Parameter	Description
pattern	Specification of a character string that forms the display text. The character string • must be in inverted commas • can consist of a series of alphanumeric characters and KIX placeholders. e. g.: " `T#<KIX_TICKET_TicketNumber>:` `<KIX_TICKET_Title> (State: <KIX_TICKET_State>)"` **Attention**: There are not placeholders for all objects!

Each object must be enclosed in curly brackets. Several objects are separated from each other by commas.

Example Configuration of the display values in the agent portal

```
{
  "displayValues": [
    {
      "objectType": "Organisation",
      "pattern": "<KIX_ORGANISATION_Name> (<KIX_ORG_DynamicField_DisplayValue>)"
    },
    {
      "objectType": "Ticket",
      "pattern": "T#<KIX_TICKET_TicketNumber>: <KIX_TICKET_Title> (State:
<KIX_TICKET_State>)"
    },
    {
      "objectType": "Contact",
      "pattern": "<KIX_CONTACT_Firstname> <KIX_CONTACT_Lastname> |
<KIX_CONTACT_PrimaryOrganisation> (Kontakt ID: <KIX_CONTACT_ID>)"
    },
    {
      "objectType": "ConfigItem",
      "pattern": "<KIX_ASSET_Name> (<KIX_ASSET_SectionGeneral_0_Type_0>)"
    },
    {
      "objectType": "ConfigItem::Service",
      "pattern": "<KIX_ASSET_Name> (<KIX_ASSET_SectionGeneral_0_Type_0> #
<KIX_ASSET_SectionGeneral_0_Criticality_0>)"
    }
  ],
  "id": "display-value-configuration",
  "name": "Display Value Configuration",
  "type": "Agent Portal",
  "valid": true,
```

```
"application": "agent-portal"
```
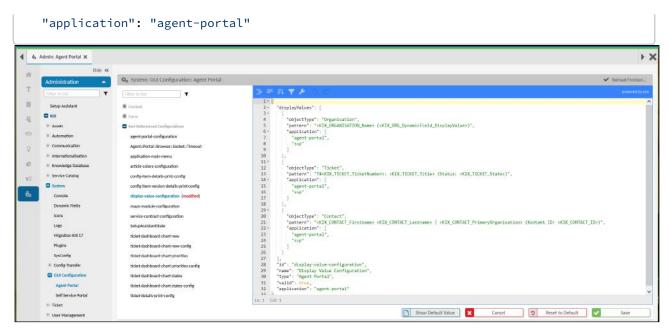

Fig.: Changed configuration of the display values

13.5.2 Configuring the CKEditor

The KIX uses the CKEditor to enter formatted text, e.g. in tickets or articles. The CKEditor comes with a compact set of functions. You can adapt the editor to your own needs.

Content on this page:

13.5.2.1 Enable/disable spell check
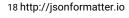
Configuration Key	· agent-portal-configuration · self-service-portal-configuration

By default, the CKEditor contains a spell checker plug-in (SCAYT), which transmits the message content to an external SCAYT server and is therefore not DSGVO compliant. Therefore, KIX is shipped with the SCAYT plug-in disabled.

If necessary, you can reactivate the spell checker for each portal separately and on your own responsibility:

1. Navigate in the admin module to
    1. KIX Start: *System > SysConfig*
    2. KIX Pro: *System > GUI Configuration > Agent Portal* or *Self Service Portal*.
2. Open one of the two configuration keys mentioned above for editing, as required.
    1. KIX Start: If necessary, copy the value of the key into an external JSON editor for better editing (e.g. jsonformatter.io[18]).
3. Navigate to the section " `ckEditorConfiguration` ". This contains the configuration of the CKEditor.
    1. If necessary, collapse the " `toolbar` " code section for a better overview.
4. At the end of the " `ckEditorConfiguration` " section, add the following key-value pair:

`"scayt_autoStartup": true`

---

18 http://jsonformatter.io

```
    "adminRoleIds": [▭],
    "id": "agent-portal-configuration",
    "name": "Agent Portal Configuration",
    "type": "Agent Portal",
    "valid": true,
    "ckEditorConfiguration": {
        "language": "en",
        "toolbar": [▭],
        "resize_minWidth": 200,
        "resize_minHeight": 125,
        "resize_maxWidth": 1200,
        "resize_maxHeight": 2000,
        "autoGrow_minHeight": 125,
        "autoGrow_maxHeight": 2000,
        "autoGrow_onStartup": true,
        "extraAllowedContent": "b; h1 h2 h3 ul li; div[type]{*}; img[*]{*}; col[width]; style[
            *]{*}; *[id](*)",
        "toolbarCanCollapse": true,
        "removeButtons": "",
        "removePlugins": "pastefromexcel,elementspath",
        "extraPlugins": "emoji",
        "codeSnippet_theme": "github",
        "scayt_autoStartup": true
    },
    "application": "agent-portal"
}
```

5. KIX Start: Minimize the view in the external JSON editor and copy the code back to the value of the SysConfig key.

6. Finally, save the changes to the configuration.

7. Click "Reload frontend configuration" to enable spell checking in the CKEditor.

In our forum you can also find hints on how to disable/enable the spell checker in CKEditor:

- https://forum.kixdesk.com/index.php?topic=11914.msg17280#msg17280
- https://forum.kixdesk.com/index.php?topic=12028.0

13.5.2.2   Configure function buttons of the editor

Configuration Key	• agent-portal-configuration • self-service-portal-configuration

KIX delivers the CKEditor with a compact toolbar. You can adjust the editor's default configuration centrally - each for the agent portal and the self-service portal:

1. Navigate in the admin module to
    1. KIX Start: *System > SysConfig*
    2. KIX Pro: *System > GUI Configuration > Agent Portal* or *Self Service Portal*.

2. Open one of the two configuration keys mentioned above for editing, as required.
    1. KIX Start: If necessary, copy the value of the key into an external JSON editor for better editing (e.g. jsonformatter.io[19]).

3. The section `"ckEditorConfiguration"` contains the configuration of the CKEditor.
    You can customize the configuration as needed.
    For information about the configuration options, refer to the English-language documentation of the

---

19 http://jsonformatter.io

CKEditor:

https://ckeditor.com/docs/ckeditor4/latest/features/toolbar.html.

4. Finally, save the changes to the configuration.
5. Click "Reload frontend configuration" to refresh the view in the CKEditor.

13.5.2.3 Configure display in editor body

Configuration Key	Frontend::RichText::DefaultCSS

You can configure the display of the editor content based on CSS in the SysConfig key "*Frontend:RichText::DefaultCSS*".
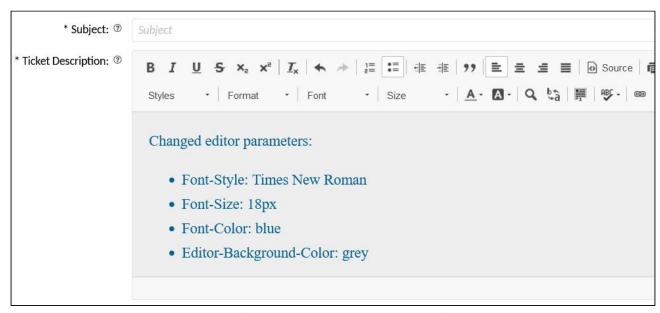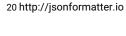
Fig.: CKEditor with changed parameters (see example in section "Selectors")

To change the appearance of the editor:

1. Navigate to *System > SysConfig* and open the configuration key "*Frontend::RichText::DefaultCSS*".
2. If necessary, copy the value of the key into an external JSON editor (e.g. jsonformatter.io[20]) for better editing.
3. Change the parameters specified under "Value" as required and optionally add further selector value pairs to configure individual font styles.
4. Then minimise the source code in the JSON editor and copy it back into the "Value" field of the configuration key.
5. Save your changes and click on "Reload frontend configuration" in the overview of configuration keys to make the changes to the editor effective.
   Afterwards, the display in the editor will be according to your configuration.

---

20 http://jsonformatter.io

Selectors

The JSON string in the configuration key corresponds to an "Array of Hashes" Each hash is a CSS rule (Selector + Value).

Example configuration of the screenshot above

```
[
  {
      "Selector": ".cke_editable",
      "Value": "font-family:Times New Roman; font-size:18px; color:#096392;
background-color:#eee;"
  }
]
```

In CSS, the order of the specifications is relevant. Subsequent style specifications overwrite the previous ones.

In the following code example, the original blue text colour is overwritten by red. All paragraphs in the editor text are thus not in blue but in red font colour.

Example: Style order

```
[
  {
    "Selector": ".cke_editable",
    "Value": "font-family:Geneva,Helvetica,Arial,sans-serif; font-size: 12px;
color:blue;"
  },
  {
    "Selector": "p",
    "Value": "color:red;"
  }
]
```

If the CSS rule stored in the configuration does not work, you can check whether the selector can be addressed in a more targeted way.

You can also use " `>` " to address selectors within other selectors.

Example: In a table, a <p> tag (paragraph) should be formatted differently than in normal continuous text. However, since the <p> tag is already defined by the global configuration of the editor, a change to the <p> tag would also affect all other <p> tags. By specifying " `td > p` ", however, paragraphs within a table cell can be addressed directly.

Example: Paragraph in a table cell

```
{
    "Selector": "td > p",
    "Value": "font-size: 10px;"
}
```

Overview of possible selectors

Selector	Description	Notes
.cke_editable	Body of the editor	• General style specifications such as editor background or font style, font size, font colour etc. (see example above). • Use the selector `.cke_editable` for the <body>, otherwise it will be overwritten by the standard CSS of the CKEditor..
h1	Headline 1	
h2	Headline 2	
h3	Headline3	
h4	Headline 4	
p	Paragraph	
a	link	
tt	Typewriter	
pre	Formatted	E.g. for source code
big	big type size	relative indication
small	small type size	relative indication

Selector	Description	Notes
ins	Inserted Text	
.marker	Marker	With a preceding dot! It is a class.
table	Table	
tr	Table rowl	
td	Table cell	
th	Table header	
div	Special Container	Content block / Container
code	Computer-Code	
span	Area	E.g. for special formatting
var	Variable	
kdb	Keyboard Phrase	E.g. user input via keyboard
samp	Sample Text	E.g. output of a programme or part of a programme
q	Inline Quotation	Inline quote (in inverted commas)
cite	Cited Work	Citation; title or author of a work
del	Deleted Text	Strikethrough text

> ✅ **Tipp**
>
> The CKEditor does not support all selectors offered by CSS. To find out which selectors are supported, you can proceed as follows:

1. Write any text in the editor of a new ticket or article.
2. Format the text as desired (style, font, size, etc.).
3. Click on the "Source code" button. This will display your text as plain text.
4. In the displayed source code you can see which selectors are used and how.
5. You can then adjust the selectors and HTML tags accordingly in the configuration.

ⓘ **Info**

Further information on CSS selectors can be found, for example, at: https://wiki.selfhtml.org/wiki/
CSS/Selektoren

13.5.3 Use URL of the browser

Every open tab, every view, every search etc. has a unique URL, which is displayed in the address bar of the browser. This URL is stored in the favourites, for example, when an agent saves their search.

Each object (ticket, contact, organisation, view, search, etc.) can be called up directly using the URL. For example, calling up the ticket with the ID "123": `http://your.agentenportal.kix.cloud/tickets/123`.

This allows you to use the URL from the address bar of your browser to call up the current view in a different context. For example,

- send URL to other agents by email
- have URL called up by external applications or services
- use URL as a hyperlink in a widget (see page 473) , so that a search or an object (e.g. the contact) can be called up directly.

At the end of the URL you will also find <u>the ID of the currently open object</u> (contact/organisation/ticket etc.). You can use the ID, for example, in placeholders, when configuring roles, etc. to reference a specific object directly.
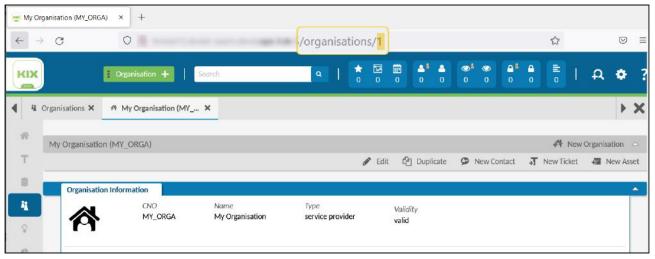
Fig.: The number of current organisation in the browser URL

13.5.3.1 Resolve contacts using the URL

The ticket creation screen can be called up from external applications or services. This is done by calling up the corresponding URL (e.g. `http://localhost:20001/tickets?new`). If additional parameters are added to the URL, the contact can be set on the new ticket and the desired ticket template can be selected.

Example: When a call is accepted, the telephone system or telephone client calls up the KIX URL and provides the URL with information about the caller. KIX can resolve this information and use it to open the desired ticket template and set the corresponding contact on the new ticket.

Prerequisite: The telephone system supports the calling of URLs.

The URL for the call must look like this:

- KIX.cloud: `http://your.agentenportal.kix.cloud/tickets? new&from=0123456789&templateId=2`
- KIX On-Premises: `http://localhost:20001/tickets? new&from=0123456789&templateId=2`

Several parameters are strung together with "&".

Parameters	Description	Example
from	Requests information about the contact, e.g. the telephone number. Starts a full text search in the contact. The identified contact is entered in the new ticket to be created. If several contacts are found, the first contact found is always entered on the ticket.	http://localhost:20001/tickets? new**&from=0123456789**
templateId	ID of the ticket template that is loaded when the URL is called up. Tip: To determine the ID, navigate to Workflow > Templates. Move the mouse pointer over the desired template and read the ID in the browser footer.	http://localhost:20001/tickets? new&from=0123456789 **&templateId=2**

13.6   Display of support information

KIX enables the display of important data and variables in the system and thus the identification of possible sources of error.

As part of your support contract or upon explicit request, you can send this information directly from the system to our support team so that our employees can help you analyse it.

13.6.1   Show support information

Console command	Console::Command::Admin::SupportData::Collect

You can display the support information in the KIX console in order to analyse it.

- In the Admin module, navigate to *System > Console*.
- Call up the command " `Console::Command::Admin::SupportData::Collect` ".
- After clicking on "Execute", the support data is displayed in the console. Depending on the amount of data, this may take a moment.

13.6.2   Sending support information

Console command	Console::Command::Admin::SupportData::Collect --send

KIX can send the support information directly from the system to our support team so that we can assist you with the analysis. The prerequisite for this is that you have a valid support contract with us.

⚠️ **Shipping only after prior consultation with our support!**

- In the Admin module, navigate to *System > Console*.
- Call up the command " `Console::Command::Admin::SupportData::Collect` ".
- Enter the parameter `--send` in the text field.
- After clicking on "Execute", the support information is sent to the e-mail address stored in the system (see below).

13.6.3   Send support information automatically

SysConfig key	Daemon::SchedulerCronTaskManager::Task###SendSupportData

If you have a corresponding contractual agreement with us, you can activate a cron job that triggers the regular sending of support information.

⚠️ **Activation of the cron job only after prior consultation with our support!**

- In the Admin module, navigate to *System > SysConfig*.
- Search for and open the key
  "`Daemon::SchedulerCronTaskManager::Task###SendSupportData`".
- Set the key to "valid".
- Optional and in consultation: Change the execution interval if necessary.
  Initially, execution takes place every Monday, 04:00 (Schedule: "04" "1").
- Save the change.
  KIX then sends the support information at regular intervals to the recipient address stored in the system (see below).

13.6.4 Change recipient address

SysConfig key	SupportData

Initially, the support information is sent to the e-mail address of our support team: support@kixdesk.com[21].
In consultation with our support team, you can enter a different e-mail address as the recipient in the SysConfig key "*SupportData*".

The specified e-mail address is used by both the cron job and the console.

---

21 mailto:support@kixdesk.com

14 Ticket

A ticket is created every time a service request or similar is received via email or an agent creates a case, also known as a process. Tickets are similar to an index card and are designed to map a process in its entirety.

Tickets can be created for a wide variety of reasons, e.g. if a machine needs to be repaired, a printer is out of toner, an elevator is faulty, spare parts need to be ordered, and many more. Different **teams** are responsible for the different remits. To enable the ticket to be assigned to the team responsible for the process, the ticket can be created in a certain team folder either automatically when the email is received or manually by an agent. All tickets are saved one after the other in a team folder. This creates a type of queue where the tickets are sorted by priority and age.

A ticket initially contains the topic/subject line of the process and the customer's contact details, such as contact partner, email, telephone, etc. Each ticket also has an employee responsible for it and an owner, who may be the same person. When the ticket – and thus the process – is edited, further information is generated as a result, and this information is then added to the ticket in the form of **articles**. For example, this could be email correspondence with and without attachments, the agent's own notes, notes from other agents, data sheets, etc. In this way, the "index card" is transformed into a complete "folder" that maps the entire process. If the process is complete, the ticket, too, will be closed.

The sub-menus in the menu Ticket enable you to configure the basic settings for tickets. You are able to create teams so that the tickets can be assigned to the relevant remits. You can create ticket priorities, ticket types, and ticket states so that the tickets can be identified by properties. You can also create text modules to reduce the amount of typing the agent has to do.

All the settings you configure here are available to the agents when editing tickets. However, KIX also generates all analyses, tables, overviews, etc. in accordance with the settings configured here.

ⓘ **Note**

When creating tickets via the API (e.g. with Insomnia), the request must contain the mandatory fields on the ticket. If the ContactID is missing, the ticket creator is set as the contact and its PrimaryOrg as the organisation. If no contact is assigned to the user, a message will be displayed. Furthermore, you can store default values for team, priority and status in the SysConfig keys, which are automatically set if information is missing (see table below).

The menu Ticket has the following sub-menus:

Menu	Description	SysConfig keys for default values
Priorities	Create and manage ticket priorities in order to identify how urgent the tickets are.	*Ticket::Priority::Default*
States	Create and manage ticket states in order to identify the extent to which a ticket has been processed.	*Ticket::State::Default*
Teams	Create and manage a team structure in order to assign tickets to certain teams.	*Ticket::Queue::Default*
Text Modules	Create and manage text modules that can be used to quickly enter recurring texts in the agent portal.	
Types	Create and manage ticket types in order to assign tickets to a type of process.	

14.1 Priorities

Under the menu item *Ticket > Priorities* you can create and manage priorities for tickets. With priorities, the urgency level of tickets is marked in colour and hierarchically. The table in the content area lists all priorities created in the system.

The priority of a ticket can be assigned manually by an agent when creating/editing a ticket, as well as automatically via jobs or email filters .

Agents can sort and filter the overview tables in the dashboards according to the priorities and also search for tickets with a certain priority in the complex search.
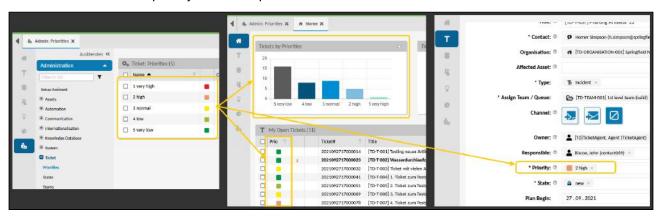
Fig.: Maintained Priorities Shown in Home Dashboard and When Editing a Ticket

KIX comes supplied with 5 priorities. You can change these priorities, including their images, to suit your own requirements.

Levels	Classification
1	Very high (red)
2	High (orange)
3	Normal (yellow)
4	Low (light green)
5	Very low (green)

> ✅ **Tip**
>
> The colour coding of the priorities in the chart widget of the Home Dashboard can be configured in the SysConfig key "*home-dashboard-ticket-chart-widget-priorities-config*".

14.1.1 Creating a Priority

KIX comes supplied with five priorities, ranging from 1 (very high) to 5 (very low). You are able to adapt these priorities (levels of urgency) to suit your requirements by, for example, changing the sequence or adding other priorities. Any changes made to priorities immediately take effect for all tickets.

> ⚠️ **Important**
>
> Always start the name with a number as the sorting function in the tables and drop-downs is based on this prefixed number.

To create a new priority, proceed as follows:

1. In explorer, navigate to *Ticket > Prioritie*s. In the content area, a table listing all the priorities created in the system opens.
2. In table, click "New Priority". A form dialog for creating the new priority opens.
3. Complete form, select icon if required, and set validity to "valid".
4. Click "Save" to save new priority.

The new priority has now been created and can be assigned to new tickets.

To edit a priority, proceed as follows:

1. In explorer, navigate to *Ticket > Priorities.* In the content area, a table listing all the priorities created in the system opens.
2. In table, click priority to be changed. A form dialog for editing the priority opens.
3. Amend information as required and click "Save" to apply changes.

The changes now take effect for all tickets.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Enter a short name for the priority to ensure the name is clearly shown in the widget "Übersicht Ticketpriorität" (Overview of Ticket Priorities).

Field (selection)	Description
Icon	The icon is used to graphically identify the priority in the table overviews ("Prio" column).
	You can choose an icon from one of the provided libraries or upload a graphic file.
Validity	• **valid:** The priority is active and can be assigned to tickets.
	• **invalid/invalid-temporarily:** The priority cannot be assigned to a ticket. The priority will be retained for existing tickets so that they temporarily: can continue to be processed. If the ticket information is opened for editing, a red border around the field will highlight if a priority is missing. A new, valid priority can be selected.

14.2   States

The menu *Ticket > States* enables you to create and manage the state for the tickets. The state indicates the extent to which a ticket has been processed, e.g. new, open, closed, etc. This enables agents to identify at any time which tickets still need to be processed, which ones are pending, and which have been completed.

The state of a ticket can be assigned both manually by an agent when creating/editing a ticket and automatically using jobs or email filters.

KIX comes supplied with a range of different states. You can narrow down the ticket status even further by creating your own states and assigning them to an existing state type. For example, the state type "closed" could be supplemented with an additional state "closed successful". Agents can select from the states created here when creating new tickets, can filter and sort tables by these states, and can search for tickets with a certain state in Advanced Searches.
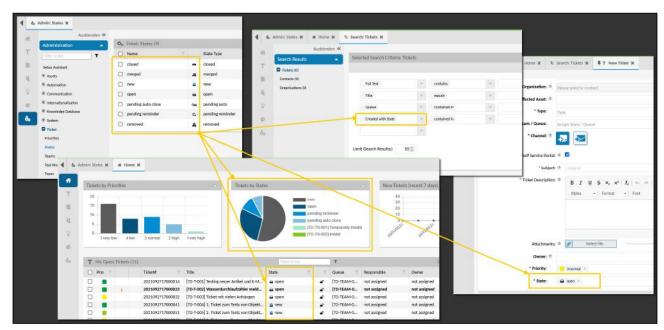
Fig.: States in Ticket Dashboard, When Creating New Tickets, and in Advanced Searches

> ### ⓘ Info
>
> If a ticket is assigned the state "pending [...]", a dynamic time in the future will be shown as a default value for the target waiting time. This default value is initially determined from the current time plus the value saved in the SysConfig. You can adapt the default value for the target waiting time by changing the key "*Ticket::Frontend::PendingDiffTime*" in the menu *System > SysConfig*.
> **Please note:** A target waiting time must be specified in the ticket to ensure the ticket can be saved even without manually changing the time information.

14.2.1 Creating or Editing a Ticket State

The state indicates the extent to which a ticket has been processed. KIX comes supplied with a range of different states that you can adapt, reduce, or expand to suit your specific requirements (see table below). Any changes made to a state immediately take effect for all tickets.

To create a new state, proceed as follows:

1. In explorer, navigate to *Ticket > States*. In the content area, a table listing all states created in the system opens.
2. In table, click "New State". A form dialog for creating a new state opens.
3. Complete form, select icon if required, and set validity to "valid".
4. Click "Save" to save new state.

The state has now been created and can be assigned to tickets.

To edit a state, proceed as follows:

1. In explorer, navigate to *Ticket > States*. In the content area, a table listing all states created in the system opens.
2. In table, click state to be changed. A form dialog for editing the state opens.
3. Amend information as required and click "Save" to apply changes.

The changes now take effect for new tickets. Your changes are not applied to existing tickets.

The form dialog contains the following input fields, among others:

Field (selection)	Description	
Name	Give the state a meaningful, unique name so that agents can sensibly categorize the ticket state.	
State Type	Assign the state to a state type in order to classify the state. You will then be able to evaluate tickets on this basis, e.g. all tickets with the state type "open". The following types can be selected:	
	new	A new ticket has been created from an inbound email.
	open	Default state for tickets assigned to teams and agents.

Field (selection)	Description	
	closed	The ticket has been closed; the process is complete.
	pending reminder	Automatic email reminder to the agent(s) highlighting that the unlock timeout has expired.
	pending auto	Tickets of this status type experience an automatic status change according to the SysConfig key "*Ticket::StateAfterPending*" as soon as the reminder time has been reached. For example, the status "pending auto close" is followed by the status "closed". If there is no entry in the SysConfig key "*Ticket::StateAfterPending*", errors will be noted in the KIX log.
	removed	The customer has removed the ticket; the ticket no longer applies.
	merged	The ticket has been merged with another ticket.
Icon	The icon is used to graphically identify the state in the table overviews (column "State"), and is used in the selection field "State" when creating a new ticket. You can choose an icon from one of the provided libraries or upload a graphic file.	
Validity	valid:	Tickets may be assigned this state.
	invalid/ invalid-temporarily:	The state cannot be assigned to a ticket. The state is retained for existing tickets so that they remain identified as such. If the ticket information is opened for editing, a red border around the field will highlight if another state should be selected

14.3 Teams

In the menu *Ticket > Teams* you can create and manage teams. Teams are also be refferred as queues and can be considered a type of "team inbox". They are used to distribute the tickets to the relevant remits.

KIX comes supplied with a small selection of teams that you can adapt and supplement to suit your own needs. The table in the dashboard lists the teams created in the system. The teams displayed depend on the permissions (not basic permissions) of the administrator.
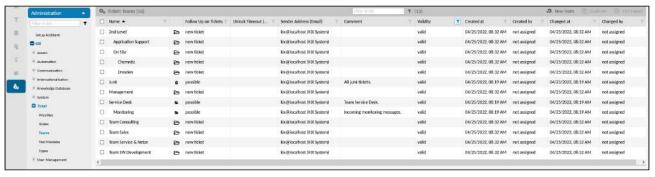
Fig.: Overview of teams

The buttons in the table header allow you to edit and create new teams:

Button	Description
New Team	Opens a form dialogue for creating a new team.
Duplicate	Opens a form dialogue analogous to creating a new team. • is filled with the values of the team from which the action originated. • Exception: Form field "Name" has the content "Copy of [name of duplicated team]". The name can be changed.
CSV-Export	Exports the teams selected with a tick to a CSV file.

Teams can be assigned to a superordinate team. This creates a structure tree of team folders, which is displayed as an explorer in the ticket dashboard of the agent portal. Agents can use it to view and edit the tickets of a team.

In practice, the team structure is often based on the company structure and the teams reflect the individual task areas/departments. The distribution of roles determines which agents have access to which teams and which rights they have. Agents can thus access the teams within the scope of their user rights and process the tickets collected in them. We therefore recommend that you work out an optimal structure tree for your teams in advance. This will help you maintain an overview and make it easier for you to assign rights to the individual teams in the future.
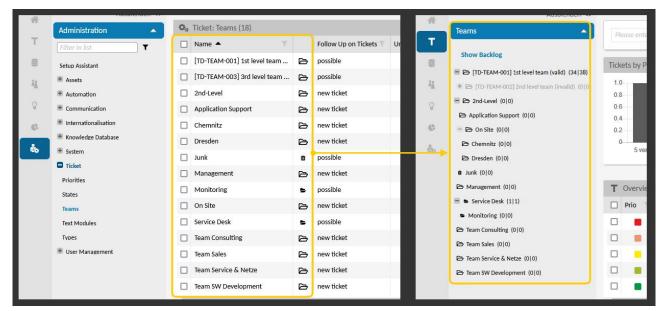
Fig.: The teams that were created can be found in the Ticket Dashboard

Below are some examples of when teams are used:

1. A customer emails a service request to the address support@YourCompany.com. By assigning the email account support@YourCompany.com to the team "Support", a new ticket will be created in the team folder "Support". Agents who have access to the team folder are now able to view and process this service request.

2. A customer replies to an email that was sent from a team. KIX checks the inbound message to see if a ticket number has been provided. If a ticket number has been provided, the email will be converted into an article and appended to the ticket with the corresponding ticket number. If no ticket number has been provided, a new ticket will be created and sent to the team folder that has been linked to the email account. Any email filters that were set will be taken into account when this happens.

3. A customer calls the Customer Service team to report an incident. If this message can be assigned to an existing ticket, the agent creates a new article for that ticket. If a ticket does not exist yet for this message, the agent creates a new ticket and assigns it to a team. Either the agent themselves or other agents can now view and process this ticket, depending on each user's rights and remits.

> ⓘ **Set owner when changing the team**
>
> You can define that the agent is reset to "not assigned" if only the team is changed when editing a ticket. This prevents an incorrect agent from being set in the new team. If the team and the agent are changed at the same time, the set agent will be retained.
> By setting the SysConfig key "*Ticket::EventModulePost###120-ForceOwnerResetOnMove*" to valid/invalid, you can switch this behaviour on or off. By default, it is switched off (invalid).

14.3.1 Creating or Editing a Team

When delivered, KXI contains a small selection of teams that you can use directly, change and add additional teams. Assigning a team to a higher-level team creates a tree structure of hierarchically ordered teams. As a rule, this structure reflects the hierarchy of departments or areas of responsibility in the company.
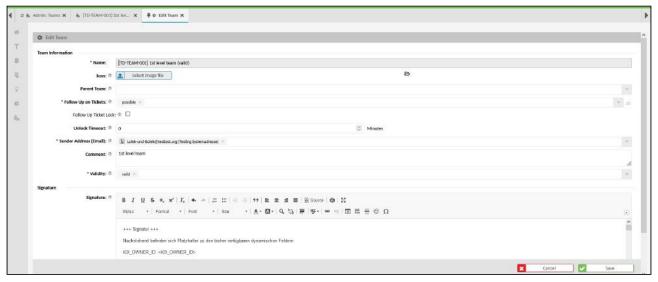
Fig.: Editing a Team

Preparation:

Assigning a ticket to a team can be done in different ways. For example by:

1. manual assignment when creating or editing a ticket
2. automatic assignment by a job (see page 170) (Macro Action: Set Team)
3. automatic allocation when retrieving an email account.

For the automatic distribution of tickets when an email account is retrieved, the inbox, outbox and email filters must be configured.

Inbox

Each team is able to receive emails from different sources. One way this can be done is by assigning one or more email accounts to a team. Another way is to use filter rules to manage how emails are automatically distributed to the individual teams.

Outbox

Each team has its own email address. This is the sender address for all emails sent from the team when processing tickets. Generally speaking, the inbox of this address is directed to the team. In addition to this, each team can also have its own email signature (actual signature, contact details, legal notice, etc.). This signature is then used on all emails sent from the team. The signature is stored when a team is created.

Therefore, before creating a team:

- The email address (see page 228) of the team must be created. It identifies the sender of the emails.
  Configuration in the menu *Communication > Email > Email Addresses*
- The email accounts for the inbox (see page 201) must be configured. The emails are retrieved from these accounts.
  Configuration in the menu *Communication > Email > Inbox*
- The email filters (see page 233) for the automatic distribution of emails must be configured.
  Configuration in the menu *Communication > Email Filter.*
- The outbox (see page 222) must be configured for sending emails.
  Configuration in the menu *Communication > Email > Outbox* or in the Setup Assistant (see page 58) .

To create a new team, proceed as follows:

1. In explorer, navigate to *Ticket > Teams.* In the content area, a table listing all teams created in the system opens.
2. In table, click "New Team". A form dialog for creating the new team opens.
3. Complete form (see table below) and set validity to "valid".
4. Click "Save" to save new team. The new team has now been created.
5. In the menu *Communication > Email,* you can now configure the inbox and outbox for the team so that tickets can be received and messages can be sent.

To edit a team, proceed as follows:

1. In the explorer, navigate to *Ticket > Teams*. A table opens in the content area that lists all teams created in the system.
2. In the table, click on the team you want to change. A form dialogue opens in which you can edit the team.
3. Change the information and apply the changes with "Save".

Your changes take effect immediately. The agents may need to refresh their browser page to see the changes.

Note: Changes to a team are only effective in the future. Existing tickets remain unaffected.

To copy a team, proceed as follows:

1. In the explorer, navigate to *Ticket > Teams*. In the content area, a table opens that lists all teams created in the system.
2. Mark the team to be duplicated with a tick. Only 1 team may be selected.
3. Click on "Duplicate" in the table header. A form dialogue opens in which you can edit the copy of the team.
   Note: The form field "Name" has the content "Copy of [name of duplicated team]". You can change this name.
4. Change the details in the duplicate as required and apply the changes with "Save". The duplicate is now created.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Give the team a meaningful, unique name, e.g. the name of a department (such as Service Desk, 2nd Level Support, Purchasing, etc.)
Icon	The icon is used to graphically identify the team folder in the Ticket Dashboard and in the team overview in the Admin area. You can choose an icon from one of the provided libraries or upload a graphic file.
Parent Team	By selecting a parent team, you can generate a tree structure like the one used in an explorer.

Follow Up on Tickets	Specify how KIX should proceed when the team receives a follow-up to a closed ticket. For more information, see: Notes on email distribution (see page 207) .	
	possible:	The ticket is reopened and a new item is created. The SSP then allows the action "Follow Up Inquiry".
	reject:	The ticket is not reopened. The SSP then does **not allow** the action "Follow Up Inquiry".
	new tickets:	A new ticket is created. The SSP does **not allow** the action "Follow Up Inquiry".

Follow Up Ticket Lock	Place a tick if, in the event of a reply to the same ticket, this ticket will be locked for the last owner. The field is only displayed if the option "possible" is selected for "Follow Up on Tickets".

Field (selection)	Description	
Unlock Timeout	States the period after which the ticket will be unlocked. Tickets that an agent is processing are locked for other agents. In other words, other agents will be unable to process them. The unlock timeout specifies the period after which a ticket from this team will be unlocked, thus making it available to other agents again. 0 = no unlock timeout; the tickets will not be unlocked for other agents; they can only be unlocked manually by the agent themselves.	
Sender Address (Email)	Outbound replies from the team are sent from this email address. From the point of view of the email recipient, this is the sender, e.g. support@YourCompany.com The drop-down menu contains all addresses created under Communication > Email > Email Addresses.	
Validity	valid:	Tickets can be assigned to this team and processed.
	invalid/ invalid-temporarily:	No tickets can be assigned to the team. The team will no longer be listed in the team explorer of the Ticket Dashboard. Existing tickets will continue to be listed in the ticket overview so that they can be processed. If the ticket information is opened for editing, a red border around the field will highlight if the ticket should be assigned to another team.
Signature	This custom signature is used in emails sent from the team. The signature can contain elements such as the contact details, data privacy statement, and legal notice for emails. When creating a signature, you are able to use placeholders for variable text fragments (see section "Placeholders (see page 746) ").	

Field (selection)	Description

Assigned Roles	Define here (optionally) which user roles should have which authorisations on the team.
	This sets the base permissions (see page 655) to the team.
	
	• **Read:** Allows reading tickets in this team. It is not possible to change or edit the ticket.
	• **Write:** Allows creating and moving tickets in this team. However, the ticket cannot be viewed without Read.
	• **Read & Write:** Corresponds to full access. This permission is required to edit a ticket in any way
	
	Example:
	
	Fig.: Example for team "Service
	• Role "Team Red" - Read & Write
	  Users of the role "Team Red" may:
	  • read and edit all tickets in the "Service" team
	  • move tickets to the team "Service"
	  • create new tickets in the "Service" team
	• Role "Team Orange" - Read (Read)
	  Users of the role "Team Orange" are allowed to:
	  • read all tickets in the team "Service
	  • not edit any tickets in the team "Service"
	  • not move tickets to the "Service" team
	  • not create new tickets in the team "Service"
	• Role "Team Yellow" - Write
	  Users of the role "Team Yellow" are not allowed to
	  • not read tickets in the team "Service"
	  • edit tickets in the team "Service"
	  • move tickets to the "Service" team
	  • create tickets in the "Service" team

Field (selection)	Description
	With your selection, the selected roles automatically receive the corresponding basic permissions for the team. *Fig.: Role Team Red has all permissions on teams 9, 10, 11 and 12. It has no read permissions for team 13.* The agents, responsible persons and teams available for selection on the ticket depend on the basic permissions set here. The "Top 10" potential agents are offered for selection in the dropdown. These are the first maximum of 10 valid agents who have READ + WRITE authorisation on tickets in the target team (sorted by login name). Entering a search pattern performs a search of all valid and authorised agents (READ+WRITE) according to the search pattern.

14.3.2 Setting Up the Default Team (Default Queue)

Configuration key:	PostmasterDefaultQueue

You can set up the default team for receiving tickets, and thereby specify the team folder in which new tickets should be created by default. This is the folder in which all new tickets will be created if the dispatching type "Default" has been selected when setting up email accounts. "Service Desk" is preselected as standard for the team folder.

Fig.: Default Team Folder

To change the default team folder, proceed as follows:
1. In explorer, navigate to *System > SysConfig*.
2. Search for key "*PostmasterDefaultQueue*". Click key to open it for editing. A form dialog fo specifying a different team folder opens.
3. Under "Value", enter name of team to be used as default in future, e.g Raw. Ensure that this team has already been created.
4. Save your change.

To restore default settings, click "Reset to Default". As the final step, save your changes.

14.4   Text Modules

You can create text modules in the Admin module to handle commonly used expressions and phrases. The agents can then use them when creating tickets, when creating, answering, and forwarding articles, and in signatures (when creating new teams or editing teams). Text modules are often used for phrases forming part of polite formalities, for greetings, and for closing statements, as well as instructions for standardized procedures and solutions.

Agents can use the created text modules if they enter the following control string in the editor: "::" (double colon). A selection list displaying the available text modules opens in a context menu. The selection can be narrowed down further by optionally entering more text.
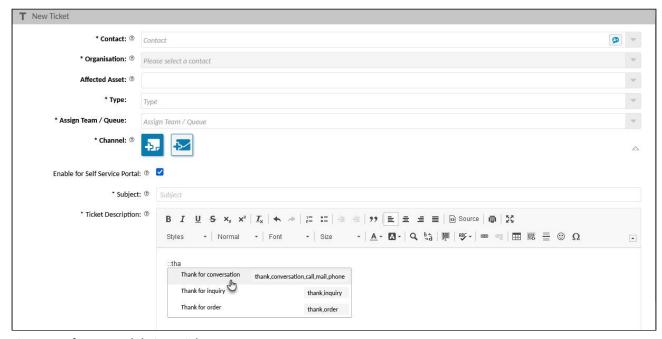
Fig.: Use of Text Module in Article Text

> ⓘ **Migration notes KIX 17 > KIX 18**
>
> Text modules from KIX 17 can be taken over. The migration script used for this removes any superfluous database structures (columns).

14.4.1   Creating or Editing a Text Module

Text modules are created in the admin module and can then be used in the agent portal when creating and editing tickets and articles. Changes to text modules are only made for the future. These changes to existing tickets will not be taken into account.

Text modules are created in the menu *Tickets > Text Modules.* The table in the content area lists all created text modules and indicates in which language they have been created. KIX currently does not support translations for text modules. Therefore, please create a separate text module for each language.

Text modules can contain variable text fragments such as the personal form of address for the customer, the current date, or the relevant ticket number. To achieve this, placeholders are inserted in the text as variables. These are then replaced with the actual content when they are used. Further information on the use of placeholders can be found in the chapter "Placeholders (see page 746) ".
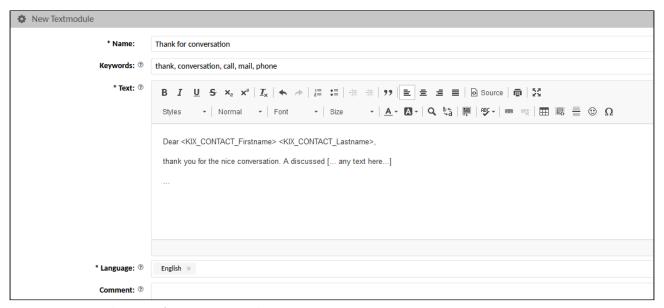
Fig.: Text Module "Thanks for conversation"

To set up a new text module, proceed as follows:

1. In explorer, navigate to *Ticket > Text Modules*. In the content area, a table listing all text modules created in the system opens. In table, click "New Text Module".
   A form dialog for saving the master data and the text for the text module opens.
2. Complete form and save text for text module in editor. You can use placeholders in your text to incorporate variable text fragments (also see section "Placeholders (see page 746) ").
3. As the last step, click "Save" to save your entries.

The text module has now been created and can be used by the agents.

To edit a text module, proceed as follows:

1. In explorer, navigate to *Ticket > Text Modules*. In the content area, a table listing all text modules created in the system opens.
2. In table, click text module to be edited. A form dialog for changing the master data and the text for the text module opens.
3. Change information as required and, as the last step, save your entries by clicking "Save".

Your changes only take effect for any future use of the text module.

To copy a text module, proceed as follows:

1. In explorer, navigate to *Ticket > Text Modules*. In the content area, a table listing all text modules created in the system opens.
2. In table, check off relevant field to select text module to be copied. The "Duplicate" button is activated.
3. Click "Duplicate" button. A form dialog for changing the master data and the text for the text module opens.
   Attention: The form field "Name" will contain the text "Copy of [name of duplicated text module]". Rename text module.
4. Change information as required and, as the last step, save your entries by clicking "Save".

he form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Give the text module a meaningful name. An agent can use this name to find the text module in the selection list.
Keywords	Optionally, enter keywords. Agents can use these keywords to find the text module if they do not know the name of the text module. Use commas to separate multiple keywords.
Text	Enter text of text module into editor. The text can be formatted using the functions provided by the editor. When in the source code view you can also use HTML tags. You are able to use placeholders to incorporate variable text fragments in the text.
Language	Select language to assign text module to a language. This means that, for example, German-speaking agents will only be provided with text modules written in German.
Validity	• **valid:** The text module can be used by the agents. • **invalid/invalid-temporarily:** The text module is (temporarily) unavailable and cannot be used. The (temporarily) text modules are retained in the places where they are already being used, but can no longer be used for new texts.

14.5   Types

The menu *Ticket > Types* enables you to create and manage ticket types. You can use these types to classify what kind of tickets you are dealing with, e.g. an incident or a service request. Agents can choose from the ticket types created here when creating a new ticket. They can also search for these types in Advanced Searches, e.g. search for all service requests within a certain period of time.
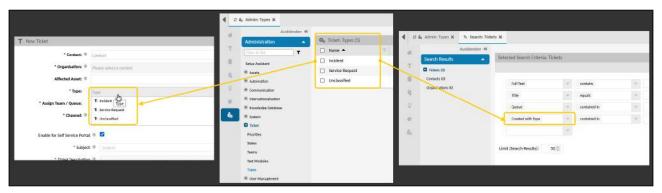
Fig.: Ticket Types When Creating New Tickets and in Advanced Searches

14.5.1   Creating or Editing a Ticket Type

KIX is supplied with a small selection of ticket types: Incident, Service Request, and Unclassified. You are able to change these and add other ticket types.

To create a new ticket type, proceed as follows:

1. In explorer, navigate to *Ticket > Types*. In the content area, a table listing all types created in the system opens.
2. In table, click "New Type". A form dialog for creating a new ticket type opens.
3. Complete form, select icon if required, and set validity to "valid".
4. Click "Save" to save new type.

The type has now been created and can be assigned to tickets.

To edit a ticket type, proceed as follows:

1. In explorer, navigate to *Ticket > Types*. In the content area, a table listing all types created in the system opens.
2. In table, click ticket type to be changed. The zoom view for the ticket type opens.
3. In opened zoom view, click "Edit". A form dialog for editing the ticket type opens.
4. Amend information as required and click "Save" to apply changes.

The changes are now effective.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Name	Give the type a meaningful, unique name so that agents can sensibly assign the ticket type.
Icon	The icon is used to graphically identify the type. It is used, for example, in the selection field "Type" when creating a new ticket and in the overview of ticket types in the Admin Dashboard.
	You can choose an icon from one of the provided libraries or upload a graphic file.
Validity	• **valid:** The type can be assigned to a ticket.
	• **invalid/invalid-temporarily:** The type cannot be assigned to a ticket. The type is retained for existing tickets so that they remain identified as such. If the ticket information is opened for editing, a red order around the field will highlight if another ticket type should be selected.

15 User Management

The menu *User Management* is where you create all KIX users and assign them system rights.

Users

In order for a user to log in to KIX and work with it, he must be created as a user in the system. Users are all users who have access to the Agent Portal and/or Self Service Portal. For example, all agents who are allowed to read, create and edit tickets, administrators or selected contacts with access rights.

As an alternative to user management, you can use the user data and rights of your LDAP/AD server (domain controller) to allow agents and contacts to log on to KIX. The advantage here is the central user management for both your infrastructure and KIX. The Microsoft Active Directory (AD) as well as all common LDAP directory services are supported (also see "Authentication/Authorisation and Connection to Active Directory (see page 671) ").

Roles

Roles define what a user may and may not do. Each user must be assigned at least one role. For this reason, each new user is automatically assigned the role "Agent User". A user must have at least the role "Agent User" to be able to log in to the system.

A user can have several roles. A user's permissions are determined from the totality of the roles they have been assigned. It is possible to not only assign various users to a single role, but also to assign a single user different roles.

KIX is supplied with pre-configured roles that you can assign straight away. The zoom view for a role details the permissions that make up that role (see the "Permissions" lane).

> ⓘ **Please note**
>
> In the interests of providing a contemporary rights concept in KIX, we have opted against the additional use of groups.

15.1 Roles/Permissions

A user must have roles assigned to them in order for them to be able to log in to KIX and work with the system. The roles define the user's rights in the system. As such, you can specify that users are only allowed to access certain areas of KIX in line with their remits, and are only allowed to carry out certain functions within these areas. In this way, service technicians, for example, can be granted access to areas and functions that remain off limits to other employees (e.g. call center employees).

A user can have several roles. To access the agent portal, at least the "Agent User" role is required to log in. Therefore, every user with the usage context "Agent Portal" is automatically assigned the basic role "Agent User". Accordingly, a user with the usage context "Self Service Portal" is automatically assigned the basic role "Customer". If the user's access to the agent portal or self-service portal is withdrawn, the role assignment is also removed. Therefore, the basic roles "Agent User" and "Customer" should not be assigned manually to the user accounts.

Roles are managed and configured in the menu *User Management > Roles/Permissions*. You can assign a role one or more users. Alternatively, you can assign a user one or more roles (menu *User Management > Users)*.

A user's permissions are determined by all of the roles they have been assigned. By combining several roles, you can therefore tailor the permissions for each user. The table in the Admin Dashboard lists all roles created in the system.

KIX is supplied with predefined roles. When used in combination with one another, these roles cover the main application scenarios. You can assign these roles to users straight away. If required, you can change the permissions in the roles or define your own roles. However, take the **greatest possible care** when doing so, as you could otherwise inadvertently **lock yourself out** of the system! We would be glad to assist you should you require any support with additional roles and permissions.

The zoom view for a role shows you which rights have been defined in a role and which users have been assigned to it. Moreover, the zoom view for a user shows you which roles they have been assigned.

⚠ Attention!

The initial user delivered with KIX works in the system as a **root user.** He has full access to all system areas down to the lowest system level!
In order not to endanger the security of the system, please note:
- Only work with the initial user in **exceptional cases**!
- Create an admin user (role "Super User") and work with it. The role "Super User" also has all system rights, but at user level. The Setup Assistant will support you in this. Then log in again with this user.
- Do **not delete** the initial user! Basic KIX functions are bound to this user. By doing so, you would also delete these functions and "undermine" them.

> ⓘ **Notes**
>
> If you have made changes to the role assignments, the users concerned must update their system by logging in again so that the changed authorisations take effect. The users will receive a message informing them of this.
>
> The tab "New ticket" will not be displayed in the New dialogue if there are no CREATE permissions on /tickets/*/articles.

15.1.1 Zoom View for a Role

KIX comes with pre-configured roles that you can assign directly to users. In the detailed view, you can see which permissions make up a role and which users it is assigned to, you can view the details of a role.

To open the zoom view, navigate to *User Management > Roles/Permissions.* In the content area, a table listing all the roles saved in the system opens. In table, click a role to open associated zoom view. If you want to make changes to the role or the user assignment, click "Edit" in title bar of zoom view.
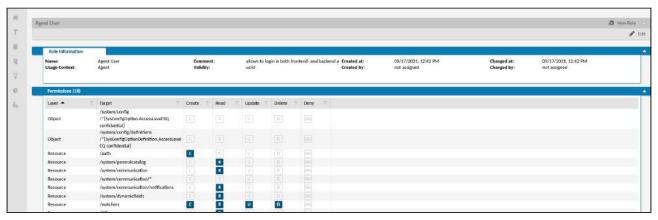
Fig.: Zoom View of a Role

In addition to the "Role Information" lane containing the role's general reference data, the zoom view also has other lanes:

Lane	Description
Permissions	Breakdown of permissions for a role. Shows which rights (Create, Read, Update, Delete, Deny) the role owner has and for which areas. These rights are shown in blue.
Asigned Agents	List of all users who have been assigned to the role.

15.1.2 Configuring and Assigning a Role

KIX is supplied with a range of pre-configured roles that you can assign to users straight away. These roles are sufficient to cover most requirements. Nevertheless, you can, if required, change the permissions that make up these roles and define additional roles yourself.

> ⚠ **Attention!**
>
> If you make changes to the roles and permissions, there is a risk of **locking yourself out** of the system! Please do not hesitate to contact us if you require additional roles. We would be glad to create them for you.

To assign users to a role or remove assigned users, proceed as follows:

1. In explorer, navigate to *User Management > Roles/Permissions. I*n the content area, a table listing all the roles saved in the system opens.
2. In table, click role with user assignment to be changed. The zoom view for the role opens.
3. Click "Edit" in title bar of opened zoom view. A form dialog showing the permissions and user assignment opens.
4. In selection field "Agents", select users to whom you wish to assign role. You can select more than one option.
5. To remove user from selection, click small cross in user's label.
6. As the last step, click "Save".

The permission will be transferred to/withdrawn from the selected user immediately. The users will need to log in to the system again in order for the changes to take effect.

To create a new role, proceed as follows:

1. In explorer, navigate to *User Management > Roles/Permissions.* In the content area, a table listing all the roles saved in the system opens.
2. In table, click "New Role". A form dialog for creating a role opens.
3. Save necessary information under "Role Information" and set role to "valid".
4. Under "Permissions", create rights concept for role. Take great care when doing so!
    1. In the "Type" selection field, select the object type that the role can access. (see table below)
    2. Enter the path to the selected object type in the adjacent text field "Target", e.g. "/contacts".
    3. Place a tick in the checkboxes for the rights you wish to assign to the selected resource.
    4. If necessary, make a note of a comment.
    5. If necessary, select further resources as described under a) to d).
5. Assign one or more users to the new role under "Agent Assignment".
6. Save the new role with "Save".

The role is immediately active for the selected users. However, the users will need to log in again.

To edit a role, proceed as follows:

1. In the explorer, navigate to *User Administration > Roles/Permissions.* A table opens in the content area listing all the roles stored in the system.
2. In the table, click on the role you want to edit. The detailed view of the role opens. Click on "Edit" in the title bar of the opened detail view. A form dialogue opens in which you can see the permissions of the role.
3. Change the current permissions as required under "Permissions".
4. If necessary, select additional users under "Agent Assignment". Multiple selection is possible. To remove a user from the selection, you can click on the small cross in the user's label.
5. Finally, click on "Save".

The changes take effect immediately. The users will need to refresh their system in order for the changes to take effect.

> ✅ **Tip**
>
> Roles can also be created and updated using the script kix.18.ManageRoles.pl[22] provided on GitHub[23] (see section "Manage Roles and Permissions - kix18.ManageRoles.pl" there). In addition, the RoleData_Sample.csv[24] provided contains the default roles and permissions provided by KIX.
> To use the scripts, you need knowledge of how to use command lines. The scripts do not work with Windows operating systems.
> **Note**: The scripts provided on GitHub do not claim to be complete. However, we make every effort to keep them up to date.

The form dialog contains the following input fields, among others:

Field (selection)	Description	
Name	Use a meaningful and logical name for the role, as this will make it easier to manage the roles.	
Usage Context	Select for which portal the rights of the role apply (Agent Portal and/or Self Service Portal).	
Validity	valid:	The role can be assigned to users. Users who have this role can use the functions and areas that have been defined with this role.

---

22 https://github.com/kix-service-software/kix18sync/blob/master/bin/kix18.ManageRoles.pl
23 https://github.com/kix-service-software/kix18sync
24 https://github.com/kix-service-software/kix18sync/tree/master/sample

Field (selection)	Description	
	invalid/ invalid-temporarily:	The role is (temporarily) not being used. The role cannot be assigned to any users. Users who have the role will have this authorization withdrawn; they will then be unable to exercise the associated permissions.
Permissions	Create permissions concept for this role. Specify which areas the role owners are permitted to access and which system rights they have to do this.	
	Allow access to admin module	Additional switch to show or hide the admin area for selected roles. If this option is enabled, users with this role can see the Admin module. If this option is not enabled, users with this role cannot see the Admin module even if they have permissions on system resources. Can be used, for example, if a "Customer Manager" with CRU rights on "/system/objecticons" is allowed to set the avatar for a customer, but should not have access to the admin area for the object icons. **Still requires the appropriate permissions to be active in the admin area.** Default configuration: • Role "Customer Manager" does not have access to the admin area. • Roles "Superuser", "System Admin", "Textmodule Admin", "FAQ Admin" have access to admin area • SystemUser (UserID1) always has access to admin area The assignment can alternatively be done in the SysConfig key "*agent-portal-config*". The roles that have access to the admin area are stored in this key.

Field (selection)	Description
	Selection field "Type" Select the object type **Resource:** All objects of a resource (e.g. tickets, contacts, FAQ) as well as all subordinate resources (articles for tickets), unless other permissions have been assigned for them. Explicitly identified objects are a special case • z. e.g. read access to all contacts (without further restrictions): `Ressource | /contacts | -R---` • z. E.g. read access to queue with ID 4 (not to the tickets in it): `Ressource | /system/ticket/queues/4` **Base:** Base authorisation for business objects (e.g. tickets in teams). If role assignments are selected for a team, the base permissions are automatically set here. (see also: Base permissions) • z. e.g. read and write permissions on tickets in all teams: `Base::Ticket | * | CRUD-` • z. e.g. read permission on all tickets in team 3: `Base::Ticket | 3 | -R---` **Object**: Specific objects of a resource (e.g. article for any ticket) • z. e.g. read access to the articles of all tickets: `Object | /tickets/*/articles | -R---` • z. E.g. read access to tickets in queue with ID 4,5 or 6: `Object | /tickets/*{Ticket.QueueID IN [4,5,6]} | -R---` **Property:** Properties of a concrete object of a resource • z. E.g. restriction of read access to the attributes "First name", "Last name", "Email": `PropertyValue | /contacts/*{Contact.[Firstname,Lastname,Email]} | -R---`

Field (selection)	Description
	Please Note: To access a property, access to a concrete object is required. To access a concrete object, access to the resource is required. To access a resource, access to the parent resource or resource level is required.
	Text field "Target" — The path to the resource or the attribute/value (e.g. "/contacts").
	Checkboxes — Check off relevant checkbox(es) to set corresponding permission (also see table "Permission concept KIX 18 (see page 645) "). • C: CREATE authorisation • R: READ authorisation • U: UPDATE authorisation • D: DELETE authorisation • Deny: DENY authorisation
	Comment — Optional notes
Agent Assignments	In selection field "Agents", select the users to which this role Assignment should be assigned. You can select more than one option. Users who have already been assigned this role will already be selected. Click small cross in user's label to remove them from selection.

15.1.3 Overview of preconfigured roles in KIX

Role	Permission
Agent User	Basic role. This role is required by **all users** to log in to the system (except "Super User" role)! • Allows logging in to both the frontend and backend applications • does not, however, grant any other permissions. • News messages can be read. The role is automatically assigned when the agent portal is selected under "Access" in the user or contact configuration. If the user's access to the agent portal is withdrawn, the role assignment is also removed. To avoid inconsistent behaviour, this role should **not be assigned** in explicit mappings such as in LDAP/AD configurations (GroupDNBasedRoleSync). Allows the display and data entry of dynamic fields on tickets, FAQs, organisations or contacts when using KIX Connect.
Anonymus Self Service Portal User	This role is stored on the Self Service Portal. It is not a user role. The Self Service Portal logs on to KIX with this role. It is required for the data exchange between KIX and the Self Service Portal. Therefore, the initial configuration of this role must not be changed.
Asset Maintainer	Like Asset Reader, but additionally enables the user to: • create new or update existing entries for configuration elements • create, update and delete links
Asset Reader	Enables the user to read configuration element information in any asset class and read links.

Role	Permission
Base::Tickets	Base permission for tickets in queues. The target is always a QueueID. Base permissions act directly after resource permissions. First, a permission to a resource (e.g. "/tickets") must exist, then the basic permissions are checked. If access is permitted, further object permissions are checked and finally property permissions are applied. Base permissions are granted when teams are created/edited, but can also be granted directly at a role definition. For detailed information, see Base Permissions (see page 655) .
Customer	Basic role for logging into the Self Service Portal. Enables the customer in the Self Service Portal to • Read: tickets, FAQ articles, Asset data, News messages, Contacts • Create FAQ articles • Change his user data • Set item flags The Role is assigned automatically when the Self Service Portal is selected under "Access" in the user or contact configuration. If the user's access to the Self Service Portal is withdrawn, the role assignment is also removed. To avoid inconsistent behaviour, this role should **not be assigned** in explicit mappings such as in LDAP/AD configurations (GroupDNBasedRoleSync). Functionally, the role is similar to the "Agent User" role for the Agent Portal. Allows the display and data entry of dynamic fields on tickets, FAQs, organisations or contacts when using KIX Connect.
Customer Manager	Like Customer Reader, but additionally enables the user to: • create or update an existing contact or organization entry • create, update and delete links
Customer Reader	Enables the user to read information on organizations and contacts and read links.

Role	Permission
FAQ Admin	Users with this role will see a restricted menu tree for FAQ category administration in the Admin module explorer. Allows to create, edit and delete FAQ categories.
FAQ Editor	Like FAQ Reader, but additionally enables the user to • create new or edit existing FAQ articles • create, update and delete links
FAQ Reader	Enables the user to read an FAQ article in any FAQ category and read links.
News Manager	Enables the creation and editing of news messages for display in the Self Service Portal (KIX Pro).
Report Manager	Extends user rights to create reports. Can create new report configurations and customize existing report configurations.
Report User	Extend user rights to read reports. Can use existing report configurations to create concrete reports. Can retrieve reports manually from GUI and save them as zip file.
Service Contract Manager	Enables the reading, creation, editing and deletion of service agreements. Also requires at least the "Asset Reader" role.
Super User	Full Access to all resources
System Admin	Requires an Agent User role in order to log in to the system. Enables system configurations to be changed and edited in the Admin module.

Role	Permission
Textmodule Admin	Users with this role see a restricted menu tree for the administration of text modules in the explorer of the Admin module. Allows to create, edit and delete text modules.
Ticket Agent Base Permission	Contains the basic permissions to edit tickets without specific queue assignments. It is used in conjunction with team/queue-specific roles to allow collaboration/reading in specific queues. This role is the basis for team-specific collaboration.
Ticket Reader	Allows you to read any ticket in any queue and read links
Ticket Agent	Like Ticket Reader, but additionally enables the user to • create a new or edit an existing ticket • create, update and delete links • reading individual entries of service agreements • reading templates and actions.
Ticket Agent (w/o Team)	Like Ticket Agent, but • no access to concrete teams (queues) • no access to tickets in concrete teams (queues). Allows basic ticket access, but requires team-specific roles, e.g. "Ticket Agent (Servicedesk)". Has read access to individual entries of service agreements. The role contains the basic permissions of a ticket agent, but prevents access to tickets in certain teams. These can be created based on the role "Ticket Agent (Service Desk)". ⚠ As of v28, this role still contains the basic permissions for editing tickets without concrete queue assignments, but it cannot be used in connection with basic permissions. It will therefore be dropped in the future.

Role	Permission
Ticket Agent (Servicedesk)	Allows ticket editing in the following teams. Usage requires assignment of the "Ticket Agent (w/o Team)" role, i.e. a user must hold both roles. • Service Desk • Service Desk Monitoring
Webform Ticket Creator	Enables the creation of new tickets via the web form (see page 253) . The role was created specifically for the needs of the web form. If you use the web form, you must create a user with exactly this role and store it in the web form. Web forms are created and configured in the menu "*Communication > Web form*".

15.1.4 Permission concept KIX 18

15.1.4.1 Permission levels

The KIX18 authorisation concept distinguishes between different access levels that are mutually dependent. This means that the user must first be authorised to access a resource (e.g. "/tickets"). Then the base permissions are checked. If this access is permitted, further object permissions are checked and finally property permissions are applied.

The KIX18 permission concept distinguishes between different access levels that are interdependent:

- **Resource level:** controls access to object collections
  e.g. tickets, FAQ, teams, etc.
  Special case: a collection with one element, e.g. "/tickets/123" defines a collection consisting of exactly one object.

- **Base level:** Controls access to a business object
  e.g. on tickets in teams/queues
  (see also Base Permissions (see page 655))

- **Object level:** Controls access to concrete objects, which are described by properties of the objects.
  e.g. tickets in certain teams or tickets to certain organisations

- **Property** level controls access to object properties
  e.g. the status of a ticket (Ticket.State), the priority of a ticket (Ticket.Priority) or the time booked in the ticket (Ticket.AccountedTime).

> ⓘ **Note**
>
> There is a requirement relationship between the levels. This means: Without access to the resource, there is no access to the object. Without access to the object, there is no access to the property.

15.1.4.2 Permission types

The following permission types are used in KIX 18:

Right	Notation	Description	Note
CREATE	C	Write permission. The user is permitted to create a **new** element (e.g. ticket, team, asset, customer, contact, or FAQ).	Corresponds to an HTTP POST
READ	R	Read-only permission. The user is only permitted to view **existing** elements; they cannot edit them.	Corresponds to an HTTP GET
UPDATE	U	Editing permission. The user is permitted to edit **existing** elements, e.g. move a ticket into a team.	Corresponds to an HTTP PATCH
DELETE	D	Delete permission. The user is permitted to delete **existing** elements provided that the system allows this (e.g. a note for a ticket).	Corresponds to an HTTP DELETE
DENY	(without)	Deny permission. **All access** to the element will be **strictly** forbidden. The DENY rule has the **highest priority**. As such, it also **denies** authorized access permissions that result from other role assignments.	The DENY rule has the **highest priority** and cannot be overwritten. It **revokes** all permissions granted elsewhere and is therefore relevant for the combination of roles.

For easier readability, the notation "CRUDX" is used:

Notation	Description
-R---	for reading only
CR---	for creating and reading
CRU--	for creating, reading and changing
---D-	for delete
-----	for no access (which can be extended again by other roles)
----X	for absolutely no access (which cannot be cancelled by other roles)

15.1.4.3   File analogy

The authorisations to the individual levels are comparable to a collection of files
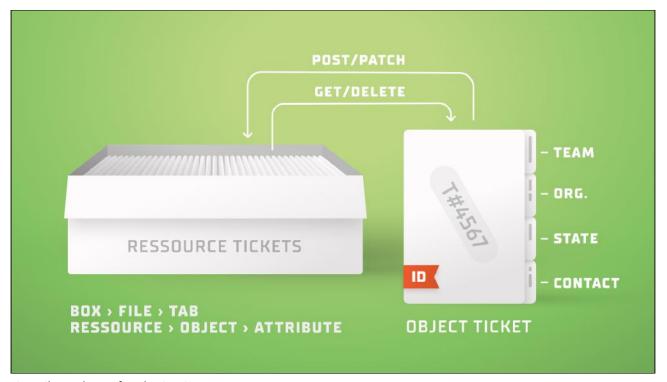
Fig.: File analogy of authorisations

Levels	Permissions				
	Create	**Read**	**Update**	**Delete**	**Deny**
1. Resources	Create in resource *User can create new files.*	Read in resource *User can read files.*	Change in resource *User can change files.*	Delete in resource *User can delete files.*	Withdraw all access to resource *User must do absolutely nothing with files.*
2. Base	Create tickets in a specific team. *User can create and move new files*	Read tickets in a specific team. *User can read files.*	Create and move tickets in a specific team. *User can create and move new files.*	Delete tickets in a specific team. *User can delete files.*	Withdraw all access to tickets in the team. User must do absolutely nothing with files.
3. Objects	Object may be created as transmitted *User may create THIS record.*	Read existing objects User is allowed to read THIS file.	Objects may be changed as transmitted *User may change THIS record.*	Delete existing objects *User may delete THIS file.*	Block all access to an object *User must do absolutely nothing with THIS file.*
4. Properties	Allows setting of certain properties at creation *User is allowed to create a file with THIS chapter.*	Allows reading of certain properties *User is allowed to read THIS property of the record.*	Allows certain properties to be changed *User is allowed to set THIS property on the record.*	n. a.	Block all access to a property *User is not allowed to do absolutely anything with this property.*

15.1.4.4 Sub-resources

The following applies to sub-resources:

1. The permissions on sub-resources correspond to the permissions granted on the superordinate resources, unless otherwise stated.
2. Permissions on sub-resources may not be more extensive than permissions on superordinate resources.
   This means, for example: If there is no update right on the resource "/system", no update right can be granted on the resource "/system/automation".
3. Permissions on identifying sub-resources depend both on base permissions on the resource and on the concrete permissions on the parent, identified object.

Identifying sub-resources (ticket items)

Permissions on identifying sub-resources represent a small speciality. In order to be able to describe them, the asterisk placeholder is used as in "/ticket/*/articles". The asterisk has the same effect as in the file system, for example, and means "all elements directly under /tickets/".

Example:

Creating an item at ticket no. 123 therefore requires:

- Create and update right to resource "/tickets".
  AND
- Update right on object "/tickets/123".
  AND
- Create right on resource "/tickets/*/articles" (in the example with * == 123)

15.1.4.5 Base permissions

Base permissions are stored on "surrounding" objects for business objects. Business objects are tickets, assets, FAQ articles, contacts and organisations. This results in the following "pairings":

- Tickets and teams/queues
- Assets and asset classes
- FAQ articles and FAQ categories
- Contacts and organisations
- Tickets and organisations

> ⚠ **Note**
>
> Currently, base permissions are only possible for tickets in teams/queues. In the medium term, this approach will be extended to other pairings.

The base permissions for tickets in teams (queues) are granted directly when creating/editing a team. In the process, roles are selected and their respective permissions are stored in relation to the team.

- **Read:** Allows reading tickets in this team. It is not possible to change or edit the ticket.
- **Write:** Allows the creation and moving of tickets in this team. However, the ticket cannot be viewed without Read.
- **Read & Write:** Corresponds to full access. This permission is required to edit a ticket in any way.

> ⓘ For the most common use cases, such as restricting access to tickets based on the team assignment of the ticket, only the base permissions are required. Furthermore, the base permissions are the basis for determining the teams/queues that can be selected for tickets to be created or edited and, based on these, also the agents/responsible persons.
> Further information can be found under: Base Permissions (see page 655)

15.1.4.6   Object permissions

Object permissions allow a further check of the object for certain properties for granting/withdrawing permissions. Use cases include the exclusion of tickets with sensitive content (e.g. type "Incident Security") or specific customer organisations.

The HTTP method defines which permission is granted/withdrawn.

With CREATE/UPDATE, the check is based on the transmitted properties to be (re)set (in the HTTP request POST/PATCH).

With READ/DELETE, the properties existing in the backend or in the database are used.

Different properties can be checked. These checks can be linked with the logical AND operator "&&". Alternative options (OR links) can be mapped by several authorisations. Furthermore, absolute values or properties of the executing user can be checked using the placeholder " `$CurrentUser` ".

Optionally, it is possible to restrict property permissions. Properties for specific objects can be defined using IF.

Permissible comparison operators are: `LT, LTE, GT, GTE, CONTAINS, LIKE, IN, STARTSWITH, ENDSWITH, EQ, NE.`

The relational operators can be negated by means of `"!".`

Examples for checking object properties

```
# ALLOW full access to ticket
# WHERE title contains "Security"
#   AND priority-ID is 3
```

```
Object | /tickets/*{Ticket.Title CONTAINS "Security" && Ticket.PriorityID LT 3} |
CRUD-

# ALLOW to read tickets
# WHERE SLA is not 5
#   AND team/ueue is not 1,2 or 3
Object | /tickets/*{Ticket.SLAID NE 5 && Ticket.QueueID !IN [1,2,3]} | -R----

# ALLOW to read tickets
# WHERE title LIKE "*something*"
Object | /tickets/*{Ticket.Title LIKE "*something*"} | -R----

# DO NOT ALLOW ACCESS to tickets
# WHERE contactID differs from current users contact id AND
#   AND tickets organization is not current users primary organisation
Object | /tickets/*{Ticket.ContactID NE $CurrentUser.Contact.ID &&
Ticket.OrganisationID NE $CurrentUser.Contact.PrimaryOrganisationID} | ------

# DO NOT ALLOW ACCESS to articles
# WHERE customer visible flag is not set
Object | /tickets/*/articles/*{Article.CustomerVisible NE1} | -----
```

15.1.4.7 Property permissions

Property permissions only affect existing objects. A CREATE permission cannot be set or revoked with it.

The optional specification of property permissions allows the restriction of access to object attributes. To be able to specify these, knowledge of the existing attributes is required.

With regard to dynamic fields, these can vary in different environments. These can vary in different environments.

A concrete application is in the role "Customer". It restricts the ticket properties that can be displayed in the Self Service Portal by defining a white list.

Example property permissions

```
# LIMIT shown ticket attributes to the mentioned (line break for improved
readability)
Property | /tickets/*{Ticket.
[TicketNumber,Age,Articles,Changed,ContactID,Created,CreateTimeUnix,DynamicFields,Org
anisationID,PriorityID,QueueID,StateID,TypeID]} | -R---
```

15.1.4.8 Roles

Roles are defined by containing a set n of permissions in addition to general header data such as name, validity, comment, etc. (1: n relationship → 1 role : n permissions).

Several roles can be assigned to a user. The resulting permissions are the union of all individual role permissions. A DENY always has the highest priority (see authorisation types).

Examples:

```
Role1  | resource | /resource/xyz/abc | -R---
+ Role2 | resource | /resource/xyz/abc | -----
+ Role3 | resource | /resource/xyz/abc | C----
= Result | resource | /resource/xyz/abc | CR---
```

```
Role1  | resource | /resource/xyz/abc | -R---
+ Role2 | resource | /resource/xyz/abc | CRUD-
+ Role3 | resource | /resource/xyz/abc | ----X
= Result | resource | /resource/xyz/abc | ----X
```

Example - Customer "Secret Company"

Scenario:

> Not all agents are allowed to view the data and tickets of the customer "Secret-Company" or not all agents work for this customer.

Solution outline:

> The predefined role "Ticket Agent" could be used, but would have to be restricted. However, a role is also needed for agents who are allowed to work for the customer "Secret-Company". The role "Ticket Agent" is therefore copied to the new role "Ticket Agent without Secret-Company".
> In the following, it is assumed that the organisation "Secret-Company" has the ID 2. This defines the role "Ticket Agent without Secret-Company" as follows (object/template action IDs may vary). The supplementary permissions are provided with a comment.

Rollendefinition "Ticket Agent ohne Secret-Company"

```
Resource | /contacts                                   | -R---
Object   | /contacts/*{Contact.PrimaryOrganisationID NE 2}   | -R--- # specific
(limit access to contacts not belonging to Org wit ID 2)
Resource | /links                                      | CRUD
Resource | /organisations                             | -R---
Object   | /organisations/*{Organisation.Number NE "SECRET"} | -R--- # specific
(limit access to org. with other Number than "SECRET")
Resource | /system/automation                         | -RU--
Resource | /system/automation/*                       | -----
Resource | /system/automation/macros                  | --U--
Resource | /system/automation/macros/*                | -----
Resource | /system/communication                      | -R---
```

```
Resource | /system/communication/*                      | -----
Resource | /system/communication/channels               | -R---
Resource | /system/communication/sendertypes            | -R---
Resource | /system/communication/systemaddresses        | -R---
Resource | /system/objectactions                        | -R---
Resource | /system/objectactions/*                       | -----
Resource | /system/objectactions/2                       | -R---
Resource | /system/objectactions/3                       | -R---
Resource | /system/objectactions/4                       | -R---
Resource | /system/objectactions/6                       | -R---
Resource | /system/slas                                  | -R---
Resource | /system/templates                             | -R---
Resource | /system/templates/*                           | -----
Resource | /system/templates/4                           | -R---
Resource | /system/templates/5                           | -R---
Resource | /system/templates/7                           | -R---
Resource | /system/textmodules                           | -R---
Resource | /system/ticket                                | -R---
Object   | /system/ticket/*{Ticket.OrganisationID EQ 2}  | ----- # specific
(prohibit access to tickets of Org 2)
Resource | /system/ticket/locks                          | -R---
Resource | /system/ticket/priorities                     | -R---
Resource | /system/ticket/states                         | -R---
Resource | /system/ticket/types                          | -R---
```

15.1.4.9   Effect of permissions in the user interface (GUI)

Resource permissions are required to gain general access or access to specific areas of the GUI. For example, at least one READ on the resource " `/tickets` " is required to access the ticket module. This also applies to access to all other objects such as FAQ (`/faq`), assets (`/cmdb`), etc.

The affiliation of users to roles is a prerequisite for using the configurable ticket and item actions as well as the ticket templates.

The permissions and displayed contents of a customer user in the SSP are defined by the default role "Customer".

Access to asset management objects continues to be defined in the backend through the configuration of the assignment mapping and supplements the authorisation system (SysConfig key "*AssignedConfigItemsMapping*").

> ✅ **Tip**
>
> For the maintenance of roles, https://github.com/kix-service-software/kix18sync and Libre Office Calc, MS Excel can also be used (CSV, UTF-8 encoded, semicolon-separated).
> MacOS users must note the special end-of-line identifier that is used when CSVs are created or modified under MacOS.

15.1.5 Base Permissions

Content on this page:

15.1.5.1 What are base permissions?

Base permissions are stored on "surrounding" objects for business objects. Business objects are tickets, assets, FAQ articles, contacts and organisations. This results in the following "pairings":

- Tickets and teams/queues
- Assets and asset classes
- FAQ articles and FAQ categories
- Contacts and organisations
- Tickets and organisations

> ⚠️ **Please note**
>
> Currently, base permissions are only possible for tickets in teams/queues. In the medium term, this approach will be extended to other pairings.

The base permissions for tickets in teams (queues) are granted directly when creating/editing a team. In the process, roles are selected and their respective permissions are stored in relation to the team.

- **Read**: allows reading tickets in this team. It is not possible to change or edit the ticket.
- **Write**: allows the creation and moving of tickets in this team. However, the ticket cannot be viewed without Read.
- **Read & Write:** corresponds to full access. This permission is required to edit a ticket in any way.
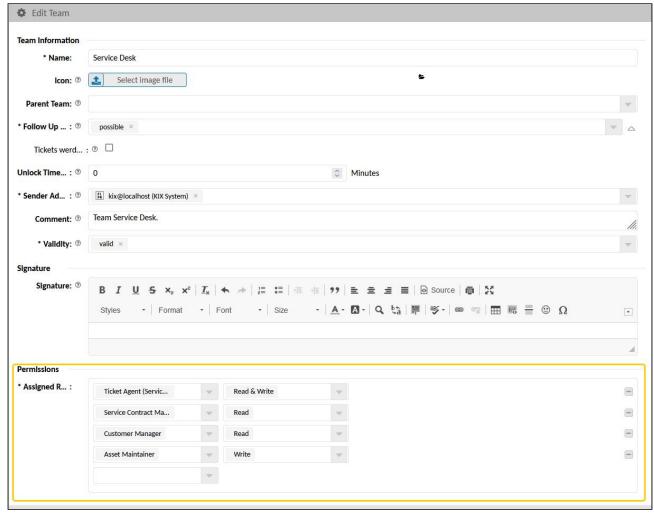
Fig.: Maintenance of base permissions in the Service Desk team

15.1.5.2   How does this fit into the KIX authorization concept?

Base permissons basically work like object authorisations: They control whether access to a business object is allowed.

They act directly after resource authorisations. This means that there must first be authorisation to access a resource (e.g. "/tickets") at all. Then the base permissions are checked. If this access is permitted, further object permissions are checked and finally property permissions are applied.
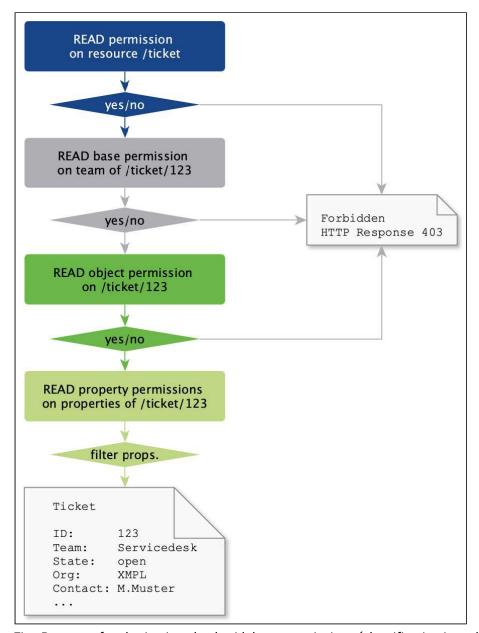
Fig.: Process of authorisation check with base permissions (classification in authorisation concept)

15.1.5.3 What is the effect of base permissions?

In order to create a ticket in a team X, the executing user needs at least Write permissions   on the selected team. Conversely, this means that the user can only select the teams in which he has the required permission. The selection of teams in ticket creation and editing is restricted.

In order to edit a ticket in a team X, a user needs Read & Write permission on the team in which the ticket is located. Conversely, this means that the user can only set agents as agents or responsible persons who have this authorisation in the team. The selection of agents and responsible persons in ticket creation and editing is restricted. This also affects the collective action on tickets.

When searching for tickets and in the team view, only those teams are available to which the user has (at least) read permission. Also, only those tickets can be viewed whose teams have (at least) Read permission.

The base permissions apply to every user - regardless of the context. This means that they are also effective in the Self Service Portal.

In the admin module and when configuring jobs, macro actions, notification rules or other automatic mechanisms, these restrictions do **not** apply to team selections.

15.1.5.4   What changes for existing KIX18 installations (pre-v28)?

Before object permissions are granted, the possibilities of the base permissions must be exhausted. **The previous approach to permissions resigns for special cases** based on criteria other than team permissions.

In order for existing installations to see a performance gain, a change from object permissions to base permissions is necessary. The previous permissions did not allow correct selection of agent, responsible person or team. Authorisation scenarios based on teams/queues should therefore be revised. However, this is not absolutely necessary. The previous authorisations continue to apply, but represent a further burden on performance.

The authorisation roles initially delivered with KIX will continue to work as described:

- Role "Customer" continues to allow full access of an SSP user according to the possibilities and permissions in the Self Service Portal.
- Role "Ticket Reader" continues to allow read access to all tickets, regardless of which team/queue they are in.
- Role "Ticket Agent" still allows full access to all tickets, regardless of which team/queue they belong to.
- Role "Ticket Agent Base Permission" contains the base permissions for editing tickets without concrete queue assignments. It is used in conjunction with team/queue-specific roles to allow collaboration/reading in specific queues. This role is the basis for team-specific collaboration.
- Role "Ticket Agent (w/o teams)" still contains the base permissions for editing tickets without specific queue assignments, but cannot be used in connection with base permissions. It will therefore be dropped in the future.

> ⚠ **Important note for update v27 > v28**
>
> After installing/updating to v28, own roles based on the default roles "Customer", "Ticket Agent (w/o teams)" etc. must be provided with base permissions on all teams/queues in order to maintain the visibility of tickets. The same applies if these roles have been renamed. If the work was done without these roles and no further base permissions are stored on the teams/queues, the existing accesses will be retained.

15.1.5.5 How can I design special roles for working in teams?

To assign team-specific permissions, we recommend the following procedure:

1. Create the specific roles for the base permissions.
   These roles do not need to be granted any permissions manually. Simply creating them is sufficient.
2. Create or edit teams (queues).
   Define which role has which authorisation within this team.
   You can specify the roles created in step 1 as well as all other roles available in the system.
   The selected roles automatically receive the set base permissions.
3. Assign at least the roles "Agent User", "Ticket Agent Base Permissions" and the specific roles from step 1 to the users concerned.

15.1.5.6 Technical details

The base permissions for tickets granted in the teams/queues are automatically stored in the corresponding role definitions. In addition to "Resource", "Object" and "Property", there is also "Base::Ticket" as a permission layer. The target here is a single queue ID. If base permissions are to be set for all teams (queues), an asterisk (*) can be entered as a wildcard instead of all queue IDs.

Permissions (70)						
Layer ▲	Target	Create	Read	Update	Delete	Deny
Base::Ticket	1	C	R	U	D	DN
Base::Ticket	*	C	R	U	D	DN
Object	/reporting/reports/*{Report.DefinitionID !IN [3,4,5,6,7,8]}	C	R	U	D	DN
Property	/system/automation/macros/15{Macro.[Exec]}	C	R	U	D	DN
Resource	/system/templates	C	R	U	D	DN
Resource	/system/templates/*	C	R	U	D	DN
Resource	/system/objectactions	C	R	U	D	DN
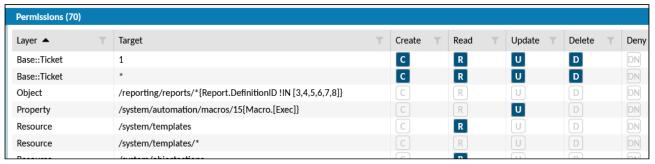
Fig.: Permission layer and permissions

The base permissions are mapped as follows:

Base permission	Mapping
Read	-R--- (Read)
Write	C-U-- (Create, Update)
Read & Write	CRUD- (Create, Read, Update, Delete)

15.1.6 Ticket creation without update permissions

You can configure that an agent who only has read permissions can create a ticket or asset without being able to make further updates to it after it has been created.

Proceed as follows:

1. The contact, the organisation and, if applicable, an affected asset must be set on the ticket. The user in question therefore needs at least read permissions on these objects.
   Therefore, navigate to the menu "*User Management > Users*" in the Admin module and assign the following roles to the respective user:
   1. Ticket Reader
   2. CustomerReader
   3. AssetReader
2. To create tickets, the user needs CREATE permissions.
   Therefore, navigate to *User Management > Roles* in the Admin module and open the role "TicketReader" for editing. Set the CREATE permission to:
   1. Resource /tickets
   2. Base :: Ticket *
3. If the user should also be able to create new assets, he needs CREATE permissions for this as well.
   Therefore, navigate to *User Management > Roles* in the Admin module and open the role "AssetReader" for editing. Set the CREATE permission to:
   1. Resource /cmdb/configitems

Save each of these changes. The user can create new tickets or assets after refreshing his browser window.

15.2 Users

You can create and manage system users. On the one hand, these can be your own company's <u>agents</u>, such as service technicians, support staff, administrators, etc. On the other hand, they can be customer <u>contacts</u> (contact persons of other companies) who have access to the agent or self-service portal.

<u>Agents</u> are created in the Admin module, in the *User Management > Users menu*. By default, they are assigned to their own organization (My_ORGA/My Organisation). By selecting another organization, they can be assigned to a customer. Each user created in the system is automatically also a contact and is listed in the overview of contacts in the Agent Portal.

<u>Contacts</u> are created in the customer module of the agent portal or via the green plus-button. Initially, they do not have any system permissions. However, users with admin rights can specify which portals the contact is allowed to access and grant system permissions when creating or editing a contact. This makes a contact a system user with corresponding permissions.
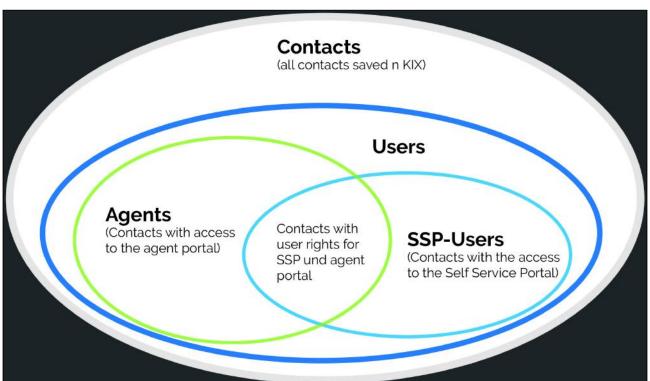
Fig.: Representation of users and contacts

Both the agents and the contacts with access to the agent and/or self-service portal are managed together as system users in the user administration, which reduces the administrative maintenance effort. The table in the dashboard lists all system users as search results. Who has access to which portal is indicated by a check mark.

To search for a user, enter the user's name or other user parameters such as email address or user login in the search field. If you want all users to be displayed, enter the asterisk (*) as a wildcard in the search field. Loading all users may take a moment.
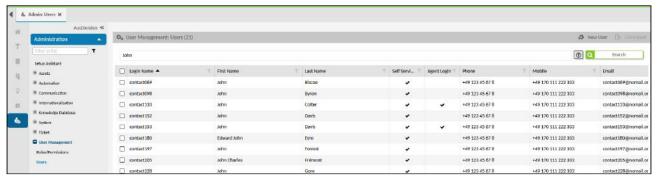
Fig.: The overview of the users created in the system with the identification of the portal access.

> ⚠️ **Attention!**
>
> The initial user works in the system as a root user with **complete** authorization to all system areas. We recommend that you only work with this user in exceptional cases. Therefore, create an additional super user (role "Super User") if you have not already created one in the setup assistant. If necessary, also create an admin user (role "System Admin") and work under these two roles in order not to endanger the security of the system.
> Do **not delete** the initial user! Basic functions of KIX are bound to this user, which you would also delete and "undermine".

Each system user must be assigned at least one role so that you can specify their permissions in the system. The zoom view for a user indicates which roles have been assigned to them and their preferences. To open zoom view, click a table entry.

Alongside the general information on a user, you can also pre-configure their preferences (language and team assignments). All users can change these in their account. The zoom view for a user always shows the current values. If the user changes their settings, you can also keep track of them here and edit them again.

> ✅ **Tip**
>
> As an alternative to user management, you can use LDAP/AD directory services (see chapter "Authentication/Authorisation and Connection to Active Directory (see page 671) "). The administration of users in KIX can thus be completely or partially omitted.

15.2.1 Zoom View for a User

You can display the details of a user. Navigate to *User Management > Users* and enter the user you are looking for as a search term in the search field. The use of the asterisk (*) as a wildcard is supported. Entering 3 asterisks (***) lists all users created in the system. Depending on the number of users, this may take a moment.

As a result of your search, a table will open in the content area listing the users found. Click on a user in the table to open his or her detailed view. The buttons in the title bar of the detailed view allow you to edit and create new users. Click on "Edit" if you want to make changes to the user data or change the permissions.

The "Reset user widgets" button resets the changes made by the user to the GUI (e.g. personalisation of the Home Dashboard) to the default configurations. The default values are the configurations defined in the SysConfig keys (menu *System > SysConfig* or *System > GUI Configuration > Agent Portal | Self Service Portal*),
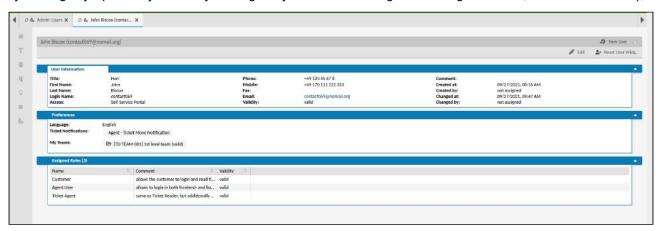
Fig.: Zoom View of a User

In addition to the "User Information" lane containing the agent's general reference data, the zoom view also has other lanes:

Lane	Description
Preferences	You always see the preferences that are currently configured for a user. If a user changes their settings, you can keep track of them here and change them again if required by clicking "Edit".
Assigned Roles	List of all roles assigned to the agent. You can click a role to see what permissions make up the role. Doing so opens the zoom view for the role

15.2.2 Creating and Editing a User and Assigning Roles

The menu *User Management > Users* enables you to create and manage the KIX users so that they can log in to the system and work with KIX in line with the rights they have been assigned. On the one hand, these can be the agents of your own company, but also customer contacts with access to the agent or self-service portal. Both are managed together as users in the system, which simplifies administrative maintenance.

Each new user is initially assigned to your own company ("MyOrganisation") and automatically receives the role "Agent User". This role cannot be withdrawn. A user must have at least the role "Agent User" to be able to log in.

If a user is assigned a usage context (access), he or she automatically receives the basic roles required for it. They do not have to be explicitly assigned manually. If the access "Agent Portal" is selected, the role "Agent User" is assigned. If the access "Self Service Portal" is selected, the role "Customer" is assigned.

> ⚠️ **Important**
>
> Please note that the login names must be unique in the system and cannot be changed retrospectively as they act as identifiers in the database.

To create a new user, proceed as follows:

1. In explorer, navigate to *User Management > Users.* In the content area, a table listing all the users saved in the system opens.
2. In table, click "New User". A form dialog for creating the user opens.
3. Complete the form (see table below).
4. Under "Access", select the portals to which the user is to have access and set the access data.
   The roles "Agent User" and "Customer" are automatically set according to your selection. These roles are the minimum required to log in to the respective portal.
   You can add more roles to extend the user's permissions.
5. Set the validity to "valid" and save your entries with "Save".

The user has now been created and can work with KIX in line with the permissions they have been assigned.

To assign a user additional roles (permissions), proceed as follows:

1. In Explorer, navigate to *User Management > Users.*
2. Search for the desired user in the search box. As a result of your search, the users found are listed in the table in the content area.
3. In table, click user to which you want to assign roles. The zoom view opens.
4. Click "Edit" in title bar of opened zoom view. A form dialog for editing the user opens.
5. Under "Role Assignment", open the "Roles" selection field and select the roles you want to assign to the user. Multiple selection is possible (see also Overview of Pre-Configured roles in KIX).
6. To remove one or more roles, click on the small cross in the label of a role.

7. Click "Save" to save role assignment.

The user now has the corresponding rights. The user may need to refresh their system for the changes to take effect.

To edit a user, proceed as follows:

1. In Explorer, navigate to *User Management > Users.*
2. Search for the desired user in the search box. As a result of your search, the found users are listed in the table in the content area.
3. In table, click user to be edited. The zoom view opens.
4. Click "Edit" in title bar of opened zoom view. A form dialog for editing the user opens.
5. Change the details and set the validity to "valid". The changes are effective immediately.

The form dialog contains the following input fields, among others:

Field (selection)	Description
Title	If applicable, enter user's personal title, e.g. "Dipl.-Ing." or "B.A."
First Name	Enter user's first name (mandatory field).
Last Name	Enter user's last name (mandatory field)
Organisations	Assign an organisation to the user. This is usually the company for which the user works. You can select several organisations if the user works for different organisations. The contact's organisations stored here are available for selection on the ticket. To select the organisation, enter at least 3 letters to select from a list of already created customers. It is possible to use the asterisk as a placeholder, e.g. "Mu*". Entering 3 asterisks (***) lists all organisations. This may take a moment.
Primary Organisation	The primary organisation is the company for which the user primarily works. Select one of the organisations indicated under "Organisations".
Phone, Mobile, Fax	These fields are for saving the phone number, cell phone number, and fax number of the user.
Email	The user's email address, e.g. john.doe@kixdesk.com. The user uses this email address to communicate with customers or employees. Please ensure that you only enter valid email addresses.

Field (selection)	Description
Email1 to Email5	Specify secondary email addresses for receiving emails. Up to 5 additional email addresses can be stored. All fields are optional. • When an email is received in the system, all stored email addresses (primary and secondary) are searched in order to assign the ticket to the corresponding contact. • The email addresses can be referenced via KIX placeholders (e.g. `<KIX_CONTACT_Email3>`). • Within a contact, an email address can be specified several times. • An already assigned email address cannot be assigned to another contact. Unless the plausibility check in the SysConfig key: ContactEmailUniqueCheck has been deactivated (**Attention**: Observe notes (see page 250) !). • The CSV import is possible. • A contact search for "email" (also full text) searches across all email addresses. • In the tables (e.g. Organisations Dashboard), the primary email address is displayed. Additional columns for displaying email addresses 1 to 5 can be configured in the table (see also: Configuring Dashboard Tables (see page 497)). • During active email dispatch (manual recipient selection), only the primary email address is available for selection. • System notifications are sent via the primary email address.
Street, Zip, City, Country	These fields are for entering the user's address details.
Avatar example	You can upload a photo of the user that can be displayed in the sidebar (see Configuring of Sidebar (see page 412)).
Comment	Notes regarding the user.
Validity	• **valid:** The user can log in and work on the system in line with the rights they have been assigned. • **invalid/invali-temporarily:** The user (temporarily) cannot work on the system (vacation, illness, guest agent; user leaves the temporarily company). In the Admin module, you can also select invalid users in selection fields, e.g. in jobs or notifications. As a result, you can create users as part of preparatory work, and then set them to "valid" at the relevant point in time.

Field (selection)	Description
Access	Set the context of use. It determines which portals the user has access to (multiple selection possible). Without selecting the access, only a contact is created in the Organisations module - without access and permissions. If access is selected, a contact becomes an agent. Depending on the selection of access, further input fields are displayed. **Agent portal:** • Allows access to the agent portal • The user is automatically assigned the basic role "Agent User". • Defaults for the user's personal settings can be stored. **Self Service Portal:** • Allows access to the Self Service Portal • The user is automatically assigned the basic role "Customer".
Login Name	The login name with which the user logs in to the system (mandatory field). This name must be unique in the system. The fields required to log in are case-sensitive.
Password	The password with which the user logs in to the system. Each user is able to change their password themselves at regular intervals in their preferences. 🚫👁 Clicking these buttons shows or hides the password when (re)assigned. **Please Note**:After changing the password, you may also need to change the password for the user in the web form, too. **Login attempts:** In terms of the login attempts, there is no restriction on the number of login attempts. Assign new password if user has forgotten their password. **Tip:** You can use the SysConfig key "*PreferencesGroups###Password*" to define the rules for the password.
Roles	Define the user's access permissions by selecting one or more roles. All roles set to "valid" are available for selection. To remove a role, click "x" in role designation (see also Overview of Pre-Configured roles in KIX (see page 640)). **Please note:** The role "System Admin" is a user with advanced permissions for the Admin area.

Field (selection)	Description
Language	Here, set the language in which user uses KIX. The user can change this default setting at any time in their preferences.
My Teams	Specify the teams to which the user should be assigned. The user will receive notifications and tickets from these teams by default. The user can change this setting themselves in their preferences. All teams to which the user has at least read rights are available for selection.
Notification for Tickets	Specify which notifications the user should receive by default. The user can change this setting themselves in their preferences. All notifications created in the menu *Automation > Notifications* where the Option "Show in agent preferences" is checked off, are available for selection.
User Token (KIX Pro)	Users with access to the agent portal receive a personal token. This is used to identify the user in queries (e.g. reporting), not for API authentication. It is automatically generated when a new user is created or a user is edited who does not have a personal token.
	KIX Pro Reporting (see page 702) uses the user token to form a URL per output format together with the token of the report definition. The URL can be used to import reports into third-party systems (e.g. MS Excel) without additional user authentication. In the event of an access violation, a corresponding message is displayed in the retrieving system. Further information can be found in the KIX log.
	The token can be regenerated at any time, e.g. if there is suspicion that unauthorised third parties have gained knowledge of the token.
	Each user can view and regenerate the token in their personal settings.

15.2.3 Asign user rights to contacts

You can grant access to the system to selected contacts. This also allows a contact to work in the Agent Portal and/or Self Service Portal.

If an agent has the appropriate rights, he can define which portals the contact is allowed to access when creating or editing a contact. In addition, he can store the login data and grant the system permissions. This makes a contact a system user. Contacts with access to the Self Service Portal are automatically assigned the "Customer" role. This role cannot be removed.

The table in the *User Management > Users* menu contains all contacts with user rights. This includes both agents and users of the Self Service Portal. Both are managed together as users in the system, which simplifies administrative maintenance. Tick marks in the table indicate which portal a user has access to. In the overview of contacts in the customer module, a check mark also indicates whether and to which portals a contact has access rights.
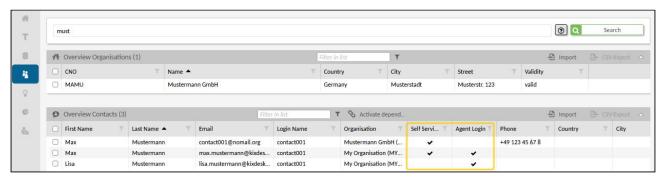
Fig.: Overview of contacts with access rights to the system

How to assign user rights to a <u>new</u> contact:

1. Create a new contact:
   Navigate to the "Customers" module and click on "New contact".
   Optionally, you can create a new contact via the "+New" button > "New contact" tab.
2. In the dialog that opens, enter the contact information for the contact.
3. Under "Access", select the portals to which the contact should have access.
   Depending on your selection, further input fields for entering the access data are displayed.
4. Enter the access data such as password, user name, role assignment and personal settings.
   Detailed information on this is provided in the table in the chapter Creating and editing a user and assigning roles. (see page 664)

How to assign user rights to an <u>existing</u> contact:

1. Navigate to the "Customers" module and find the desired contact using the search function.
2. Click the contact in the contacts overview to open its detailed view.
3. Click on the "Edit" button. A form dialog opens where you can edit the contact.
4. Under "Access", select the portals to which the contact should have access.
   Depending on your selection, further input fields for entering the access data are displayed.

5. Enter the access data such as password, user name, role assignment and personal settings. Detailed information on this is provided in the table in the chapter Creating and editing a user and assigning roles. (see page 664)

Other input fields are shown depending on which portal has been selected:

Field	Description
Login Name	The login name with which the user logs in to the system. This name must be unique in the system. The fields required to log in are case-sensitive.
Password	The password with which the user logs in to the system. Each user is able to change their password themselves at regular intervals in their preferences.
Roles	Define the user's access permissions by selecting one or more roles. All roles set to "valid" are available for selection. **Please note:** The role "System Admin" is a user with advanced permissions for the Admin area.
Language	Here, set the language in which user uses KIX. The user can change this default setting at any time in their preferences.
My Teams	Specify the teams to which the user should be assigned. The user will receive notifications and tickets from these teams by default. The user can change this setting themselves in their preferences. All teams to which the user has at least read rights are available for selection.
Notification for Tickets	Specify which notifications the user should receive by default. The user can change this setting themselves in their preferences. All notifications created in the menu *Automation > Notifications* where the option "Show in agent preferences" is checked off, are available for selection.

15.2.4 Authentication/Authorisation and Connection to Active Directory

The authentication and authorisation of users in KIX is controlled via so-called authentication and synchronisation backends. By default, authentication against KIX's own user management is used.

Authentication and authorisation are set up by configuring the JSON string in the key: "*Authentication##000-Default*" (menue: *System > SysConfig*).

Content on this page:

15.2.4.1 Use directory service (AD, LDAP)

Configuration key:	Authentication###000-Default

You are able to use the user data and permissions from your LDAP/AD server (domain controller) to manage the user logins. The advantage here lies in the centralized user management, both for your infrastructure and for KIX. As a result, there is no need to maintain users in KIX as this is done centrally in the Active Directory. However, the initial user is always required.

You can set up unlimited authentication backends and configure them separately (see example below). The prerequisite is at least one LDAP/AD server accessible from the backend of KIX with configured users and rights. Several hosts of different formats can be specified. In the KIX log, you can find out which backend performed the authentication. For this purpose, the name of the backend is noted in the log entry of a successful authentication (menu *System > Logs > File: kix.log.[yyyy-mm]*).

Both the Microsoft Active Directory (AD) and all common LDAP directory services are supported. KIX's own database serves as a fallback for authorisation (see note above).

There is no limit to the number of users. Contact and user data are stored locally so that no delays occur due to latencies in communication with LDAP/AD.

When a user logs in to KIX, the data in KIX is compared with that of the configured LDAP servers. If there are discrepancies, the data in KIX is synced with that of the LDAP servers. The sequence in which the LDAP servers have been stated defines which backend is checked first (from top to bottom). If an error occurs here (unable to reach server, user does not exist, etc.), the next backend will be checked.
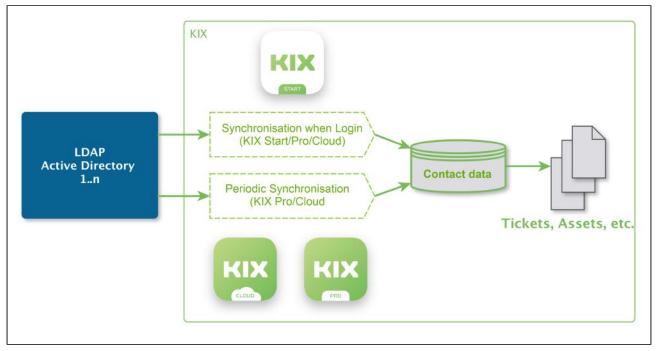
Fig.: Scheme of LDAP/AD connection to KIX 18

The JSON string contained in the SysConfig key "*Authentication##000-Default*" is already fundamentally prepared. For the use of an LDAP directory service, only the corresponding parameters such as domain, path, directory and attributes must be specified. KIX automatically assigns the roles stored in KIX to the user. This is based on the user's group assignments in LDAP/AD. Which attributes (properties) are mapped to which roles must be specified when configuring the JSON string.

> ⚠ **Please note**
>
> In this context, passwords for user logins can only be changed via the LDAP. Any changes made to them in KIX will have no effect.

> ✅ **Tip**
>
> In KIX Pro, contact data can be synchronised with LDAP/AD in addition to user data. The maintenance of users and contacts is thus carried out centrally via the LDAP/AD and does not require the user to actively log in to make their contact data available in KIX.

Configure connection
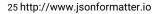
To connect KIX to an LDAP/AD server, proceed as follows:

1. In explorer, navigate to *System > SysConfig*.

2. Find key "*Authentication###000-Defaul*t". Click key to open it for editing. A form dialog that already contains a pre-prepared JSON string in the field "Wert" (Value) opens.

3. To make this string easier to edit, copy string to JSON editor (e.g. www.jsonformatter.io[25])

---

25 http://www.jsonformatter.io

4. Amend configuration by specifying host and connection parameters (see Example Configuration).

```
{
    "Config": {
        "AccessAttr": "uid",
        "AlwaysFilter": "(!objectclass=computer)",
        "BaseDN": "dc=cape-it,dc=de",
        "Charset": "utf-8",
        "Die": 0,
        "GroupDN": "cn=kixallow,ou=posixGroups,dc=example,de=com",
        "Host": "zeus.intra.cape-it.de",
        "Params": {
            "async": 0,
            "port": 7389,
            "timeout": 120,
            "version": 3
        },
        "SearchUserDN": "uid=<LDAP_LOGIN>,cn=staff,cn=users,dc=cape-it,dc=de",
        "SearchUserPw": "<LDAP_PASSWORD>",
        "UID": "uid",
        "UserAttr": "UID",
        "UserSuffix": ""
    },
    "Enabled": 1,
    "Module": "Kernel::System::Auth::LDAP",
    "Name": "CapeIT LDAP",
    "Sync": [
        {
            "Config": {
                "AttributeBasedRoleSync": {___},
                "ContactUserSync": {___},
                "GroupDNBasedUsageContextSync": {
                    "SomeDN": {
                        "IsAgent": 1,
                        "IsCustomer": 0
                    },
                    "SomeOtherDN": {___}
                }
            },
            "Enabled": 1,
            "Module": "Kernel::System::Auth::Sync::LDAP"
        }
    ]
},
{
    "Config": {
        "CryptType": "sha2"
    },
    "Enabled": 1,
    "Module": "Kernel::System::Auth::DB",
    "Name": "Local Database"
}
```

> ⬥ **Attention!**
>
> **Changes to the last code block can lead to exclusion from the system!**
> The KIX database (last backend in the JSON ring) should normally neither be removed nor disabled. It serves as a fallback in case the LDAP server is not available or the user is not present. Only change the configuration in exceptional cases! Otherwise you could **lock yourself out of the system completely.**

5. Minimise the JSON string and copy it (without spaces and breaks) back into the "Value" field. Save the change.

The default configuration has now been replaced with a modified configuration. When a user logs in to KIX, KIX will communicate with the directory services linked in this way and will check the logins.

Attributes of the configuration

The following attributes can be used for authentication against an LDAP/AD. The attributes listed in the example (server, LDAP attributes, etc.) must be replaced with your respective values, and blanks and line breaks must be deleted.

UsageContext

Defines which Auth-Config is used for which Usage Context (optional specification).

Set the UsageContext if a user is both a customer and an agent and different authentication methods are used for both usage contexts (Agent: KIX-DB, SSP: LDAP). This allows you to control the user's access to the portals:

- Context of use "Customer": The user cannot log on to the agent portal.
- Usage context "Agent": The user cannot log in to the Self Service Portal
- No specification of the context of use: The user can log in to both portals.

Value (Example)	Notes

{
  "Config": {...},
  "Enabled": 1,
  "Module": "Kernel::System::Auth::LDAP",
  "Name": "openLDAP|nur Agent",
  "UsageContext": "Agent",
  "Sync": [...]
}
	The specification has no effect on the assigned usage contexts in the synchronisation. It only restricts the use of the auth. backend.

Module

Specify LDAP authentication.

Value (Example)
`Kernel::System::Auth::LDAP`

Host

Host name, IP address or URI of an LDAP/AD server.

Value (Example)	Notes
`ldap.myCompany.de`	Alternatively, a list of hosts or URIs can be specified: `["ldaps://dc1.example.com", "ldaps://dc1.example.com"]` If LDAP is to be queried over SSL, enter "`ldaps://<actual hostname>`" as the hostname.

BaseDN

Point of entry into the directory structure.

Value (Example)
`dc=example,dc=com`

GroupDN

Group to which an entry must belong in order to be authenticated.

Value (Example)
`cn=kixallow,ou=posixGroups,dc=example,dc=com`

AccessAttr

With LDAP retrieval of the group from GroupDN, the user assignments appear under Attribute u `id` or `member` (Active Directory: `member`)

Value (Example)
`uid	member`

UserAttr

For LDAP-posixGroups UID, For Not-LDAP-posixGroups with full DN (Active Directory: DN)

Value (Example)
UID | DN

UserSuffix

Is appended to each login entry during authentication and sent to the LDAP/AD server, e.g. user enters "m.power", "m.power@ValueInUserSuffix" is sent to the AD server.

Value (Example)
<empty> | @somedomain.tld

SearchUserDN

User DN of the user, which KIX uses to connect to the LDAP server in order to run the search.

Value (Example)
cn=kix,ou=user,dc=example,dc=com

SearchUserPw

Password of the user, which KIX uses to connect to the LDAP server in order to run the search.

Value (Example)
some_password

UID

Name of the LDAP attribute used to uniquely identify an LDAP entry (user login). Usually "UID" for LDAP and "sAMAccountName" for Active Directories.

If no *UserLogin* is specified in the Sync Map, a UserLogin is created based on the *UID* in the KIX database and set to invalid. *UID* and *UserLogin* can point to the same attribute without problems.

The *UID* serves as a fallback if no other authentication is possible.

Value (Example)
UID	sAMAccountName

AuthAttr

Alternative, unique identifier for login. Enables the use of a different, unique contact attribute than the user login stored in the system for the user (only applies to authentication via AD).

If, for example, "mail" is specified, the user's email address stored in the user administration is accepted for authentication.

If this attribute is not set or if this authentication fails, the setting made under UID is used (fallback).

Value (Example)	Notes
<empty>	mail	userPrincipalName	principalName	...	**Tip:** If numerical, hard-to-remember user IDs are used, the user can always log in with their current email address. If this changes (but the user name remains the same), the user can still log in.

AlwaysFilter

Permanent filter used for LDAP/AD requests (see https://tools.ietf.org/html/rfc2254).

You can also enter group filters here. Further examples are available here: http://www.selfadsi.de/ldap-filter.htm.

Value (Example)	Notes
"(&(mail=*)(sn=*))"	• All non deactivated user accounts: `(&(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))` • All user entries where "departmentNumber" and "mail" have been set: `(&(objectClass=user)(departmentNumber=*)(mail=*))`

Params

LDAP/AD parameters for the authentication module.

Value (Example)	Notes
```{ "port": "389", "timeout": "10", "async": "0", "version": "3" }```	"636" should be entered for LDAPS (LDAP via SSL) and "`ldaps://<actual hostname>`" should be specified as the host name (see above). It is therefore not necessary to specify *"scheme": "ldaps"*.

ContactUserSync

Data synchronisation (Sync)

Matching of a user's contact attributes upon successful authentication.

The specifications for the mapping are made as a key-value pair: `"KIXAttributName" :` `"LDAPAttributName".`

The contact entry assigned to a user account is created or updated. The specification of the following attributes is required at least for this purpose:

- Email
- Firstname
- Lastname
- UserLogin

> ⚠ **Attention**: If these details are not set in the LDAP/AD entry, no contact and thus no user entry can
> be created if they are not present. It is then NOT possible to use the system for contacts and users
> that have (not yet) been entered manually or otherwise.

Value (Example)

```
"ContactUserSync": {
 "Email": "mail",
 "Email1": "SET:mail@example.com",
 "Email3": "mail",
 "Email5": "automatedmail",
 "Title": "title",
 "Firstname": "givenname",
 "Lastname": "sn",
 "Street": "streetAddress",
 "City": "l",
 "Zip": "postalCode",
 "Phone": "telephoneNumber",
 "Mobile": "mobile",
 "Fax": "facsimileTelephoneNumber",
 "UserLogin": "sAMAccountName",
 "PrimaryOrganisationID": "SET:123",
 "OrganisationIDs": [
     "department",
     "SET:13"
 ],
 "DynamicField_Source": "SET:ActiveDirectory1",
 "DynamicField_XYZ": "department",
 "DynamicField_ZIPCity":"CONCAT: postalCode}-{l}-DE"
}
```

Notes:

1. All attributes specified in the mapping must exist. Otherwise, no correct assignment can take place.
   Also make sure that the spelling is correct.

2. Concrete values are specified with a preceding " `SET:` " (e.g.: " `Email1":`
   `"SET:mail@example.com"`).

3. For the target attribute "Email" and for "Email1" to "Email5", the individual array values are
   automatically split if an array value exists in the LDAP source attribute. A separate configuration is
   not required.

4. Each KIX attribute can be composed of n LDAP attributes and separator strings. This means that
   multiple attributes (arrays) can be written in LDAP as a concatenated character string in a single
   target attribute.

1. Prerequisite: The target attribute must support the supplied data type and the definition of the dynamic field.

2. The array values are concatenated as a single value with a separator symbol (analogue "CONCAT" → see below).

3. Directive: `"ARRAYJOIN[<SeparatorString>]:{<ADAttributeName>}"`

   a. `ARRAYJOIN` : Command for creating a concatenated character string (array)

   b. `SeparatorString` : Specification of the separator (comma, semicolon, hyphen, etc. including any spaces). Specification in square brackets.

   c. `ADAttributname` : Name of the LDAP/AD attribute to be concatenated. The attribute is specified in curly brackets.

4. Example: `"objectClass"` is an attribute that occurs several times in LDAP/AD.

   a. The notation: `"DynamicField_Source": "ARRAYJOIN[, ]:{objectClass}"` creates an array from all LDAP/AD attributes with the designation "objectClass" and writes this as a comma-separated list in the dynamic field "DynamicField_Source".

5. **Note**: the "mail" attribute supports a maximum of 6 values in the array.

5. The **assignment to an primary organisation** is defined by means of "PrimaryOrganisationID". In addition to an LDAP/AD attribute, a fixed value can also be configured (" `PrimaryOrganisationID": "SET:123` "). "123" is interpreted as the ID of an organisation known in KIX. However, if an LDAP/AD attribute is specified (" `PrimaryOrganisationID": "extensionattribute12` "), the value of the attribute is interpreted as a customer number and evaluated by lookup to a KIX-internal organisation ID. If nothing was found, then the value contained in the LDAP/AD attribute is interpreted as an organisation ID. If there is no organisation for this either, the fallback to organisation ID 1 takes place.

6. The **assignment to *n* further organisations** is defined by means of " `OrganisationIDs` ". For this purpose, an array of LDAP/AD attribute names or fixed Org. ID assignments is specified. This allows *n* LDAP attributes or SET values to be configured. The lookup behaviour is identical to " `PrimaryOrganisationID` ". For each match, the information stored at the contact is completely replaced. If the " `OrganisationIDs` " specification is not maintained in the mapping, the value specified in PrimaryOrganisationID is automatically added to any existing entries in the list of organisations. In this case, no complete replacement takes place.

7. **Dynamic fields** can be filled with attribute values from the LDAP/AD. Thus, for example, dynamic fields can be used for the object type contact in order to

   1. store the AD from which the contact originates in a contact entry

   2. or to provide further information such as supervisor, social media handle, room number, cost centre, etc. also in KIX.

   3. KIX provides the DF "Source" on contacts for storing the AD/LDAP source. Others can be set up according to individual needs.

8. **Concatenation (chaining) of LDAP attributes:** Each KIX attribute can be composed of *n* LDAP attributes and separator strings by means of "`CONCAT`", e.g. to combine several LDAP attributes in one KIX contact attribute.

   1. Instead of specifying an LDAP attribute directly, these are named in curly brackets.
      Example: "`PrimaryOrganisationID": "CONCAT:{givenName} from the {company}-{department}"`).

   2. All characters not contained within curly brackets are transferred directly into the contact attribute.
      Example: `"CONCAT:{postalCode}-{l}-DE"` becomes "09113-Chemnitz-DE".

GroupDNBasedRoleSync

Data synchronisation (Sync)

Assignment of authorization roles in KIX based on affiliation to groups in AD/LDAP.

- Level 1 key defines the AD/LDAP group
- Level 2 key names the KIX authorization role and the assignment of the role by means of value "1".
  - setting a "0" leads to explicit revocation of the authorization role

Value (Example)	Notes

"GroupDNBasedRoleSync": {
 	
 "dc=serviceteam,dc=example,dc=com": {
    "Ticket Agent": 1,
    "Asset Maintainer": 1,
    "Customer Manager": 1,
    "FAQ Editor": 1
  }
}
	The basic roles "Agent User" and "Customer" are assigned exclusively and indirectly via the assignment of the usage context (UsageContext via IsAgent/IsCustomer). To avoid inconsistent behaviour, these roles should not be assigned in explicit mappings as in GroupDNBasedRoleSync - unless the GroupDNBasedUsageContextSync parameter is not specified at all in the configuration. There may then be entries in the log that the role has not been assigned. This is due to an attempted double allocation and can be ignored ("Known Error").

AttributeBasedRoleSync

Data synchronisation (Sync)

Assignment of authorization roles in KIX based on pattern matching in AD/LDAP attributes.

- Level 1 key defines the AD/LDAP attribute.
- Level 2 key defines the pattern to be made.
- Level 3 key names the KIX authorization role and the assignment of the role using value "1".
  - setting a "0" leads to explicit revocation of the authorization role

Value (Example)

```
"AttributeBasedRoleSync":{
    "sAMAccountName": {
      "m.mustermann": {
        "Superuser": 1
      }
    }
}
```

GroupDNBasedUsageContextSync

Data synchronisation (Sync)

Automatic setting of the usage context from the LDAP/AD group membership

Specifying a "0" value removes the usage context regardless of any other assignments that may exist through other group memberships.

The specification of a superordinate LDAP group overwrites specifications of a subordinate group.

Value (Example)	Notes

"GroupDNBasedUsageContextSync": {
 	
 "cn=agentsonly,dc=example,dc=com" :{
    "IsAgent": 1,
    "IsCustomer": 0
  },
  "cn=agents,dc=example,dc=com": {
    "IsAgent": 1
  },
 	
 "cn=customer,dc=example,dc=com": {
    "IsCustomer": 1
  }
}
	1. Assigning a context of use already assigns the required basic roles "Agent User" or "Customer" to the user. Changes to the context of use result in the withdrawal or change of the respective basic role.
	
	2. As of v27, the setting of the usage context can be completely omitted. If the specification "GroupDNBasedUsageContextSync" is completely missing, the usage contexts assigned to the roles to be synchronised based on GroupDNBasedRoleSync will be assigned. For example, if the role "Ticket Agent" is assigned, the usage context "IsAgent" is set; if the role "Customer" is assigned, the usage context "IsCustomer" is set.

Example configuration

The following configuration example shows how the connection to an LDAP server could look. Two authentication backends are configured: one for the self-service portal and one for the agent portal (see attribute "UsageContext").

```
 1  [
 2    {
 3      "Enabled": 1,
 4      "Module": "Kernel::System::Auth::LDAP",
 5      "Name": "1st Backend: Customer Contact From LDAP",
 6      "UsageContext": "Customer"
 7      "Config": {
 8        "AlwaysFilter": "(&(objectclass=person)(mail=*))",
 9        "BaseDN": "dc=example,dc=com",
10        "Charset": "utf-8",
11        "Die": 1,
12        "Host": "ldap.example.com",
13        "Params": {
14          "async": 0,
15          "port": 389,
16          "timeout": 10,
17          "version": 3
18        },
19        "SearchUserDN": "cn=read-only-admin,dc=example,dc=com",
20        "SearchUserPw": "password",
21        "GroupDN": "dc=customercontacts,dc=example,dc=com",
22        "AccessAttr": "uid",
23        "UID": "uid",
24        "UserAttr": "UID",
25        "AuthAttr": "mail",
26        "UserSuffix": ""
27      },
28      "Sync": [
29        {
30          "Enabled": 1,
31          "Module": "Kernel::System::Auth::Sync::LDAP",
32          "Config": {
33            "ContactUserSync": {
34              "Email": "mail",
35              "Firstname": "givenname",
36              "Lastname": "sn",
37              "UserLogin": "uid",
38              "PrimaryOrganisationID": "SET:13",
39              "OrganisationIDs": [
40                  "department",
41                  "SET:13"
42              ],
43              "DynamicField_Source": "SET:Backend001-LDAP"
44            },
```

```
45              "GroupDNBasedRoleSync": {
46                "dc=customercontacts,dc=example,dc=com": {
47                  "Some Customer Role": 1,
48                }
49              },
50              "GroupDNBasedUsageContextSync": {
51                "dc=serviceteam,dc=example,dc=com": {
52                  "IsAgent": 1,
53                  "IsCustomer": 1
54                },
55                "dc=agentsonly,dc=example,dc=com": {
56                  "IsAgent": 1,
57                  "IsCustomer": 0
58                },
59                "dc=customercontacts,dc=example,dc=com": {
60                  "IsCustomer": 1
61                },
62                "dc=externals,dc=example,dc=com": {
63                  "IsCustomer": 1
64                }
65              }
66            }
67          }
68        ]
69      },
70      {
71        "Enabled": 1,
72        "Module": "Kernel::System::Auth::LDAP",
73        "Name": "2nd Backend: Agent Users from Active Directory (via multiple
    LDAPS)",
74        "UsageContext": "Agent",
75        "Config": {
76          "AlwaysFilter": "(&(objectclass=person)(mail=*))",
77          "BaseDN": "dc=example,dc=com",
78          "Charset": "utf-8",
79          "Die": 1,
80          "Host": [
81            "ldaps://dc1.example.com",
82            "ldaps://dc2.example.com"
83          ],
84          "Params": {
85            "async": 0,
86            "port": 636,
87            "timeout": 10,
88            "version": 3
89          },
90          "SearchUserDN": "cn=read-only-admin,dc=example,dc=com",
91          "SearchUserPw": "password",
92          "GroupDN": "dc=serviceteam,dc=example,dc=com",
93          "AccessAttr": "member",
94          "UID": "sAMAccountName",
95          "UserAttr": "DN",
96          "UserSuffix": ""
```

```
 97          },
 98        "Sync": [
 99          {
100            "Enabled": 1,
101            "Module": "Kernel::System::Auth::Sync::LDAP",
102            "Config": {
103              "ContactUserSync": {
104                "Email": "mail",
105                "Title": "title",
106                "Firstname": "givenname",
107                "Lastname": "sn",
108                "Street": "streetAddress",
109                "City": "l",
110                "Zip": "postalCode",
111                "Phone": "telephoneNumber",
112                "Mobile": "mobile",
113                "Fax": "facsimileTelephoneNumber",
114                "UserLogin": "sAMAccountName",
115                "PrimaryOrganisationID": "department",
116                "OrganisationIDs": [
117                  "department",
118                  "SET:13"
119                ],
120                "DynamicField_Source": "SET:Backend002-AD"
121              },
122              "GroupDNBasedRoleSync": {
123                "dc=serviceteam,dc=example,dc=com": {
124                  "News Manager": 1,
125                  "Ticket Agent": 1,
126                  "Asset Maintainer": 1,
127                  "Customer Manager": 1,
128                  "FAQ Editor": 1
129                }
130              },
131              "AttributeBasedRoleSync": {
132                "sAMAccountName": {
133                  "j.doe": {
134                    "Superuser": 1
135                  }
136                }
137              }
138            }
139          }
140        ]
141      },
142      {
143        "Enabled": 1,
144        "Module": "Kernel::System::Auth::DB",
145        "Name": "Local Database",
146        "Config": {
147          "CryptType": "sha2"
148        }
149      }
```

```
  150   ]
```

15.2.4.2   Automatic login (ValidUser)

With the help of the authentication backend "ValidUser", access can be granted solely on the basis of the login name and the calling IP address. Practical application scenarios are, for example:

1. A self-service portal that only allows messages to be created, but does not display any open operations.
2. Clients that are used with the same IP address by the same person and do not have to meet any or only very low security requirements.

The prerequisites are that a valid user account is used and the call satisfies the defined conditions (login name/s and client IP address).

> ⬥ **Attention!**
>
> The authentication backend "ValidUser" is initially inactive. Use this only in limited exceptional cases for user accounts with restricted authorisations!
> Limit access to **actually necessary** users and IP addresses to avoid creating an unintentional security gap!

Activate SSO in the frontend

"ValidUser" is one option for Single Sign On (SSO). SSO must be explicitly activated for the Agent and Self Service Portal. To do this, you must activate the respective configurations in the environment file of your KIX On-Premises configuration. A restart of the stack is then required.

Environment

```
SSO_ENABLED=true
SSO_ENABLED_SSP=true
```

Attributes of the configuration

The following attributes can be used for automatic login:

Attribute	Value	Description
Module	`Kernel::System::Auth::ValidUser`	Set automatic login

Attribute	Value	Description
Enabled	`0	1`	Disables / enables the authentication backend
UsageContext	`"Agent"	"Customer"`	Defines which authentication configuration is used for which usage context (optional specification).
		Set the UsageContext if a user is both a customer and an agent and different authentication methods are used for both usage contexts (Agent: KIX-DB, SSP: LDAP). This allows you to control the user's access to the portals:
		• Context of use "Customer": The user cannot log on to the agent portal.
		• Usage context "Agent": The user cannot log in to the Self Service Portal
		• No specification of the context of use: The user can log in to both portals.
RelevantClientIPs	`".*"`	Regular expression to which the IP of the client must correspond.
		"X-Forwarded-For" headers are also evaluated to restrict to specific client IP addresses.
RelevantUsers	`".*"`	Regular expression to which the user name must correspond.

> **ⓘ Note**
>
> Instead of the URL parameter "username", an **HTTP header "X-KIX-User"** can also be evaluated. This
> means that authentication can be left to upstream proxy servers and a kind of single sign-on can be
> mapped.
> In this case, make sure that **the header is not taken over by external requests** to prevent
> unauthorized login!

Example configuration

In the following example, the Self Service Portal user "mamu" can log on directly to the Self Service Portal
without entering a password if he comes from the IP addresses `"172.21.16.*"` . He only has to call up
the URL to the Self Service Portal, e.g.:

- On-Premises: http://your.kix-docker-host.org:20002/login?username=mamu&password=not_relevant
- KIX.Cloud: https://your.agentenportal-ssp.kix.cloud/login?username=mamu&password=not_relevant

A user with the login "mamu" and access to the Self Service Portal must be created in the user
administration.

```
 1  [
 2      # ...other authentication backends...
 3    {
 4      "Name": "Valid User",
 5      "Module": "Kernel::System::Auth::ValidUser",
 6      "Enabled": 1,
 7      "UsageContext": "Customer",
 8      "Config": {
 9        "RelevantClientIPs": [
10          "^172.21.16.\d\d?\d?$"
11        ],
12        "RelevantUsers": [
13          "(mamu | login\d+)"
14        ]
15      }
16    },
17    # ...other authentication backends...
18  ]
```

15.2.5 Delete contacts and users

You can remove contacts and users from the database via a command line call, e.g. to manually clean up errors after LDAP/AD or Auth synchronisations. The procedure is to be understood as a workaround for the not yet existing deletion of contacts directly in the application and should only be used if actually required.

The following command line call is provided as an example and can be made from the Docker host. The specifications for EMAIL and DBCONTAINER must be adapted.

> **⚠ Info**
>
> No conditions are checked. Therefore, before using it, make sure that the entries can actually be deleted. Also make sure that no tickets or assets are assigned to the contacts.

Delete contacts and user

```
user@dockerhost$ EMAIL="contact0815@company.com"
user@dockerhost$ DBCONTAINER="kixpro_db_1"
user@dockerhost$ docker exec -it $DBCONTAINER psql -h $DBCONTAINER -U kix -d kix -c "
    -- delete users-relation...
    DELETE FROM user_preferences
    WHERE user_id IN (SELECT user_id FROM contact WHERE email = '$EMAIL');

    DELETE FROM role_user
    WHERE user_id IN (SELECT user_id FROM contact WHERE email = '$EMAIL');

    DELETE FROM users
    WHERE id IN (SELECT user_id FROM contact WHERE email = '$EMAIL');

    -- delete contact-relation...
    DELETE FROM contact_organisation
    WHERE contact_id IN (SELECT id FROM contact WHERE email = '$EMAIL');

    DELETE FROM contact_preferences
    WHERE contact_id IN (SELECT id FROM contact WHERE email = '$EMAIL');

    DELETE FROM contact WHERE email = '$EMAIL';
"
```

It can happen that user accounts are no longer assigned to a contact. These "pending" users are difficult to find in the interface. If necessary, they can also be deleted via DB.

As with contacts, make sure that no tickets are or were assigned to such a user (ticket history). Editing references of config items (assets) can also prevent deletion. Before executing the following statement, please make sure that

- no ticket or asset is assigned to the user (for the sake of clarity, this is not included here as a condition).
- user accounts have not been deliberately created without a contact assignment.

Remove floating users

```
user@dockerhost$ DBCONTAINER="kixpro_db_1"
user@dockerhost$ docker exec -it $DBCONTAINER psql -h $DBCONTAINER -U kix -d kix -c "
    DELETE FROM user_preferences
    WHERE user_id IN (
        SELECT id FROM users u
        WHERE NOT EXISTS (SELECT id FROM contact WHERE user_id = u.id) AND id > 3
    );

    DELETE FROM role_user
    WHERE user_id IN (
        SELECT id FROM users u
        WHERE NOT EXISTS (SELECT id FROM contact WHERE user_id = u.id) AND id > 3
    );

    DELETE FROM users
    WHERE id IN (
        SELECT id FROM users u
        WHERE NOT EXISTS (SELECT id FROM contact WHERE user_id = u.id) AND id > 3
    );
"
```

15.2.6 Import users or organisations

You can import contacts and organisations in the *agent portal > "Organisations"* module. When importing contact data, you can also create / update users and define their context of use (agent portal, self-service portal).

For differentiation:

- Organisation: your client's company / company
- Contact: address record; Usually, however, employees of an organization without KIX access
- User:
    - Agent with KIX login who works with KIX (address data not required)
    - Contact, with user login for access to the agent and / or self service portal

For the import you need authorizations to maintain users (at least the role of Customer Manager).

You can import existing and new contacts / organizations and define how existing data and blank values are to be dealt with.

If a contact is imported that has a user ID, the contact is assigned to the user. A user who does not exist will be created again. The user's contexts of use are updated accordingly. The roles "Agent User" and "Customer" are added automatically if "IsAgent" or "IsCustomer" is specified.

The import is carried out using a CSV file as the source. The first line of the CSV file (headings) must contain the KIX-internal attributes of the contacts / organizations in an exact manner (note upper/lower case!). At least all mandatory attributes (*) must be specified (exception: see Validity). Otherwise the import will fail.

15.2.6.1 KIX attributes

Attributes of contacts and users:

* Mandatory attribute

Attribute	Value/Description	Hint(s)
City	City	
Comment	optionally comment	
Country	Country	
Email*	valid EMail-Address	

Attribute	Value/Description	Hint(s)
Fax	Telefaxnumber	
Firstname*	Firstname	

Attribute	Value/Description	Hint(s)
IsAgent	1 | 0 Control access to the agent portal (context of use)	• If IsAgent and IsCustomer are not specified or are both 0, the user is created as a contact without permissions. • IsAgent and IsCustomer are only processed if a value is specified in the "UserLogin" attribute.

IsCustomer	1 | 0
	
	Control access to the Self Service Portal (context of use)
		• A user login can only be assigned once. If a UserLogin is assigned several times, the first contact will receive it, and an error message will appear for all others.
		• If a user login is specified for a data record, the user is created or updated at the contact
		    • This gives existing contacts access to the Agent and / or Self Service Portal (depending on the definition in the CSV file).
		    • The context of use of existing users is changed according to the information in the CSV file.
		    • Non-existing users will be created. The new user receives the login name specified under "UserLogin" and access to the portals (as specified in the CSV file).
		
		Note for administrators: The role assignment of the users defined before the import remains after the import. Exception: the definition of "Customer" and "Agent User":
		
		• If IsCustomer = 1, the "Customer" role is automatically added / replaced.
		• If IsAgent = 1, the "Agent User" role is automatically added / replaced.
		
		No further roles are assigned. This must be done afterwards under Admin> User Management> Users or Roles.

Lastname*	Lastname	

Attribute	Value/Description	Hint(s)
Mobile	Mobile phone number of	
Phone	Telephone number of the contact/user	
PrimaryOrganisationID*	Number ID of the user's organisation (Is present if file originates from KIX contact export).	🛈 Enter only PrimaryOrganisationID OR PrimaryOrganisationNumber!
PrimaryOrganisationNumber*	Designation CNR of the assigned organisation (Must already exist in KIX), must already be available in the KIX	
Street	Street	
Title	Title	
UserLogin	User login	
ValidID*	1 (valid) | 2 (invalid) | 3 (temporarily invalid)	🛈 Specify only ValidID OR Validity! If the data record has no validity, this attribute can be set automatically using the selection under "Replace value".
Validity*	valid | invalid | temporarily invalid	
ZIP	Zip Code	

Attributes of organisations:

* Mandatory attribute

Attribute	Description and Hint(s)
City	City
CNR*	Customer number (Number of organisation)
Comment	optionally comment
Country	Country
Name*	Name
Street	Street
URL	URL / Web address
ValidID*	1 (valid) | 2 (invalid) | 3 (temporarily invalid) ℹ Specify only ValidID OR Validity!
Validity*	valid | invalid | temporarily invalid ℹ Specify only ValidID OR Validity!
Zip	Zip Code

Note: If the data record has no validity, this attribute can be set automatically using the selection under "Replace value".

15.2.6.2   Carry out the import

Preconditions:

1. There is a CSV file with the contacts (or organisations) to be imported in your file system.
2. The first line of the CSV file contains the attributes to be imported and mandatory attributes in the correct form (see table above)
3. The contacts to be imported are contained in the CSV file. The mandatory attributes are indicated.

   Tip: Export one or more contacts / organisations beforehand. This means that you have a valid CSV file that you can fill with values.

	A	B	C	D	E	F	G	H	I
1	Email	Firstname	Lastname	IsAgent	IsCustomer	Phone	PrimaryOrganisationID	UserLogin	Validity
2	l.lauterbach@nomail.org	Ludwig	Lauterbach	1	0	+49 123 45 67 8	1	llauterbach	
3	r.regner@nomail.org	Rita	Regner	0	0	+49 123 45 67 8	6		
4	f.frohgemut@nomail.org	Fritz	Frohgemut	0	1		1	ffrohgemut	
5	s.sonnenschein@nomail.org	Susi	Sonnenschein	1	1	+49 987 65 43	5	ssonnenschein	
6	m.lang@nomail.org	Marta	Lang	1	0		5	mlang	
7	r.rudolf@nomail.org	Rebecca	Rudolf	0	0		2		

Fig .: Example of a CSV file for importing contacts

Proceed:

1. Navigate to the "*Organisations*" module in the agent portal
2. In the "Organisations overview" or in the "Contacts overview" click on "Import". An import dialog opens in a new tab.
3. Select the CSV file with the contacts/organisations to be imported as the source.
4. If necessary, adjust the import options. Enter the separators used in the CSV file and the character set used so that KIX can read the CSV file correctly (see table below).
5. You can optionally specify how empty values should be handled and/or which values should be replaced during the import (see table below)
6. Carefully check the contacts/organisations to be imported in the overview!
   ⚠ Important: After the start, the import can neither be stopped nor undone. It is therefore advisable to split large amounts of data into several small imports.
   1. Deactivate the checkmark in the 1st column if you want to exclude individual objects (contacts / organizations) from the import.
   2. Pay attention to the correct display of umlauts, change the character set if necessary.
7. Start the import by clicking on "Start import". The import may take a moment depending on the amount of data. Then the contacts / organizations are imported into KIX.
8. You can now switch to the "Organizations" module or the user management of the admin module, and check and process the imported data records.

The form dialogue contains the following input fields, among others:

Source	Selection of the CSV file with the contacts / organizations to be imported. The file is read in immediately and the contacts are listed in the overview of the contacts to be imported. If the file cannot be read in, you will receive a corresponding message. Check the attribute names and the data records in the CSV file again.
Charset	Specify the character set used in the CSV file. The standard is UTF8. The import of Excel files may require the use of an ISO format. Check the correct display of the letters and characters in the overview of the contacts to be imported. If the display is incorrect, change the character set.
Split Option	Enter the separator(s) with which the individual values are separated from one another (tabulator | comma | semicolon | colon | period). Multiple choices possible.
Text Separator	Enter the character by which the individual data records are separated from each other (double or single apostrophe).
Replace Values	Optionally, specify which values are to be set or replaced. If the data records have no validity, for example, the validity can be set automatically for all imported data records. • **Replace empty value**: Fills empty values of an attribute with the specified values. The attribute to be set must be contained in the CSV and must not be a mandatory field (exception: validity). Example: If the CSV contains the attribute "City", this does not have to be filled with values. The city can be set here for all imported contacts. • **Force:** Sets the specified value for all imported records. Overwrites existing values. For example, sets the validity to "valid" for all imported contacts, even if they were previously created in the system with "(temporarily) invalid". • **Ignore**: Existing data is ignored and not replaced. Use this option if the CSV file contains attribute values that should not be replaced, e.g. Agent login.

15.2.7 SysConfig settings for users

You can influence the behaviour of the system for logged-in users. To do this, use the following SysConfig keys (*System >SysConfig* menu) or console commands (*System > Console* menu).

15.2.7.1 Important SysConfig settings

The following SysConfig settings significantly influence the behaviour of the system for logged-in users.

SysConfig keys	Description
TokenMaxTime	Defines the maximum duration of a login in number of minutes. After the number of minutes defined here, the access token is set to invalid. The user must then log in/authenticate again.
TokenMaxIdleTime	Defines the maximum duration of an idle, inactive login in number of minutes. After the number of minutes defined here without user activity, the access token is set to invalid. The user must then log in/ authenticate again.
TokenCheckRemoteIP	Disables/enables the fixed assignment between access token and IP address. Deactivate this setting if correctly authenticated users "suddenly" no longer have access and a new login must be made. Common causes are that KIX is used behind proxy servers or changing IP addresses are assigned.

The system periodically (1/15min) checks for expired or invalid login tokens and removes them.

- The manual check can be done with the console command
  "`Console::Command::Admin::Token::List`".
- The manual removal can be done with the console command
  "`Console::Command::Admin::Token::Remove --expired`".

15.2.7.2 Authentication/Authorisation and Connection to Active Directory

- **KIX Start:** The setup is done in the menu *System > SysConfig* via the configuration key: "*Authentication###000-Default*" (see also Authentication/Authorisation and Connection Active Directory).

- **KIX Pro:** You can set up a job to fill KIX Pro with data from the LDAP/Active Directory (see also: LDAP/AD Synchronisation).

16 Reporting

The "Reports" module enables the creation of individual reports for the evaluation of key figures. You can also create individual report definitions based on SQL Select queries. The "Report Manager" role is required for this.

Content on this page:

Agents (with the "Report User" role) can then select one of these report definitions and, based on its configuration, generate and download the report in the desired output format. The report definition defines which output formats are available.
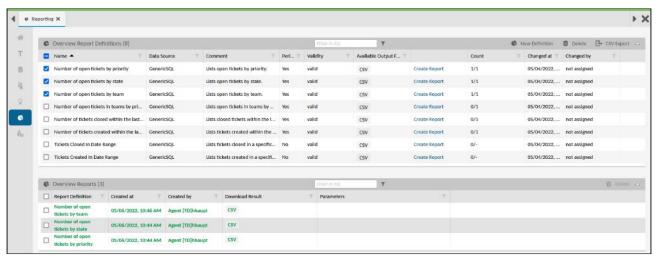
Fig .: The dashboard in the Reports module

16.1 Roles and authorizations for reporting

To work with reporting, a user needs either the role of "Report User" or "Report Manager". Both roles are extensions of user rights and can only be used in combination with other roles. Thus, for example, a user with the role of "Ticket Agent" can only call up reports if he is also given the role of "Report User".

Report Manager:

- Can create, edit and delete report definitions.
- Can use valid report definitions to create and download reports.

- **Important**: Requires at least the role "System Admin" in order to be able to assign roles in the report definition.

Report user:

- In reporting, sees all valid report definitions that can be viewed according to its authorizations.
- Can use valid report definitions within the scope of his user rights to create and download reports.
- Can specify report parameters that can be selected via the GUI before calling up a report in order to make the report more specific.
- **Important!:** Must also have the role specified in the report definition in order to be able to create reports (e.g. "Ticket Agent").

> ⬤ **Attention!**
>
> KIX generates the reports **without** a separate **check of the authorisations**. **All** objects are evaluated. "Report managers" are therefore responsible for ensuring that report data can only be viewed by authorised persons!

This means, among other things:

- Reports with predefined default values can pry out the object authorisations of the report creator:

  Example: The report creator is the owner of the role "Ticket Agent", but has no access to assets. In the report definition, the role "Ticket Agent" is specified and the number of assets is determined via the SQL statement; the entry of further parameters by the report creator is not required. Since the SQL statement is executed without a permission check, the report creator still gets an insight into the number of assets because he is the owner of the role "Ticket Agent" and thus has access to the report.

- It must be ensured that the report creator has the permissions to the objects expected by the report.

  Example: The Report Manager configures a report where the report creator can, for example, select one or more organisations as report parameters in the form. Then it must be ensured that the report creator also has the permissions to these areas (here organisations). Otherwise, the report creation will fail.

> ⬤ **Attention!**
>
> Access to reports is controlled exclusively via the assignment of authorisation roles in the report definition. For targeted access to selected reports, administrators may need to create additional, specially configured roles and assign them to the respective users (menu "User Administration > Roles/Permissions (see page 632) ").

This means, among other things:

- If, for example, a report definition is created with the roles "Report User" + "Ticket Agent", **all** ticket agents have access to these reports.
  The granularity can be increased by creating additional roles in which access to selected reports is

regulated.

- In principle, all report definitions with output format CSV are available for selection for the statistics in the Home Dashboard (KIX Pro). Availability is only limited by the roles specified in the report definition. Thus, **all** users who has the role(s) specified in the report definition can select the report for the statistics.
  By creating further roles in which access is specifically regulated, e.g. to selected organisations, the availability in the chart widget can be limited.

16.2 Reporting dashboard

The reporting dashboard contains 2 tables:

1. Overview of the report definitions
2. Overview of reports

16.2.1 Overview of the report definitions

The overview lists the report definitions created in the system. A user only sees the valid report definitions according to their authorizations. In addition to general information, the overview also includes the option of creating the report (click on "Create report").
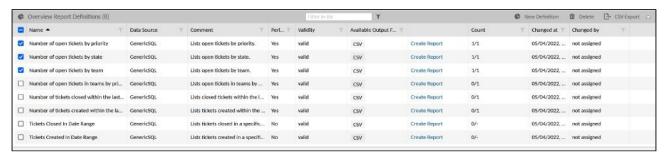
Fig .: Report definitions table

Column (extract)	Description
Name	Name of the report definition Click the name or row to edit the report definition.
Periodic	Indicates whether the report is generated periodically. Periodic reports are generated by the initial job "Periodic Reports" (see page 170) every 15 minutes.

Column (extract)	Description
Available Output Formats	Display of possible data formats in which the report can be created. The possible data formats can be specified and configured in the report definition. The following data formats are possible: · KIX Start: CSV · KIX Pro: CSV, Excel, HTML, JSON, PDF, XML
Create Report	Click the link to generate the report based on its configuration. A dialog opens in which the report parameters can be specified. The report definition defines which report parameters are available in the dialog. The generated report is then available for download in the "Overview Reports" (table below).
Count	Specification X/Y: Number of generated reports (X) of number of reports to be maintained (Y). The number of reports to be held is specified in the report definition (see page 710) .

16.2.1.1   Initial report definitions

> ⓘ **Info**
>
> It must be ensured that the report creator has at least read permissions on the parameters (objects) to be specified in the report. This means: In order to create the report "Asset Tickets Closed in Date Range", for example, the report creator needs at least read permissions on tickets, organisations and assets.

KIX Start comes with pre-configured report definitions for standard reports. If necessary, you can reconfigure these and use them as a guide for creating your own report definitions.

Report name	Description	Notes
Tickets Closed in Date Range	List of the closed tickets of selected organizations within a certain period of time, sorted by name / description of the ticket type and the ticket number. The period and the organizations can be selected in the dialog when creating the report.	
Tickets Created in Date Range	List of tickets created by selected organizations within a certain period of time, sorted by name / designation of the ticket type and the ticket number. The period and the organizations can be selected in the dialog when creating the report	
Number of open tickets by priority	Number of open tickets by priority	The reports form the basis for the statistics in the Home Dashboard and are generated automatically by means of the "Periodic Reports" job. (see page 170) **Important:** Do not change the output format, as the statistics only evaluate reports in CSV or JSON format.
Number of open tickets by state	Number of open tickets by state	
Number of open tickets by team	Number of open tickets by team	
Number of open tickets in teams by priority	Number of open tickets in teams by priority	
Number of tickets closed within the last 7 days	Number of tickets closed within the last 7 days You can change the interval in the SQL statement if necessary.	

Report name	Description	Notes
Number of tickets created within the last 7 days	Number of tickets created within the last 7 days You can change the interval in the SQL statement if necessary.	

Additional pre-configured report definitions are delivered with KIX Pro:

Report Name	Description
Assettickets Closed in Date Range	List of tickets with assigned assets that have been closed within a certain period of time. The report creator can select time period, asset classes, organizations, and ticket types.
Assettickets Created In Date Range	List of tickets with assigned assets that were created in a certain period of time. The report creator can select time period, asset classes, organizations, and ticket types.
Number Of Tickets Created Per Type And Organization	Overview of the number of tickets created in a specific month by type and organization. The report creator can select time period, organizations and ticket types.
Open Assettickets	List of open tickets with affected assets. The reporter can select ticket types, asset classes, and organizations.

16.2.2 Overview reports

The overview lists the reports available in the system. The reports of the report definitions selected with a check mark are displayed. Deleted reports no longer exist and are not listed.

	Report Definition	Created at	Created by	Download Result	Parameters	
☐	Number of open tickets by team	05/06/2022, 10:46 AM	Agent [TD]hhaupt	CSV		
☐	Number of open tickets by state	05/06/2022, 10:44 AM	Agent [TD]hhaupt	CSV		
☐	Number of open tickets by priority	05/06/2022, 10:44 AM	Agent [TD]hhaupt	CSV		
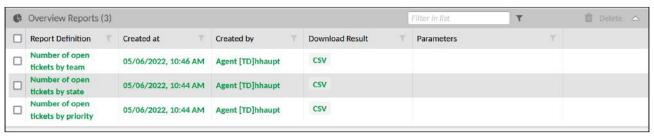
Fig .: Overview of the reports created

Column	Description
Report Definition	Name of the report definition on the basis of which the report was generated.
Created at	Date of creation of the report
Created by	User who created the report
Download Result	Output formats in which the report was created. Click on an output format to download the report in that format. If necessary, read the KIX Start user manual.

16.2.3 Deleting reports and report definitions

"Report Managers" can delete reports and report definitions that are no longer required. To do this, select the reports or report definition to be deleted with a tick and click on "Delete" in the respective table header. If a report definition is deleted, the associated reports are also deleted. Deleted reports and report definitions cannot be restored.

16.2.4 Importing reports into external systems (KIX Pro)

KIX Pro allows you to import reports into external systems (e.g. MS Excel) without additional user authentication. The import into the third-party system takes place via URL, which is provided by KIX. A URL is created for each report definition and output format, consisting of:

- the frontend FQDN stored in the system (see Setup Assistant (see page 58))
- the token stored in the personal user token (see Creating and Editing a User and Assigning Roles (see page 664))
- and the token of the report definition.

Click on the info icon in the desired output format to retrieve the URL. It is displayed in an opening overlay.

Copy the URL to the clipboard and paste it at the appropriate place in the external system:

- MS Excel: Menu File > Retrieve and transform data > From the web.
- HTML: Paste the URL into the address bar of your browser and press ENTER
- PDF: Menu File > Create > PDF from Web Site

The third-party system retrieves the data from the specified URL and displays the report. The last created report of the selected output format is displayed.

16.3   Configure a report

Before a report can be created, a report definition must be created for it. A report definition is a concrete configuration of a report to be created. This defines the data source, evaluation criteria and input parameters and defines the possible output formats. The possible parameters depend on the data source.

Report definitions are based on SQL Select queries and can therefore be configured freely and as required. Regardless of the subsequent application of the report definition, you can, for example:

- Specify how the future report will be structured and what information should be displayed in the report
- Guide the creator of the report by specifying which information is required, which information is mandatory and which data is set in the background
- Use any designation and parameters that are independent of the data source
- and much more

For each report definition, you can specify which group of people can create reports and in which output formats the reports can be created.

How to create a report definition:

1. Open the "Reports" module.
2. Click on "New definition" in the header of the table "Overview report definitions".An assistant opens in which you can configure the report definition step-by-step.
3. Configure the report definition as required (see sample configuration). To go to the next step or to switch between the individual steps, please click on the small blue arrow buttons or on the blue dots in between.
4. Finally click on "Save" to save the report definition. The report definition is now listed in the "Report definitions overview". With a click on "Create report" reports based on this configuration can now be generated at any time. You can also read about this in the KIX Start user manual.

16.3.1   Example of a report definition

The following example creates a report definition whose reports show the number of tickets of the selected type and the selected organization(s).
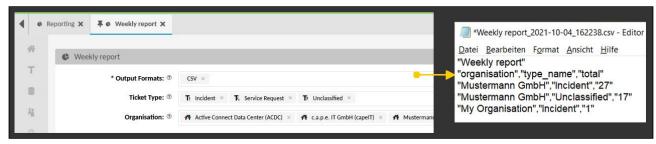
Fig .: Application of the example definition

16.3.1.1 Step 1 - Report Information

In this step, you store general information about the report definition and determine which roles are extended to include authorization to create reports.
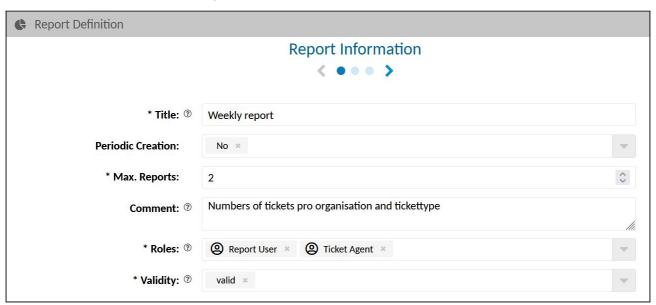
Fig.: Store report information

Field	Description
Title	Name / designation of the report definition. Enter a descriptive name so that agents can find the report they want.
Periodic Creation	You can specify whether a report is to be generated periodically. • **Yes:** The report is generated periodically. KIX is delivered with the job "Periodic Reports", which initially determines the reports to be generated periodically every 15 minutes and generates them in the output format CSV. You can change the execution times in the job. In KIX Pro you can also change the output formats. • **No:** The report is not generated periodically. It can be generated either manually or via a job configured by you.
Max. Reports	Maximum number of reports to be kept available. Excess reports previously created are automatically removed.
Comment	For example, you can enter detailed information about the report here.

Field	Description
Roles	The users of the roles selected here can use this report. The authorizations of the selected roles are extended to include the creation and reading of this report. The "Report User" role is initially pre-assigned. Users who are allowed to create reports must at least have the role of "Report User". The assignment of roles requires administrative permissions. The "System Admin" role is required for this. ⚠ **Important** - KIX generates the reports without checking the authorizations separately. All objects are evaluated. Therefore, pay attention to the authorizations within the roles! Report managers are responsible for ensuring that report data can only be viewed by authorized persons. - Select the roles only to increase their authorizations for creating reports. **Never** assign authorization to roles that have more than read rights to reports (e.g. roles "Super User" and "System Admin")! Otherwise the report definition can no longer be edited.
Validity	• **valid**: Reports can be created based on this report definition • **(temporarily) invalid**: No reports can be created on the basis of this report definition.

16.3.1.2   Step 2 - Datasource

In this step you determine the data source and define the SQL query including any parameters.

"Datasource" section

Create the SQL query here to determine which information you want to receive from which data source and which parameters should be entered by the report creator. Only SQL Select queries are possible. You can freely define individual parameters.

> ✅ **Hint**
>
> - Information on creating an SQL query can be found in relevant literature and on the Internet (e.g. https://www.w3schools.com/sql/default.asp).
> - You can find information on the structure of the KIX database
>   - in the SVG graphics (select option "fit to page")
>   - on GitHub: https://github.com/kix-service-software/kix-backend/blob/master/doc/database/KIX.svg
> - Users of on-premises installations can also generate the SQL database schema using various tools such as e.g.
>   - DBSchema: https://dbschema.com/?AFFILIATE=144826&__c=1
>   - SchemaSpy: https://dev.to/mostalive/how-to-visualize-a-postgresql-schema-as-svg-with-schemaspy-516g (Info on this: https://dev.to/mostalive/how-to-visualize-a-postgresql-schema-as-svg-with-schemaspy-516g)

Field	Description	Hint
Data source	Specification of the data source of the report definition	Currently only "GenericSQL" is possible.
DBMS	Specification of the database management system Only "Any" is possible for new reports created via the GUI.	This information is used for the standard reports supplied.

Field	Description	Hint

SQL	SQL statement
	
	Enter your SQL query here. Only select statements are executed. The code is checked before it is saved; all attributes sent to the API are validated.
	
	You can use SQL aliases as temporary column identifiers in the result table. You can refer to these aliases in step 3 (Output Format > Columns) to define the table columns to be displayed.
	
	You can create multiple arrays (e.g., array of ticket types, array of organizations, and so on).
	
	You can specify any number of freely definable parameters in the SQL statement:
	
	• Parameters are defined by the keyword "parameters". Use the following syntax: `${Parameters.<Parameter identifier>}`.
	• The name of the parameter can be freely selected.
	• Parameters form the input fields in the report.
	• The parameters used must be configured in the "Parameters" section. Add a configuration block for each parameter used.
	• The parameters used are quoted in the GenericSQL Datasource backend so that an SQL injection is not possible.
		SQL statement of the example
		
		``
		select o.name as organisation,
		tt.name as type_name, count(*) as
		total
		  from organisation o, ticket_type
		tt, ticket t
		  where t.type_id = tt.id
		    and t.organisation_id = o.id
		    and tt.id in ($
		{Parameters.TypeIDList})
		    and o.id in ($
		{Parameters.OrganisationIDList})
		  group by o.name, tt.name
		  order by o.name, tt.name
		``
		
		Aliases used in the example (keyword: "as"):
		
		• organization
		• type_name
		• total
		
		Parameters used in the example (keyword: "in"):
		
		• TypeIDList: `${Parameters.TypeIDList}`
		• OrganizationIDList: `${Parameters.OrganisationIDList}`
		
		Both parameters must be configured in the "Parameters" section.
		
		Tip: The keyword "in" enables multiple values to be specified.

Fiel d	Description	Hint
	You can give a parameter a default value as a fallback. This value is used instead of the variable if no value is available. To do this, add the following to the parameter:" `?` `<DefaultValue>` ", for example: " `$` `{Parameters.StartDate?2020-01-01` `00:00:00}` ". The default value is considered to be what is between the question mark and the closing curly bracket.	
Out put Han dler	Transformation option for the output of the reports (e.g. output as a pivot chart)	Output handlers cannot currently be configured. The output handlers are required for the execution of the standard reports provided by KIX Pro. The parameters of the output handler are directly dependent on the column names of the SQL statement. That means: Changes to the SQL statement of the standard reports lead to malfunction of the output handler.

"Parameters" section

The "Parameters" section is located below the "Output Handler" section. Click the plus button on the right to open the input fields for the parameters.

The input and selection fields that are available to the report creator in the dialog are defined with parameters. You can define any number of parameters in the SQL statement. Use the syntax: `$` `{Parameters.<Parameter identifier>}` .

Every parameter defined in the SQL statement must be specified in the "Parameters" section. This allows you to control the behavior and properties of the input and selection fields. A separate configuration block must be created for each parameter. Add configuration blocks by clicking on the plus icon

In the following, the parameters "TypeIDList" and "OrganizationIDList" specified above in the SQL statement are configured for the example.
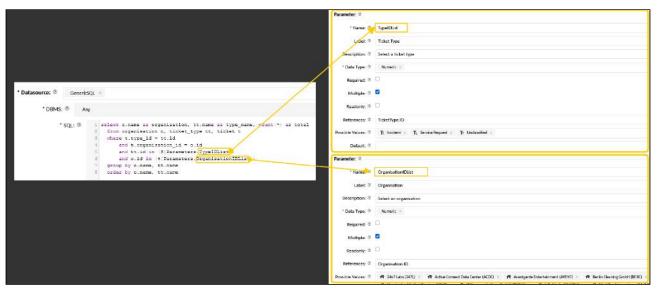
Fig .: Configuration of the parameters used in the SQL statement

Value	Description	Hint
Name	Internal identifier of the parameter (reference of the parameter from the SQL statement) (in the example: "TypeIDList" or "OrganizationIDList")	
Label	Labeling in front of the input or selection field; indicates the expected field content (in the example: "Ticket Type" or "Organization")	
Description	Brief description of the parameter. The description is displayed as a note text when the report creator ⑦ clicks.	

Value	Description	Hint
Data type	Selection of the data type The selected data type determines which data type the field will accept (in the example: each "numeric"). • **String**: characters and strings • **Numeric:** integers (e.g. for IDs) • **Date:** date • **Date Time:** Date and time • **Time:** time specification	
Required field	Mandatory field configuration **If activated:** The field is mandatory and requires input from the report creator. **If not activated:** The field can optionally be filled with values by the report creator.	
Multiple	Multiple selection in selection fields **When enabled:** The report creator can enter multiple values. **If not checked**: The report creator can only enter one value.	
Only read	The field cannot be edited by the report creator and is only displayed in the dialog for information.	

Value	Description	Hint
References	Object reference of the parameter **Syntax:** Object.Attribute, e.g. for reference to: • Team*: Queue.QueueID • User*: User.UserID • Ticket type: TicketType.ID • Organisation: Organisation.ID • Dynamic Field: DynamicField.DFName • etc. *** Syntax deviation:** When referring to team or user, the attribute is ".QueueID" or ".UserID".	 *Fig: Data source TicketType.ID and values for the ticket ID in the example* *Fig: Data source Organization.ID and values for the ID of the organizations in the example*
Possible values	Defines the values that can be selected in the report. The possible values are based on the data source selected under "References". If you refer, for example, to the ticket types (TicketType.ID), you can specify here which of the ticket types are available for the report creator to select. The same for the organizations (Organization.ID). If nothing is specified here, all objects can be selected in the report. For data type `string` : If no data source was specified under "References", possible selection values can be freely defined here, from which the report creator can choose. The selection values do not have to be part of the database.	

Value	Description	Hint
Default	You can optionally specify with which of the values specified under "Possible Values" the field should be preset (in the example: "Incident").	

16.3.1.3   Step 3 - Output Format

In this step you specify

- which table columns are displayed in the report
- in which data formats the report can be created and
- define the associated parameters for each data format.

The report author can select one or more of these data formats in the "Output format" field and thus determine the data formats in which the report is created.

If no output formats are defined here, the report can be created in all output formats with the respective default values.

Possible output formats are initial:

- for KIX Start: CSV
- for KIX Pro: CSV, Excel, HTML, JSON, PDF, XM
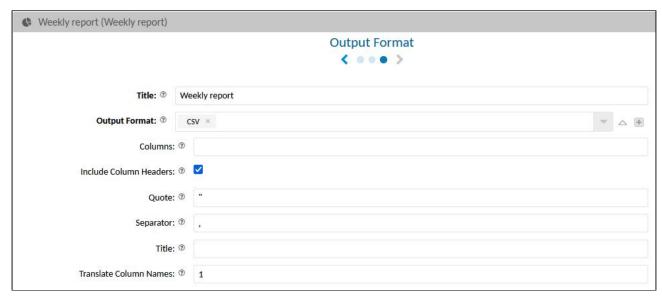
Fig .: Configuration of the output format

Field	Description	Hint
Title	Generally applicable title / heading of the generated report • Is also set as a \<title\> tag in the HTML output (KIX Pro). • Can be overwritten by the title specified in the respective output format (see below) • The parameters can be used in the title as in the SQL statement • Example: Report from $ {Parameters.startDateTime} to $ {Parameters.stopDateTime}	

Field	Description	Hint

Output format	Select the output formats in which the report can be generated.
	
	You can only select and configure 1 output format at a time. You can add additional output formats by clicking on [KIX 18 Administration> .Configure a report vKIX18_AD_V18> SF-plus.png].
	
	Depending on the selected output format, additional input fields are displayed so that format-specific parameters can be specified for each output format:
	
	• for non-technical formats (CSV, Excel, HTML, PDF)
	  • Format-specific parameters can be specified
	  • Restriction of the table columns to be displayed is possible
	  • Change of title possible
	• for technical formats (JSON, XML)
	  • No restriction of the table columns to be displayed is possible; all data is output which the SQL statement delivers.
	  • Change of title possible.
	
	Note: The statistics in the Home Dashboard only evaluate the output formats CSV and JSON.
		
		ℹ️ Only CSV can be selected in KIX Start.

Field	Description		Hint
	Columns	Optional specification of the columns to be displayed in the report. You can use it to limit the columns to be displayed in the report, for example if the SQL statement returns more result columns than required in the report. To do this, enter the relevant aliases from the SQL statement here, provided you are using aliases (in the example: `organization, ticket_type, total`). Separate multiple values with commas. This information forms the column headings. The order of the columns in the report is the order given here. If nothing is entered here, all results are taken from the SQL statement and their result values are output in the order in which they come from the SQL.	
	Include Column Headers	If selected, the column titles are also output in the report, provided the output format supports this. If nothing is specified under "Columns", no column titles will be displayed, even if this option is activated.	

Field	Description		Hint
	Quote	Character in which the individual values are enclosed (for CSV) Standard: double quotation marks (") Other z. E.g .: single quotation marks (')	For example: "John Smith Ltd.", "Incident", "27" 'John Smith Ltd.', 'Incident', '27'
	Separator	Separator between the individual values (for CSV) Default: comma (,) Other z. E.g .: semicolon (;) or dash (|)	For example: "John Smith Ltd.", "Incident", "27" "John Smith Ltd."; "Incident"; "27" "John Smith Ltd." | "Incident" | "27"
	Title	You can specify a different title for each output format. The title stored here overwrites the title of the standard configuration (1st input field on this dialog page)	

Field	Description	Hint
	Translate Column Names	For non-technical output formats, you can specify the language in which the column titles are output. KIX replaces the corresponding patterns when creating the report. **Requirement:** The language must be implemented in the SysConfig key "*DefaultUsedLanguages*" and the patterns must be stored in the "*Internationalization > Translations*" menu. **possible values:** 1 | 0 | Language abbreviations for the languages stored in the system **1**: Column title is translated into the standard language of the system (default). The standard language is set in the SysConfig key "DefaultLanguage". **0**: No translation of the column title. The default language stored in the SysConfig key "*DefaultLanguage*" is used. **Language abbreviation**: column title is translated into the selected language (e.g. de, en)	

16.4 Generate reports automatically

You can create jobs with which reports are generated automatically. For example, to be informed about the number of open fault tickets per organization every Monday at 6:00 a.m. The following example is intended to illustrate how such a job could be configured.

Information on the macro actions used in the job can be found in the overview of the actions (see page 804) in the appendix.

The reports created with a job are listed in the "Reports overview" in the "Reports" module - similar to the manually created reports.

16.4.1 Example of an SQL statement for the report definition

The job requires an existing report definition on the basis of which it can create the report. The statement of the report definition could read as follows:

```
select o.name as "organisation", tt.name as "type", ts.name as "state", count(*) as
"total"
    from organisation o, ticket_type tt, ticket_state ts, ticket t
        where t.type_id = tt.id
        and t.ticket_state_id = ts.id
        and t.organisation_id = o.id
        and tt.id in (${Parameters.TypeIDList})
        and o.id in (${Parameters.OrganisationIDList})
        and ts.id in (${Parameters.StateIDList})
    group by o.name, tt.name, ts.name
    order by o.name, tt.name, ts.name
```

16.4.2 Example configuration of the job

- Job information:
    - Job type: reporting
    - Name: Name of the job (e.g. weekly report)
    - Validity: valid
- Execution plan:
    - Weekday (s): Monday
    - Time: 6:00 a.m.
    - Events: none
- Actions (in required order)
    1. Action: "Create report" - Creates the report based on its report definition
    2. Action: "Assemble object" - Creates the attachment object (here: article attachment)
    3. Execute Macro Action - Executes a macro. (here: create a new ticket).
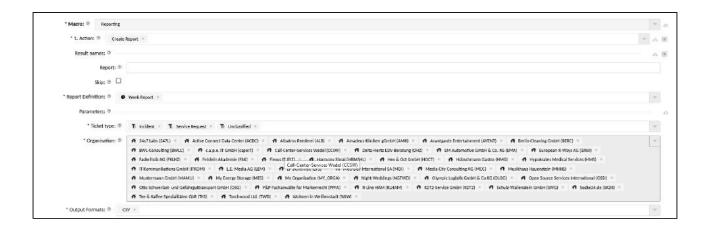
16.4.3   Configuration of the actions

16.4.3.1   1. "Create report" action

The "Create report" action creates the report based on the selected report definition. The parameters specified in the report definition are included. Enter the appropriate parameter values so that the job can set these values when generating the report. You can only specify output formats that have been specified in the report definition.

16.4.3.2   2. "Assemble object" action

This action is used to "assemble" new objects, e.g. JSON objects. In the example, the report created with "Create report" is created as a new attachment object (article attachment).

The new attachment object is saved in the `${varReportAttachment}` variable declared under "Object".

The string specified under "Definition" creates the attachment object. It has 3 properties:

- `"Filename":"dummy.csv"` → Name and file extension of the created attachment object (KIX Pro users can also choose other output formats such as HTML, XSLX etc.)
- `"ContentType":"${Report.Results:0.ContentType}"` → MIME type of the content (here: the MIME type of the report). Information on this also under: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types
  or under: https://wiki.selfhtml.org/wiki/MIME-Type/Übersicht (German Website)
- `"Content":"${Report.Results:0.Content|JSON}"` → content of the attachment object with specification of the pipe (here: content of "dummy.csv"). The specification of the pipe with JSON is necessary so that the content is validly inserted and quoted in JSON format.
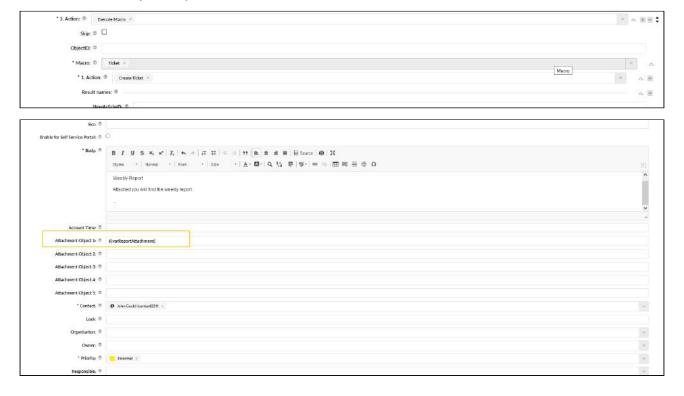
16.4.3.3 3. Action "Execute Macro"

This action executes further macros. In the example, a new ticket is created with the following information:

- the required ticket information such as status, priority, subject, channel, etc.
- by specifying the variable `$ {varReportAttachment}` the object (dummy.csv) created with "assemble object" is given to the ticket as an article attachment.
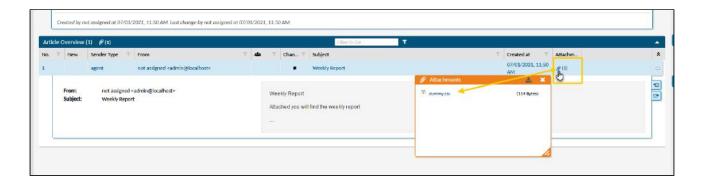
16.4.3.4 Result

A new ticket exists after the job has been executed. The ticket contains the report in the message body and the CSV file of the report in the article attachment.

> ⓘ **Hint**
>
> - The macro actions are only validated when the job is executed. You can find information on possible errors in the kix.log (System> Logs menu) or in the job.log ("History" tab in the job details).
> - The automated jobs are executed with the system user and use their language setting. The default language of the system (SysConfig key "*DefaultLanguage*") serves as a fallback.

17 Interfaces

In the following chapters you will find information on connecting external systems to KIX 18.

- KIX Start REST API (see page 731)
- Data transfer (see page 734)
- System Monitoring (see page 736)

Note: The connection to an LDAP/AD server can be found in the User administration> User (see page 661) section.

17.1 KIX Start REST API

The REST API can be used for client-server communication with the KIX backend (core) application. It does not claim to be complete and can be expanded with extensions.

The documentation for the REST API from KIX Start can be found on GitHub at:

https://raw.githubusercontent.com/kix-service-software/kix-backend/master/doc/API/V1/KIX.html

Note: For correct display, please save the link as an HTML document and then open the document in your browser.

17.1.1 File size limitation

The maximum file size for <u>all</u> requests via API to the backend is 50 MB. This is specified in the core of KIX 18. Thus, a ticket including all articles and attachments or a file import must not exceed this file size. This applies to both the requests from the agent portal and the self-service portal.

However, it may be necessary to remove this restriction (temporarily) or to reduce it further. This is done with the following configuration keys in the *System> SysConfig menu*.

> ⚠ **Caution!**
> Avoid overloading the system and your infrastructure!

Configuration key	Description
API::Provider::Transport::MaxLength	Defines the maximum file size that can be sent to the backend via the API (default: 50 MB). This affects the entire request, i.e. the ticket along with the article and attachments or the complete file import. The value can be larger or smaller than 50 MB. The value defined here does not overwrite the value of 50 MB set in the core, but it has priority. The value of 50 MB set in the core serves as a fallback.
FileUpload::MaxAllowedSize	Specifies the maximum file size per attachment. The sum of all attachments together with the ticket and articles must not be greater than the value set under "API :: Provider :: Transport :: MaxLength".

17.1.2 Limitation of requests

Configuration key:	API::Request::DefaultLimit

The number of frontend requests is initially limited to 1000 in order not to endanger the performance of the system. The request considers all resources, not only "/tickets". You can change the limit in the SysConfig key "*API::Request::DefaultLimit*" if necessary (menu "*System > SysConfig*").

Please note: If the limit in the SysConfig key is set to 0 and no limit is passed on the request, unlimited results will be returned. This can have a negative effect on performance.

17.2 Data transfer

At https://github.com/kix-service-software/kix18sync we provide you with simple scripts for importing and exporting data. The scripts act as **external tools** that only access KIX via the REST API. Data sources can be remote DB tables or CSV files.

In GitHub you will find the following directories:

- `bin:` scripts for data synchronisation
- `config:` Standard/sample configuration
- `sample:` Sample import data (CSV)

The following scripts are available:

Script	Description
kix18.ConfigImportExport.pl	Transfer of complete object configurations (jobs, dynamic fields, report definitions, actions and templates)
kix18.ManageRoles.pl	Transfer of role and authorisation information
kix18.CSVSync.pl	Client for data import from CSV files (contact, organisation, SLA)
kix18.DBSync.pl	Client for importing data from a remote DB (contacts, organisations)
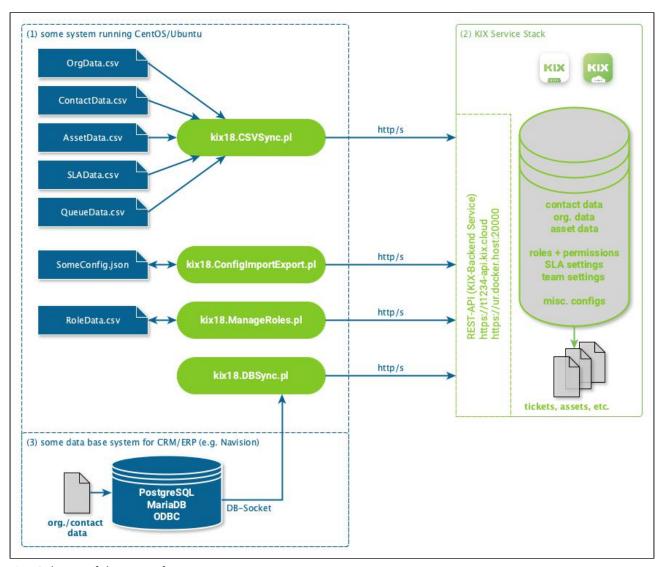
Fig.: Scheme of data transfer

17.3 System Monitoring

KIX supports the automatic assignment and processing of event messages by mail from systems such as Nagios, Icinga or CheckMK. The preconfiguration provided is based on the Nagios email structure (see example below). If your system monitoring tool uses a different message structure, the configuration must be adjusted.

Content on this page:

Possible are therefore:

1. Automatic detection of critical states (system monitoring state: DOWN / CRITICAL) and creation of a ticket
2. Automatic detection of follow-up messages for an already recorded event / ticket
3. Automatic closing of the ticket when a standard state is reached (system monitoring state: UP / OK)
4. Automatic detection of the service or host monitored by system monitoring and entry in the "Affected Assets" field (see below)

17.3.1 Setup

The following describes how to set up a system monitoring configuration using "*PostMaster :: PreFilterModule ### 0001-SystemMonitoring*" as an example. If necessary, you can set up several system monitoring configurations. To do this, use the other SysConfig keys with the name pattern "*PostMaster :: PreFilterModule ### * - SystemMonitoring*".

17.3.1.1 Parameter

Subkey	Description and notes	Example / default value
Module	⚠️ Do not change - Defines backend code modules that are used for mail processing.	Kernel::System::Post Master::Filter::System MonitoringX

Subkey	Description and notes	Example / default value
DynamicFieldContent::Ticket	⚠ Do not change - Defines the dynamic fields used by the mechanism that are assigned to tickets.	SysMonXHost, SysMonXService, SysMonXAddress, SysMonXAlias, SysMonXState
DynamicFieldContent::Article	⚠ Do not change - Defines the dynamic fields used by rhe mechanism taht are assigned to tickets.	
AffectedAssetName	Defines the dynamic field that is used to store the affected asset. The field must be of the type "AssetReference". The asset is determined on the basis of the search pattern extracted by "SysMonXHostRegExp" in an asset name search. Only one asset can be entered. If several assets match the search pattern, the first asset will be recorded.	AffectedAsset
SysMonXStateName	⚠ Do not change - Defines the dynamic field in which the system monitoring status is recorded.	SysMonXState
CreateTicketType	Defines the ticket type that is used to create the ticket. The non-localized name must be used.	Incident
CreateTicketState	Defines the ticket status that is set when the ticket is created. The non-localized name must be used.	new
CreateSenderType	⚠ Do not change - Defines the sender type of the article that is created for the ticket.	system
CreateChannel	⚠ Do not change - Defines the communication channel of the article, which is created for the ticket.	note

Subkey	Description and notes	Example / default value
CreateTicketQueue	Defines the team to which the ticket is initially assigned. The non-localised name must be used. The full name must be used. The team specified here is the default value. When creating/editing an email filter (menu *Communication > Email > Email Filter*), another team can be set. The team set in the email filter has priority over the SysConfig setting. 	Service Desk::Monitoring
CreateTicketSLA	Defines the SLA, which is initially assigned to the ticket.	
CloseNotIfLocked	Prevents the status from being set according to "CloseActionState" if the existing ticket is blocked.	0
StopAfterMatch	If set ("1"), no further email filters are used in the processing of the mail.	1
FromAddressRegExp	This system monitoring configuration only processes emails whose sender matches this pattern.	sysmon@example.com
ToAddressRegExp	This system monitoring configuration only processes emails whose To recipients match this pattern.	.*

Subkey	Description and notes	Example / default value
SysMonXAddressRegExp	Defines the search pattern with which the address of the affected system is extracted from the mail. The first capture group is used.	\s*Address:\s+(.*)\s*
SysMonXAliasRegExp	Defines the search pattern with which an alias of the affected system is extracted from the mail. The first capture group is used.	\s*Alias:\s+(.*)\s*
SysMonXStateRegExp	Defines the search pattern with which the system monitoring status is extracted from the mail. The first capture group is used.	\s*State:\s+(\S+)
SysMonXHostRegExp	Defines the search pattern with which the host name of the affected system is extracted from the mail. The first capture group is used.	\s*Host:\s+(.*)\s*
SysMonXServiceRegExp	Defines the search pattern with which the affected service is extracted from the mail. The first capture group is used.	\s*Service:\s+(.*)\s*
DefaultService	If no service is found in the system monitoring message, this value is set.	Host
NewTicketRegExp	If the system monitoring status matches this pattern, a new ticket will be created if there is no open ticket.	CRITICAL | DOWN | WARNING
CloseTicketRegExp	If the system monitoring status corresponds to this pattern, an existing ticket is given a new ticket status according to "CloseActionState". If no open ticket can be found for this service or host, the email is discarded and ignored.	OK | UP
CloseActionState	Defines the status that is set on the ticket when a system monitoring status corresponding to "CloseTicketRegExp" is received.	closed

Subkey	Description and notes	Example / default value
ClosePendingTime	If a waiting status is set in "CloseActionState" (e.g. "pending auto close"), this waiting time is set in seconds at the time the message is received.	172800

17.3.1.2 Configuration example

Configuration key: PostMaster::PreFilterModule###0001-SystemMonitoring

```
{
    "AffectedAssetName": "AffectedAsset",
    "CloseActionState": "closed",
    "CloseNotIfLocked": "0",
    "ClosePendingTime": "172800",
    "CloseTicketRegExp": "OK|UP",
    "CreateChannel": "note",
    "CreateSenderType": "system",
    "CreateTicketQueue": "Service Desk::Monitoring",
    "CreateTicketSLA": null,
    "CreateTicketState": "new",
    "CreateTicketType": "Incident",
    "DefaultService": "Host",
    "DynamicFieldContent::Article": null,
    "DynamicFieldContent::Ticket":
"SysMonXHost,SysMonXService,SysMonXAddress,SysMonXAlias,SysMonXState",
    "FromAddressRegExp": "sysmon@example.com",
    "Module": "Kernel::System::PostMaster::Filter::SystemMonitoringX",
    "NewTicketRegExp": "CRITICAL|DOWN|WARNING",
    "StopAfterMatch": "1",
    "SysMonXAddressRegExp": "\\s*Address:\\s+(.*)\\s*",
    "SysMonXAliasRegExp": "\\s*Alias:\\s+(.*)\\s*",
    "SysMonXHostRegExp": "\\s*Host:\\s+(.*)\\s*",
    "SysMonXServiceRegExp": "\\s*Service:\\s+(.*)\\s*",
    "SysMonXStateName": "SysMonXState",
    "SysMonXStateRegExp": "\\s*State:\\s+(\\S+)",
    "ToAddressRegExp": ".*"
}
```

17.3.1.3   Example email message

Example email message for DOWN message

```
 1   Return-Path: <sysmon@example.com>
 2   To: nagios-service-mailbox@example.com
 3   Subject: ** PROBLEM alert 1 - ddsrv007 host is DOWN **
 4   Date: Sun, 10 May 2021 01:00:00 +0100 (CET)
 5   From: sysmon@example.com
 6   Mime-Version: 1.0
 7   Message-Id: <20210501000000.1212121212A@xyzabc.example.com>
 8
 9   ***** Nagios  *****
10
11   Notification Type: PROBLEM
12   Host: ddsrv007
13   State: DOWN for 0d 0h 5m 0s
14   Address: 192.168.1.2
15   Info: 1st-critical report for hit in CMDB - should create ticket with aff.
     asset
16
17   CRITICAL - Time to live exceeded (192.168.1.2)
18
19   Date/Time: Mon May 10 00:00:00 CET 2021
20
21   ACK by:
22   Comment:
```

17.3.2 Automatically link assets to tickets

You can specifically and automatically link assets to tickets. Use the event handler "*TicketAutoLinkConfigItem*" for this purpose. This is a supplement to the system monitoring integration via mail filter. It enables the targeted search for assets based on asset-identifying information stored in articles (especially emails) or fields, such as IP addresses or FQDNs. The specific attributes of the asset classes are used as search criteria and searches are only carried out in matching asset classes.

The email filter for system monitoring has some limitations in this respect:

- It can only perform a simple asset search based on the asset name.
- It cannot assign multiple assets to the ticket being created.
- It cannot react to additional, different item attachments or subsequent setting of form fields.
- Moreover, the automatism is not bound to the article receipt by mail. It can therefore also react to arrivals on other channels, such as Self Service Portal or the entry in a text form field also agent users.

The configuration is done in the following SysConfig keys:

- *Ticket::EventModulePost###500-TicketAutoLinkConfigItem.*
    - Enables/disables automation and defines which events are reacted to:
        - which events are reacted to
        - which field is used to store the assets
        - whether only the first item of the ticket should be evaluated for item events (focuses on the direct addition to the email filter for system monitoring).

- *TicketAutoLinkConfigItem::CISearchPattern*
    - Determines from which ticket attributes which patterns are to be extracted. The first capture group of the regular expression to be stored as value is used for the search in the CMDB. The key is the ticket attribute, e.g. "DynamicField_SysMonXHost" or "Article_Body". If several search patterns are to be extracted from the same ticket attribute, the keys can be provided with the suffix "_ORn" (where n 1..N) - e.g. "Article_Body" and "Article_Body_OR1".

- *TicketAutoLinkConfigItem::CISearchInClassesPerRecipient*
    - This configuration is only relevant for article evaluations. It allows the restriction of the asset classes to be searched based on the to-recipient of the triggering article. The key contains an email address. The value contains a comma-separated list of the asset classes to be searched. Only the pure email address (localpart@domainpart [no real name]) is evaluated, regardless of upper/lower case.

- *TicketAutoLinkConfigItem::CISearchInClasses*
    - This configuration defines in which asset class (key), which attributes are to be searched and must match the search patterns found. Multiple attributes must be separated by commas. An attribute must contain the full name in its asset class structure, e.g. "SectionNetwork::NIC::IPAddress" for the search for IP address, in order to be able to perform a unique search.

💡 Also observe the notes (description) in the respective keys.

18 Appendix

Here you can find further information on the following topics:

18.1   Regular Expressions

You can use regular expressions to check whether entries that the user has made in form fields are valid and/or contain certain characters and character strings. Typical examples would be checking that the @ character has been used in email addresses, or specifying that only letters and numbers are permitted in a text field. In KIX, regular expressions are used in things like dynamic fields, email filters, and asset class definitions. A small selection of regular expressions is provided below. Further information about regular expressions can be found in relevant technical literature or various websites (e.g. https:// docs.microsoft.com/en-gb/dotnet/standard/base-types/regular-expression-language-quick-reference).

Example:

- If only numbers and letters (without umlauts) are permitted: ^[A-Za-z0-9]+$ (alternatively: ^[\w\s*]+$)
- If only numbers and letters (including umlauts) are permitted: ^[\w\säöüÄÖÜ*]+$

Regular expression	Description
^	Corresponds to the start of the line
+	Quantifier – preceding character/character group occurs at least 1 time
\w	Corresponds to any letter (including those with umlauts), a number, an underscore; equal to the expression `[a-zA-Z0-9_]`
\s	Corresponds to any blank or line break character; equal to the expression `[\r\n\t\f\v ]`
*	Quantifier – preceding character/character group occurs 0 times or n times
$	Corresponds to the end of the line
–	Corresponds to a range

Other regular expressions:

Regular expression	Description
`[abc]`	Corresponds to one of the characters "a", "b", or "c"
`[0-6]`	Corresponds to a number from 0 to 6
`[A-Za-z0-9]`	Corresponds to any letter from the Latin alphabet or any number.

18.2 Placeholders

KIX placeholders are variables for ticket and item-related attribute values such as the agent or contact on the ticket, or values of dynamic fields. This information varies per ticket. With KIX placeholders, you can reference the attributes on the ticket or article in order to use their variable values in the desired place.

Content on this page:

- Use of KIX placeholders (see page 747)
- Explicit suppression of localisation (see page 748)
- Restrictions for placeholders (see page 749)

KIX placeholders are integrated in the Admin module at the appropriate place and replaced by the real contents in the Agent or Self Service Portal when a function is called up, e.g. when

- execution of a job
- Use of a text module
- Use of a ticket or article action (KIX Pro)
- Use of a ticket template (KIX Pro)

Example of placeholder in a textmodule	Dear Mr. <KIX_CONTACT_Firstname> <KIX_CONTACT_Lastname>, thank you for[...]
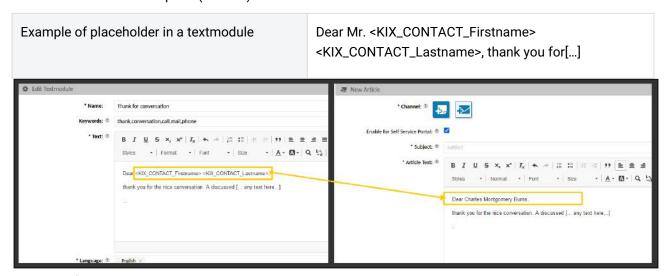
Fig.: Use of a Text Module with Placeholders in Article Text

With placeholder variables you can reference almost all objects in KIX. For example, to:

- System information (e.g. current date, processing date)
- User information (e.g. agent's name, agent's contact details)
- Ticket information (e.g. ticket title, ticket state)
- Customer information (e.g. customer's name, company)
- Article information (e.g. article subject line).

18.2.1 Use of KIX placeholders

The use of placeholders is manifold. Placeholders allow, for example:

- specify the current ticket ID or the current user in the ticket text
- to include the respective team variably in a signature
- adapt signatures, text modules, notifications, etc. to the respective context of use
- create a text module for the personal salutation
- fill dynamic fields with variable content
- send ticket notifications to the respective observer of a ticket
- apply jobs to tickets of the respective ticket owner
- and much more.

Placeholders can be used in 2 ways:

1. Placeholders that the user manually adjusts when using the text module, e.g. `DD.MM.JJJJ` for date entries.
2. Placeholders as variables that are automatically replaced by real contents at the time of their use. These variables are placed in angle brackets and included according to the scheme `<KIX_OBJEKT_Attribute>` .

The appendix contains a selection of possible OBJECTS and attributes .

Examples of placeholders:

Placeholder	Result
<KIX_TICKET_State>	Read out current ticket state (e.g. "new")
<KIX_CONTACT_Firstname>	First name of contact in ticket
<KIX_TICKET_Type>	Read out current ticket type (e.g. "open")
<KIX_TICKET_DynamicField_SomeName_ObjectValue>	Read out the first value of the dynamic field "SomeName" (e.g. in select fields)
<KIX_TICKET_DynamicField_SomeName_ObjectValue_n>	Read out the n-th value of the dynamic field "SomeName" (e.g. the 3rd value in the select field)

> ⚠ **Please note**
>
> Please note that the context, and where the ticket is in the workflow, have a decisive influence on the replacement of the placeholders. For example, it will not be possible to replace the placeholder denoting the ticket number in the signature yet if the agent is creating a new ticket via email. The ticket number and all other ticket data will not be assigned until the ticket is created, which means this information will not be available yet when the placeholders are replaced.

18.2.2 Explicit suppression of localisation

In jobs and actions, KIX placeholders are resolved in the default language of the system. This means that if German is set as the default language, the values of the placeholders are translated into German, so that invalid values can arise. The concrete setting of valid values for status, type, selection values of select fields, logins for the person responsible or editor, etc. is thus (partially) prevented.

Example: The job should set the status on the ticket using the placeholder <KIX_TICKET_State>. However, due to the localisation, the status "neu" is set instead of "new". However, the status "neu" is an invalid value and leads to an error.

To prevent the localisation of KIX placeholders, you can enter an exclamation mark (!) after the placeholder name: <KIX_TICKET_**State!**>. The exclamation mark suppresses localisation so that the value of the placeholder is set correctly.
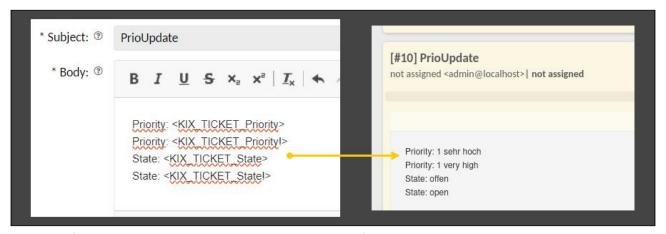
Fig.: Use of KIX placeholders with and without suppression of localisation

18.2.3 Restrictions for placeholders

To prevent agents from reading out unauthorised information, placeholders are excluded from use by the agent. If the agent enters placeholders in text fields, these are not resolved when they are saved, not even in text modules. This affects the following areas:

- Ticket creation
- Ticket editing
- Article creation
- Answering/forwarding articles
- Other actions with text fields (title, subject, article text, dynamic fields, etc.)

However, team signatures with placeholders will be displayed with resolved placeholders after saving a new ticket or a new article.

Various placeholders from the configuration (KIX_CONFIG_) are also not resolved in team signatures. These are marked in orange font in the SysConfig and have the access level "confidential".

Examples of confidential placeholders are, for example:

- <KIX_CONFIG_Ticket::Hook>
- <KIX_CONFIG_PGP::Key::Password>
- <KIX_CONFIG_DefaultUsedLanguages>
- <KIX_CONFIG_AdminEmail>
- <KIX_CONFIG_ContactSearch::UseWildcardPrefix>
- <KIX_LAST_BodyRichtext>

18.2.4 Overview of KIX Placeholders

Below you will find a list of placeholders that can be used in messages, text modules and in the system core as well as in the configurations of templates, actions and jobs.

Content on this page:

KIX placeholders are always placed in angle brackets. Syntax:
`<KIX_OBJECT_Attribute>.`

To explicitly suppress the localisation of placeholders, place an exclamation mark (!) after the placeholder name: `<KIX_TICKET_State!>.`   You can find more information on this under: Placeholders .

Universal Placeholders

Placeholder	Meaning	Example
KIX_OBJECT_LinkCount	Number of links of an object. The object corresponds to the respective context. E.g.: Number of links of an asset/ of a ticket/ of a FAQ	see brackets in the tab "Linked objects" in the detailed view of an asset/a ticket/a FAQ.

18.2.4.1 User-Related Placeholder Ranges

The placeholder ranges OWNER, RESPONSIBLE, and CURRENT reference users who are registered in KIX, and offer the following placeholders.

The placeholder ranges "KIX_RESPONSIBLE_xxx" and "KIX_OWNER_xxx" are available in existing tickets and reference the ticket owner (OWNER) and person responsible for the ticket (RESPONSIBLE) that have been entered in each case.

In contrast, the placeholder range "KIX_CURRENT_xxx" always indicates the user who is currently logged in and is executing the function concerned. In some automated processes such as jobs, this is the system user.

> ⓘ **Info**
>
> - The system distinguishes between contacts and users. It is possible that a contact is only managed as such - without the possibility of logging into the system. Such a contact then has no user assignment. Conversely, it is possible to create technical users (via REST API) that have no contact assignment.
> - Use <KIX_CURRENT_ContactID> to set the currently logged in user as contact on the ticket or in a dynamic field of the type "Contact".
> - Use <KIX_CURRENT_UserID> to set the currently logged in user as the agent/responsible person on the ticket.

Placeholder KIX_OWNER	Placeholder KIX_RESPONSIBLE	Placeholder KIX_CURRENT	Meaning	Example
KIX_OWNER_ID	KIX_RESPONSIBLE_ID	KIX_CURRENT_ID	Unique user ID for the user	123
KIX_OWNER_Firstname	KIX_RESPONSIBLE_Firstname	KIX_CURRENT_Firstname	First name of the contact assigned to the user	John
KIX_OWNER_Lastname	KIX_RESPONSIBLE_Lastname	KIX_CURRENT_Lastname	Last name of the contact assigned to the user	Doe
KIX_OWNER_Email	KIX_RESPONSIBLE_Email	KIX_CURRENT_Email	Email address of the contact assigned to the user	John.Doe@kix.cloud
KIX_OWNER_Email1	KIX_RESPONSIBLE_Email2	KIX_CURRENT_Email3	Reference to the additional email addresses (1 to 5) of the user	John.Doe@kix.cloud

Placeholder KIX_OWNER	Placeholder KIX_RESPONSIBLE	Placeholder KIX_CURRENT	Meaning	Example
KIX_OWNER_Mobile	KIX_RESPONSIBLE_Mobile	KIX_CURRENT_Mobile	Cell phone number of the contact assigned to the user	+49 123 456 789
KIX_OWNER_Phone	KIX_RESPONSIBLE_Phone	KIX_CURRENT_UserPhone	Phone number of the contact assigned to the user	+49 987 654 321
KIX_OWNER_UserID	KIX_RESPONSIBLE_UserID	KIX_CURRENT_UserID	Unique user ID for the user	123
KIX_OWNER_UserLogin	KIX_RESPONSIBLE_UserLogin	KIX_CURRENT_UserLogin	Login name of the user	jodo
KIX_OWNER_UserLanguage	KIX_RESPONSIBLE_UserLanguage	KIX_CURRENT_UserLanguage	Language setting of the user	de
KIX_OWNER_Comment	KIX_RESPONSIBLE_Comment	KIX_CURRENT_Comment	Comment on user entry	some text
KIX_OWNER_CreateTime	KIX_RESPONSIBLE_CreateTime	KIX_CURRENT_CreateTime	Time stamp indicating when the user entry was created	2020-01-01 00:00:00
KIX_OWNER_ChangeTime	KIX_RESPONSIBLE_ChangeTime	KIX_CURRENT_ChangeTime	Time stamp indicating the last change to the user entry	2020-01-01 00:00:00
KIX_OWNER_IsCustomer	KIX_RESPONSIBLE_IsCustomer	KIX_CURRENT_IsCustomer	Indicates whether the user is permitted to log in as the customer contact	0	1

Placeholder KIX_OWNER	Placeholder KIX_RESPONSIBLE	Placeholder KIX_CURRENT	Meaning	Example
KIX_OWNER_IsAgent	KIX_RESPONSIBLE_IsAgent	KIX_CURRENT_IsAgent	Indicates whether the user is permitted to log in as the agent	0 | 1
KIX_OWNER_ContactID	KIX_RESPONSIBLE_ContactID	KIX_CURRENT_ContactID	the unique contact ID of the user (if assigned)	456
KIX_OWNER_DynamicField_XYZ	KIX_RESPONSIBLE_DynamicField_XYZ	KIX_CURRENT_DynamicField_XYZ	Readable representation of the DF value (XYZ corresponds to the name of the dynamic field)	Value of the dynamic field (text)

18.2.4.2 Ticket-Related Placeholder Ranges

The placeholder range TICKET is available for tickets and represents the property of the ticket in question (as specified below). For individual fields ("Dynamic Fields"), this list must be expanded depending on the specific KIX installation concerned.

Placeholder	Meaning	Example
KIX_TICKET_TicketNumber	Ticket number	202001014200001
KIX_TICKET_Title	Ticket title /brief summary	Order to block AD account
KIX_TICKET_TicketID	Unique ID of ticket	123
KIX_TICKET_State	Name of state	open
KIX_TICKET_StateID	ID of state	2
KIX_TICKET_StateType	Name of state type; of set state	open

Placeholder	Meaning	Example
KIX_TICKET_PendingTime	Pending time if status of type "pending reminder" or "pending auto" has been set	2020-05-23 09:00:00
KIX_TICKET_Priority	Name of priority	3 normal
KIX_TICKET_PriorityID	ID of priority	3
KIX_TICKET_Lock	Designation of lock identifier	lock
KIX_TICKET_LockID	ID of lock identifier	2
KIX_TICKET_Queue	Name of team/queue	Service desk
KIX_TICKET_QueueID	ID of team/queue	1
KIX_TICKET_Organisation	Name of organization saved in ticket	ExampleCompany Corp.
KIX_TICKET_OrganisationID	ID of contact/customer user saved in ticket	123
KIX_TICKET_Contact	Name of contact/customer user saved in ticket	John Doe
KIX_TICKET_ContactID	ID of contact/customer user saved in ticket	123
KIX_TICKET_Owner	Login name of owner entered in ticket	limu
KIX_TICKET_OwnerID	ID of owner entered in ticket	123
KIX_TICKET_Type	Name of ticket type	Incident
KIX_TICKET_TypeID	ID of ticket type	2

Placeholder	Meaning	Example
KIX_TICKET_Responsible	Login name of person responsible entered on ticket	limu
KIX_TICKET_ResponsibleID	ID of person responsible ID entered on ticket	123
KIX_TICKET_Created	Time stamp indicating when ticket was created	2020-01-01 09:00:00
KIX_TICKET_Changed	Time stamp indicating the last change to the ticket	2020-04-01 09:00:00
KIX_TICKET_ID	Unique ID of ticket	12345

18.2.4.3 Placeholders for Dynamic Fields

Placeholders for predefined Dynamic Fields

Placeholder	Meaning	Example	Comment
KIX_TICKET_DynamicField_MobileProcessingChecklistXXX_Key	Checklist content depicted as JSON		

Placeholder	Meaning	Example	Comment
KIX_TICKET_DynamicField_MobileProcessingChecklistXXX_Value	Checklist content depicted as plain text	Lock Out Tag Out Checklist - Disconnect from mains: OK - Secure machine against being switched back on: NOK - Secured using: electrical tape - Determine whether de-energized: n.a. - Grounding and short circuiting: OK - Cover or shield adjacent live parts: OK	
KIX_TICKET_DynamicField_MobileProcessingChecklistXXX_Short	Extent to which checklist has been completed	2/3	
KIX_TICKET_DynamicField_MobileProcessingChecklistXXX_HTML	Checklist content depicted as HTML	*(cannot be shown here)*	
KIX_TICKET_DynamicField_RiskAssumptionRemark	Note text for transfer of risk	A long text.	Relevant for KIX Field Agent
KIX_TICKET_DynamicField_MobileProcessingState	State of mobile processing in KIX Field Agent App	assigned	Relevant für KIX Field Agent
KIX_TICKET_DynamicField_PlanBegin	Time stamp for planned start	2020-05-23 09:00:00	
KIX_TICKET_DynamicField_PlanEnd	Time stamp for planned end	2020-05-23 09:00:00	

Placeholder	Meaning	Example	Comment
KIX_TICKET_DynamicField_PlannedEffort	Planned processing effort in minutes	60	
KIX_TICKET_DynamicField_CloseCode	Close Code	0-solved | 1-solved (work around) | 2-not solved (not reproducible) | 3-not solved (too expensive) | 4-solved by caller | 5-cancelled by caller	
KIX_TICKET_DynamicField_AffectedAsset	List of affected assets	chnb001.office.company.com chnb003.intra.company.com	
KIX_TICKET_DynamicField_AcknowledgeName	System monitoring source	Nagios1	
KIX_ORG_DynamicField_Type	Organization type	customer|external supplier| internal supplier	
KIX_CONTACT_DynamicField_Source	Contact data source	"Active Directory XYZ"	

Placeholders for individual Dynamic Fields (generic)

Placeholder*	Meaning	Example	Field Types
KIX_TICKET_DynamicField_XYZ	Readable depiction of DF value (text)		• all

Placeholder*	Meaning	Example	Field Types
KIX_TICKET_DynamicField_XYZ_Key	Internal depiction of DF content (text, may contain IDs) Behaves like _ObjectValue if the placeholder is not surrounded by any other text. If surrounded by text, object-value pairs are returned with a separator. The separator that is stored in the configuration of the dynamic field is used. Please note that the "Loop" macro action requires **a comma** as a separator. ℹ We therefore recommend the general use of "_ObjectValue" instead of "_Key".		• Selection • References
KIX_TICKET_DynamicField_XYZ_Value	Readable depiction of DF value (text)	• Contact Reference: "Doe, John" <J.Doe@kix.cloud>	• all
KIX_TICKET_DynamicField_XYZ_Short	Readable depiction of DF value (text, contains brief information)	• Contact reference: "J.Doe@kix.cloud" • Checklist: "X/Y" (X points done, Y points total) • Table: "X rows" (X = number of rows)	• Contact Reference • Checklist • Table

Placeholder*	Meaning	Example	Field Types
KIX_TICKET_DynamicField_XYZ_HTML	Readable formatted depiction of DF value (HTML)	• Contact reference: "Last name, first name" <email address> as "mailto-HREF". • Checklist: as HTML table with heading and values. • Table: as HTML table with column labels and recorded values in the body.	• Contact Reference • Checklist • Table
KIX_TICKET_DynamicField_XYZ_Object_*n* (*n = any numerical value)*	Identifier to access the placeholders of the object referenced with the stored ID (at position *n*)		• currently only AssetReference
KIX_TICKET_DynamicField_XYZ_ObjectValue	Returns an array with IDs if the placeholder is not surrounded by any other text. If surrounded by text, object-value pairs are returned with a comma as the separator	• see below references to objects	• all
KIX_TICKET_DynamicField_XYZ_ObjectValue_*n* (*n = any numerical value)*	ObjectValue returns an array with IDs. The value n is returned from this array.	• see below references to objects	• all with array support

* XYZ corresponds to the individual name of the dynamic field.

Placeholder for referencing dynamic fields

Placeholders for referencing dynamic fields are replaced for:

- Asset
- Contact

- Organization
- Ticket
- FAQ.

The placeholders return a readable value (array) that contains all values of a Dynamic Field including separators. The separator specified in the configuration of the dynamic field is used.

For example, the following values can be read:

- List of values of a dynamic field of type Selection
- List of referenced objects in a dynamic field of the type:
    - TicketReference
    - AssetReference
    - Organization reference
    - Contact reference
- List of affected assets and affected services
- etc.

References to objects

The addition "_ObjectValue" or "_ObjectValue_n" can be used to address the individual values in a referencing Dynamic Field, e.g. when using the "Loop" macro action.

The structure is:

- `<KIX_OBJECT_DynamicField_DFNAME_ObjectValue[_n]>`.

Examples:

- `<KIX_TICKET_DynamicField_SomeName_ObjectValue>` - returns the value of the dynamic field "SomeName" (usually an array); if surrounded by text, the first value is returned as an object-value pair.
- `<KIX_ORG_DynamicField_SomeName_ObjectValue_0>` - returns the first value of the dynamic field "SomeName".
- `<KIX_CONTACT_DynamicField_SomeName_ObjectValue_1>` - returns the second value of the dynamic field "SomeName".
- `<KIX_ASSET_DynamicField_SomeName_ObjectValue_2>` - returns the third value of the dynamic field "SomeName".
- etc.

> ⚠ **Notes**
>
> - If you want to assign values of existing fields to dynamic fields in Macro Actions, use the placeholder `<KIX_TICKET_DynamicField_XYZ_ObjectValue>` to assign values correctly.

- The `<KIX_TICKET_DynamicField_SomeName_ObjectValue>` placeholders contain the JSON data structure for the "Checklist" and "Table" field types that make them up. This can be useful for use in automation jobs.
- In the following places, "_ObjectValue" (can be "_Key") must be used if the object value is to be used. All other placeholder variants are not suitable.
    - Macro action "Create Ticket": for the DF values
    - Macro action "Create FAQ Article": for the DF values
    - Macro action "Set Dynamic Field": for the value
    - Macro action "Loop": in the "Value" ((i.e. what is to be iterated over)
    - Macro action "Create Ticket by System Template": in the DF values (in the template)

References to object attributes

The attributes of referenced objects can be accessed. This allows you to access the individual attributes of assets, contacts, or other referenced objects.

The structure is:

- `<KIX_OBJECTTYPE_DynamicField_DFNAME_Object_0_Attribute>`.

    - `OBJECTTYPE` : Object type which allows dynamic fields and for which placeholder handling is possible (e.g. TICKET, CONTACT, ORG).
    - `DFNAME` : Name of the dynamic field. It must be a referencing Dynamic Field (e.g. AffectedAsset, ParentTickets).
    - `Object_0` : Index of the object stored in the Dynamic Field (Object_0 = 1st object, Object_1 = 2nd object, etc.). This can be used to reference a concrete object if the dynamic field contains several objects.
    - `Attribute` : respective attribute of the object (e.g. Firstname, Title) or "DynamicField" to specify the attribute structure for assets.

The references can be concatenated. For example, this allows you to access an attribute of a referenced asset (type AssetReference) that is stored on the referenced contact (type ContactReference):

- `<KIX_TICKET_DynamicField_myDFContactRef_Object_0_myDFAssetRef_Object_1_SectionGeneral_0_Serialnumber>`.

Examples:

- Contact
    - Login of the supervisor of the ticket contact:
        `<KIX_CONTACT_DynamicField_Supervisor_Object_0_UserLogin>`.
        - The supervisor is stored in a Dynamic Field of type "Contact Reference" on the contact.
- Organisation

- E-mail of the director of the ticket organisation:
  `<KIX_ORG_DynamicField_DirectorContact_Object_0_Email>`.
  - The director is stored in a Dynamic Field of type "Contact Reference" at the organisation.
- Asset
  - Model name of the first affected asset on a ticket:
    `<KIX_TICKET_DynamicField_AffectedAsset_Object_0_SectionGeneral_0_Model>`.
    - The model name is stored at the asset of the "Computer" or "Hardware" class ("Model" attribute in the "General information" section).
- Ticket
  - The planned effort on the first child ticket:
    `<KIX_TICKET_DynamicField_ChildTicket_Object_0_PlannedEffort>`.
- FAQ
  - Model name of the first relevant asset of the first FAQ entry deposited at a ticket (double referencing):
    `<KIX_TICKET_DynamicField_FAQReference_Object_0_RelatedAsset_Object_0_SectionGeneral_0_Model>`

18.2.4.4 Placeholder for the Ticket Organisation

The placeholder range ORG is available for tickets and refers to the organization saved in the ticket.

Placeholder	Meaning	Example
KIX_ORG_ID	Unique ID of organization	23
KIX_ORG_Number	Customer number/short name of organization	ExCo
KIX_ORG_Name	Full name of organization	Example Company Corp.
KIX_ORG_Url	URL of organization	http://www.examplecompany.com
KIX_ORG_Street	Street part of organization´s address	123 Mainstreet
KIX_ORG_City	City part of organization´s address	Example City

Placeholder	Meaning	Example
KIX_ORG_Zip	ZIP code part of organization´s address	12345
KIX_ORG_Country	Country part of organization´s address	USA
KIX_ORG_Comment	Comment regarding organization	Just a comment.
KIX_ORG_Valid	Validity of entry for further use	valid
KIX_ORG_CreateTime	Time stamp indicating when the organization was created	03/30/2020, 08:49 AM
KIX_ORG_ChangeTime	Time stamp indicating when organization was last changed	03/30/2020, 08:49 AM
KIX_ORG_DynamicField_XYZ	Individual dynamic field "XYZ" to organization	see Dynamic fields

18.2.4.5 Placeholder of the Ticket Contact

The placeholder range CONTACT is available for tickets and refers to the organization saved in the ticket.

Placeholder	Meaning	Example
KIX_CONTACT_ID	Unique ID of Contact	123
KIX_CONTACT_AssignedUserID	Unique user ID of the contact (if contact is also set up as a user)	456
KIX_CONTACT_UserLogin	Login ID of the contact (if contact is also set up as a user)	j.doe
KIX_CONTACT_PrimaryOrganisationID	ID of primary organization of contact	23

Placeholder	Meaning	Example
KIX_CONTACT_OrganisationIDs	Comma-concatenated list of IDs for all organizations of the contact	23,24
KIX_CONTACT_Firstname	First name of contact	John
KIX_CONTACT_Lastname	Last name of contact	Doe
KIX_CONTACT_Title	Title/form of address of contact	Mr.
KIX_CONTACT_Email	Email address of contact	somemailaddress@maildomain.com
KIX_CONTACT_Email1 KIX_CONTACT_Email2 KIX_CONTACT_Email3 KIX_CONTACT_Email4 KIX_CONTACT_Email5	Secondary email addresses of the contact	somemailaddress@maildomain.com
KIX_CONTACT_Phone	Telephone number of contact	+49 341 123 456 567
KIX_CONTACT_Mobile	Cell phone number of contact	+49 176 123 456 789
KIX_CONTACT_Fax	Fax number of contact	+49 341 123 456 567
KIX_CONTACT_Street	Street part of contact's address	123 Mainstreet
KIX_CONTACT_City	City part of contact's address	Example City
KIX_CONTACT_Zip	ZIP code part of contact's address	12345
KIX_CONTACT_Country	Country part of contact's address	USA
KIX_CONTACT_Comment	Comment regarding the contact	Just a comment.
KIX_CONTACT_Valid	Validity of entry for further use	valid

Placeholder	Meaning	Example
KIX_CONTACT_CreateTime	Time stamp indicating when the entry was created	03/30/2020, 08:49 AM
KIX_CONTACT_ChangeTime	Time stamp indicating when the entry was last changed	03/30/2020, 08:49 AM
KIX_CONTACT_DynamicField_XYZ	Individual dynamic field "XYZ" to contact	see Dynamic fields

18.2.4.6 Placeholder of the Ticket Teams (Queue)

The placeholder range QUEUE is available for tickets and refers to the Team (Queue) saved in the ticket.

Platzhalter	Bedeutung	Beispiel
KIX_QUEUE_QueueID	Unique ID of Team	23
KIX_QUEUE_Name	Designation of the team	Monitoring
KIX_QUEUE_FullName	Full name of the team (unique identifier)	Servicedesk::Monitoring
KIX_QUEUE_Signature	Configured signature in team settings	
KIX_QUEUE_Parent	Name of the structurally superior team	Servicedesk
KIX_QUEUE_ParentID	ID structurally superior teams	123
KIX_QUEUE_SystemAddressID	ID of the assigned email address	123
KIX_QUEUE_SystemAddress	Assigned Email address	somemailaddress@maildomain.com

Platzhalter	Bedeutung	Beispiel
KIX_QUEUE_FollowUp	Behaviour with demand emails and ticket closed	possible
KIX_QUEUE_FollowUpLock	Lock at demand-Emails	0 | 1
KIX_QUEUE_Comment	Comment regarding the contact	Just a comment.
KIX_QUEUE_Valid	Validity of entry for further use	valid
KIX_QUEUE_CreateTime	Time stamp indicating when the entry was created	03/30/2020, 08:49 AM
KIX_QUEUE_ChangeTime	Time stamp indicating when the entry was last changed	03/30/2020, 08:49 AM

18.2.4.7 Article-Related Placeholders

The placeholder ranges ARTICLE, CUSTOMER, LAST, FIRST, AGENT refer to an article that exists in a ticket. ARTICLE is used when forwarding or replying. CUSTOMER indicates the last article created by an external contact, LAST references the last, most recent article of the ticket, FIRST references the first, oldest article of the ticket, and AGENT indicates the last article created by an agent user. The basic article information is available as detailed below:

Placeholder KIX_ARTICLE	Placeholder KIX_CUSTOMER	Placeholder KIX_LAST	Placeholder KIX_FIRST	Placeholder KIX_AGENT	Meaning	Example
KIX_ARTICLE_ID	KIX_CUSTOMER_ID	KIX_LAST_ID	KIX_FIRST_ID	KIX_AGENT_ID	Unique Article ID	123
KIX_ARTICLE_ArticleID	KIX_CUSTOMER_ArticleID	KIX_LAST_ArticleID	KIX_FIRST_ArticleID	KIX_AGENT_ArticleID	Unique ArtikelID	123
KIX_ARTICLE_TicketID	KIX_CUSTOMER_TicketID	KIX_LAST_TicketID	KIX_FIRST_TicketID	KIX_AGENT_TicketID	Ticket ID of the article	123

Placeholder KIX_ARTICLE	Placeholder KIX_CUSTOMER	Placeholder KIX_LAST	Placeholder KIX_FIRST	Placeholder KIX_AGENT	Meaning	Example
KIX_ARTICLE_From	KIX_CUSTOMER_From	KIX_LAST_From	KIX_FIRST_From	KIX_AGENT_From	From-Sender of artikel	John.Doer@maildomain.com
KIX_ARTICLE_To	KIX_CUSTOMER_To	KIX_LAST_To	KIX_FIRST_To	KIX_AGENT_To	To-Recipient of artikel	John.Doer@maildomain.com
KIX_ARTICLE_Cc	KIX_CUSTOMER_CC	KIX_LAST_CC	KIX_FIRST_CC	KIX_AGENT_CC	Cc-Recipient of article	John.Doer@maildomain.com
KIX_ARTICLE_Subject	KIX_CUSTOMER_Subject	KIX_LAST_Subject	KIX_FIRST_Subject	KIX_AGENT_Subject	Subject line of article	Incident notification printer
KIX_ARTICLE_Body	KIX_CUSTOMER_Body	KIX_LAST_Body	KIX_FIRST_Body	KIX_AGENT_Body	Text of article content (Plaintext)	*(Cannot shown here)*
KIX_ARTICLE_BodyRichtext	KIX_CUSTOMER_BodyRichtext	KIX_LAST_BodyRichtext	KIX_FIRST_BodyRichtext	KIX_AGENT_BodyRichtext	Text **incl**. inline pictures (HTML formatted)	*(Cannot shown here)*

Placeholder KIX_ARTICLE	Placeholder KIX_CUSTOMER	Placeholder KIX_LAST	Placeholder KIX_FIRST	Placeholder KIX_AGENT	Meaning	Example
KIX_ARTICLE_BodyRichtext_*n*	KIX_CUSTOMER_BodyRichtext_*n*	KIX_LAST_BodyRichtext_*n*	KIX_FIRST_BodyRichtext_*n*	KIX_AGENT_BodyRichtext_*n*	As before, but with a text length limit of n lines. Defines, for example, how many lines of the article text are included in the quote. Fallback: SysConfig key *"Ticket::Placeholder::BodyRichtext::DefaultLineCount"* (default: 99 lines)	<KIX_ARTICLE_BodyRichtext_10> = Limitation to 10 lines, whereby empty lines are retained (0 = unlimited).
KIX_ARTICLE_BodyRichtextNoInline	KIX_CUSTOMER_BodyRichtextNoInline	KIX_LAST_BodyRichtextNoInline	KIX_FIRST_BodyRichtextNoInline	KIX_AGENT_BodyRichtextNoInline	Text **exclusive** inline images (HTML formatted)	*(Cannot shown here)*

Placeholder KIX_ARTICLE	Placeholder KIX_CUSTOMER	Placeholder KIX_LAST	Placeholder KIX_FIRST	Placeholder KIX_AGENT	Meaning	Example
KIX_ARTICLE_BodyRichtextNoInline_*n*	KIX_CUSTOMER_BodyRichtextNoInline_*n*	KIX_LAST_BodyRichtextNoInline_*n*	KIX_FIRST_BodyRichtextNoInline_*n*	KIX_AGENT_BodyRichtextNoInline_*n*	As before, but with a text length limit of n lines. Defines, for example, how many lines of the article text are included in the quote. Fallback: SysConfig key *"Ticket::Placeholder::BodyRichtext::DefaultLineCount"* (default: 99 lines)	<KIX_ARTICLE_BodyRichtextNoInline_10> = Limitation to 10 lines, whereby empty lines are retained (0 = unlimited).
KIX_ARTICLE_ContentType	KIX_CUSTOMER_ContentType	KIX_LAST_ContentType	KIX_FIRST_ContentType	KIX_AGENT_ContentType	MIME-Type of body	text/plain
KIX_ARTICLE_CreateBy	KIX_CUSTOMER_CreateBy	KIX_LAST_CreateBy	KIX_FIRST_CreateBy	KIX_AGENT_CreateBy	ID of the user who created the article	1
KIX_ARTICLE_ChangeBy	KIX_CUSTOMER_ChangeBy	KIX_LAST_ChangeBy	KIX_FIRST_ChangeBy	KIX_AGENT_ChangeBy	ID of the user who made the last change	123
KIX_ARTICLE_CreateTime	KIX_CUSTOMER_CreateTime	KIX_LAST_CreateTime	KIX_FIRST_CreateTime	KIX_AGENT_CreateTime	Timestamp creation of the article	1

Placeholder KIX_ARTICLE	Placeholder KIX_CUSTOMER	Placeholder KIX_LAST	Placeholder KIX_FIRST	Placeholder KIX_AGENT	Meaning	Example
KIX_ARTICLE_ChangeTime	KIX_CUSTOMER_ChangeTime	KIX_LAST_ChangeTime	KIX_FIRST_ChangeTime	KIX_AGENT_ChangeTime	Timestamp last change of the article	123
KIX_ARTICLE_REPLYRECIPIENT	-	KIX_LAST_REPLYRECIPIENT	KIX_FIRST_REPLYRECIPIENT	-	Use in article actions (KIX Pro) that are to generate a response to an article. The placeholder contains the recipient of a reply to an article. The mail header "Reply-To" contained in the article is considered with the highest priority. • **From:** is used if "Reply-To" is not set and "From" is <u>not</u> a system address. • **To:** is used if "Reply-To" is not set and "From" <u>is</u> a system address.	

18.2.4.8 Asset-Related Placeholders

Standard Asset-Attributes

Placeholders for standard asset attributes have the form: <KIX_ASSET_Attribute>.

Placeholder	Meaning	Example
KIX_ASSET_ID	Unique ID of the asset	123
KIX_ASSET_ConfigItemID	Unique ID of the asset	123
KIX_ASSET_Number	Number of the asset	424000026
KIX_ASSET_Name	Name of the asset	HP Pro Tower Desktop PC
KIX_ASSET_ClassID	ID of the asset class of an asset	4
KIX_ASSET_Class	Name of the asset class of an asset	Computer
KIX_ASSET_CurDeplStateID	ID of the current deployment state	16
KIX_ASSET_CurDeplState	Name of the current deployment state	Produktion
KIX_ASSET_CurDeplStateType	Type of the current deployment state	productive
KIX_ASSET_LastVersionID	ID of the current version	213
KIX_ASSET_CurInciStateID	ID of the current incident state	3
KIX_ASSET_CurInciState	Name of the current incident state	Incident
KIX_ASSET_CurInciStateType	Type of the current incident state	incident
KIX_ASSET_CreateTime	Timestamp creation of the asset	03/30/2020, 08:49 AM

Placeholder	Meaning	Example
KIX_ASSET_CreateBy	ID of the user who created the asset	
KIX_ASSET_ChangeTime	Timestamp last change of the asset	03/30/2020, 08:49 AM
KIX_ASSET_ChangeBy	ID of the user who made the last change	

Class-specific asset attributes

Placeholders for class-specific attributes have the form:

`<KIX_ASSET_ParentAttribut_0_SubAttribut_0...>` (0 = index).

For the parent attribute, the index (e.g. 0) must be specified if a sub-attribute follows. If a placeholder cannot be resolved (e.g. the index or the attribute is unknown or does not exist), it is replaced with the standard (usually: "-"). "Key" and "Value" are identical for some attribute types (e.g. text, text area).

Placeholder	Meaning	Placeholder example	Replacement example
..._Attribute_Keys	comma-separated list of all key values	<KIX_ASSET_SectionGeneral_0_Type_Keys>	1, 2, 3
..._Attribute_Values	comma-separated list of all value values	<KIX_ASSET_SectionGeneral_0_Type_Values>	Desktop, Laptop, Server
..._Attribute	like "_Values"	<KIX_ASSET_SectionGeneral_0_Type>	Desktop, Laptop, Server
..._Attribute_0_Key	Stored value, e.g. relevant for ID of GeneralCatalog entry	<KIX_ASSET_SectionGeneral_0_Type_0_Key>	1
..._Attribute_0_Value	Prepared value or like Key, if not necessary (e.g. for text)	<KIX_ASSET_SectionGeneral_0_Type_0_Value>	Desktop
..._Attribute_0	like "_Value"	<KIX_ASSET_SectionGeneral_0_Type_0_Key>	Desktop

Dynamic Fields Type "Asset Reference"

Placeholders of the form `<KIX_TICKET_DynamicField_AffectedAsset>` can be used to access concrete detailed properties of the referenced asset. For example, text modules can be created that use properties of the affected asset or service without having to copy these values to ticket fields.

- The same rules apply as for class-specific asset attributes (see above).
    - The form is: `<KIX_TICKET_DynamicField_AffectAsset_Object_0_Attribute>`
- The name of the dynamic field must be specified (e.g. AffectedAsset).
- It must be specified that the object behind it is to be addressed (`_Object`).
    - *Object* can be used to access the referenced objects in a dynamic field,
    - An index must be specified (1st Element/Asset = 0; 2nd Asset = 1; etc.)
        - A 0 after the object determines the 1st asset in the dynamic field.
    - then continue as above, e.g.
        - matures to the type of computer or hardware asset:
            - `<KIX_TICKET_DynamicField_AffectAsset_Object_0_SectionGeneral_0_Type>`.
            - `<KIX_TICKET_DynamicField_AffectAsset_Object_0_SectionGeneral_0_Type_0_Value>`
        - Accesses the planned effort of the ticket deposited service:
            - `<KIX_TICKET_DynamicField_AffectServices_Object_0_SectionGeneral_0_PlannedEffort>`.
        - Accesses the street of the address stored at the asset:
            - `<KIX_TICKET_DynamicField_AffectAsset_Object_0_SectionAddress_0_Street>`
- Deviations from the form lead to the standard fallback (usually: " `–` ")

18.2.4.9 Placeholder in Notifications

The placeholders that you already know from CONTACT are available in notifications. However, they relate to the respective recipient of the notification, provided that they have been saved as a contact in the system.

Placeholder	Meaning	Example
KIX_NOTIFICATION_RECIPIENT_ID	Unique ID of recipient	123
KIX_NOTIFICATION_RECIPIENT_Login	Login identifier of recipient	j.doe

Placeholder	Meaning	Example
KIX_NOTIFICATION_RECIPIENT_ PrimaryOrganisationID	ID of primary organization of recipient	23
KIX_NOTIFICATION_RECIPIENT_ OrganisationIDs	Comma-concatenated list of IDs for all organizations of the recipient	23,24
KIX_NOTIFICATION_RECIPIENT_ Firstname	First name of recipient	John
KIX_NOTIFICATION_RECIPIENT_ Lastname	Last name of recipient	Doe
KIX_NOTIFICATION_RECIPIENT_ Title	Title/form of address of recipient	Mr.
KIX_NOTIFICATION_RECIPIENT_ Email	Email address of recipient	somemailaddress@maildo main.com
KIX_NOTIFICATION_RECIPIENT_ Phone	Telephone number of recipient	+49 341 123 456 567
KIX_NOTIFICATION_RECIPIENT_ Mobile	Cell phone number of recipient	+49 176 123 456 789
KIX_NOTIFICATION_RECIPIENT_ Fax	Fax number of recipient	+49 341 123 456 567
KIX_NOTIFICATION_RECIPIENT_ Street	Street part of recipient's address	123 Mainstreet
KIX_NOTIFICATION_RECIPIENT_ City	City part of recipient's address	Example City
KIX_NOTIFICATION_RECIPIENT_ Zip	ZIP code part of recipient's address	12345

Placeholder	Meaning	Example
KIX_NOTIFICATION_RECIPIENT_Country	Country part of recipient's address	USA
KIX_NOTIFICATION_RECIPIENT_Comment	Comment regarding the recipient	Just a comment.
KIX_NOTIFICATION_RECIPIENT_Valid	Validity of entry for further use	valid
KIX_NOTIFICATION_RECIPIENT_CreateTime	Time stamp indicating when the entry was created	03/30/2020, 08:49 AM
KIX_NOTIFICATION_RECIPIENT_ChangeTime	Time stamp indicating the last change to the entry	03/30/2020, 08:49 AM

18.2.4.10 Placeholders from Configuration / System

These placeholders are available in the backend or system core and refer to configuration entries. All SysConfig keys can be used. The following table contains only a selection.

Note: Some placeholders have the access level "confidential" (indicated in the SysConfig by orange writing). These are excluded from use by the agent and will not be resolved if used by the agent in text fields. This ensures that agents cannot read out any unauthorised information (see also Placeholders).

Placeholder	Meaning	Example
KIX_CONFIG_Organization	Information saved in SysConfig regarding for the short name of organization	Example Company
KIX_CONFIG_OrganizationLong	Information saved in SysConfig regarding for long name of organization	Example Company Corp.
KIX_CONFIG_OrganizationAddress	Information saved in SysConfig regarding registered address of organization	123 Mainstreet, 12345 Example City, USA

Placeholder	Meaning	Example
KIX_CONFIG_OrganizationRegistrationLocation	Information saved in the SysConfig regarding where organization is registered	Registry Court Example City
KIX_CONFIG_OrganizationRegistrationNumber	Information saved in the SysConfig regarding organization's registry number	HRB 123456
KIX_CONFIG_OrganizationDirectors	Information saved in the SysConfig regarding management board of organization	Managing Director John Doe
KIX_CONFIG_OrganizationHotline1	Information saved in the SysConfig regarding telephone/fax contact 1 for organization	Tel: +49 123 456 700
KIX_CONFIG_OrganizationHotline2	Information saved in the SysConfig regarding telephone/fax contact 2 for organization	Tel: +49 123 456 725
KIX_CONFIG_Ticket::Hook	Identifier set in SysConfig "Ticket::Hook" for identifying inbound emails	T#
KIX_CONFIG_FQDN	Information saved in the SysConfig regarding FQDN of the KIX frontend	your-frontend-host.example.com
KIX_CONFIG_FQDN_Frontend KIX_CONFIG_FQDN_Backend KIX_CONFIG_FQDN_SSP	Information saved in the SysConfig regarding FQDN of the KIX frontend, KIX backend or KIX Self Service Portal	• your-frontend-host.example.com • your-backend-host.example.com • your-ssp-host.example.com
KIX_NOW	returns the current (local) time as TimeStamp - see also Variables	2022-05-30 13:28:08

Placeholder	Meaning	Example
KIX_NOW_DateTime	returns the current (local) date and time from the timestamp (analog KIX_NOW) - see also Variables (see page 786)	2022-05-30 13:28:08
KIX_NOW_Date	supplies only the current (local) date from the timestamp - see also Variables (see page 786)	2022-05-30
KIX_NOW_Time	supplies only the current (local) time from the time stamp - see also Variables (see page 786)	13:28:08

18.2.4.11   Placeholder in KIX Pro

The placeholder area TICKET is available for tickets and represents the property of the ticket under consideration. For individual fields ("Dynamic Fields"), this list has to be extended depending on the specific KIX installation.

Ticket-Related Placeholders for predefined Dynamic fields

Placeholder	Meaning	Example
KIX_TICKET_DynamicField_ParentTickets	List of parent tickets	Ticket#2020012760021 - Test ticket A, Ticket#2020012760011 - Test ticket B
KIX_TICKET_DynamicField_ChildTickets	List of child tickets	Ticket#2020012760021 - Test ticket A, Ticket#2020012760011 - Test ticket B
KIX_TICKET_DynamicField_RelatedTickets	List of ticket references	Ticket#2020012760021 - Test ticket A, Ticket#2020012760011 - Test ticket B
KIX_TICKET_DynamicField_MergeToTicket	Ticket to which the ticket in question was combined	

Ticket-related placeholder areas for SLA and SLA criterias

Placeholder	Description	Example	Comment
KIX_TICKET_Accounted Time	Legible representation of the time expenditure / time booking. Negative numbers and 0 are possible.	Indication in minutes In text modules: (number)h (number)m	If the placeholder in a new article supplies new times, these are not part of the placeholder replacement. Background: First the placeholder is replaced, then the article is created and then the time is saved..
KIX_TICKET_SLA	Name of the SLA	SLA by Affected Asset/s	
KIX_TICKET_SLAID	ID of the SLA	2	
Placeholder for SLA criterias			
Reaction time			
KIX_TICKET_SLACriteria _FirstResponse_Busine ssTimeDeviation	Deviation of the fulfillment time from the target time	748	
KIX_TICKET_SLACriteria _FirstResponse_Calend ar	Calendar on which the SLA of the response time is based	calendar001	
KIX_TICKET_SLACriteria _FirstResponse_FirstRe minderTime	Time stamp on which the reminder notification for the criterion is sent	2020-06-03 12:15	

KIX_TICKET_SLACriteria _FirstResponse_Fulfillm entTime	Timestamp at which the criterion is fulfilled	2020-06-03 13:25	
KIX_TICKET_SLACriteria _FirstResponse_Name	Name of the SLA criterion	FirstResponse	
KIX_TICKET_SLACriteria _FirstResponse_SLAID	ID of the SLA from which the response time originates	4	
KIX_TICKET_SLACriteria _FirstResponse_StartTi me	Timestamp of the start of the reaction time runtime	03.06.2020 11:05	
KIX_TICKET_SLACriteria _FirstResponse_Status	Status of the SLA criterion	undetermined	
KIX_TICKET_SLACriteria _FirstResponse_TargetT ime	Time stamp by when the reaction must be made	2020-06-03 12:45	
KIX_TICKET_SLACriteria _FirstResponse_TargetT imeDiff	Response time as specified when creating the SLA	2h 15m	
KIX_TICKET_SLACriteria _FirstResponse_TimeDe viation	Deviation of the fulfilment time from the target time in minutes	1408	
KIX_TICKET_SLACriteria _FirstResponse_Violatio n	Violation of the response time	No	
Solution time			
KIX_TICKET_SLACriteria _Solution_BusinessTim eDeviation	Deviation of the fulfilment time from the target time (business hours are taken into account) in minutes	694	

KIX_TICKET_SLACriteria_Solution_Calendar	Calendar underlying the SLA of the resolution time	calendar001	
KIX_TICKET_SLACriteria_Solution_FirstReminderTime	Timestamp at which the reminder notification for the criterion is sent out	2020-06-03 12:15	
KIX_TICKET_SLACriteria_Solution_FulfillmentTime	Timestamp at which the criterion is fulfilled	2020-06-03 13:25	
KIX_TICKET_SLACriteria_Solution_Name	Name of the SLA criterion	Solution	
KIX_TICKET_SLACriteria_Solution_SLAID	ID of the SLA from which the response time originates	4	
KIX_TICKET_SLACriteria_Solution_StartTime	Timestamp of the start of the solution time runtime	03.06.2020 11:05	
KIX_TICKET_SLACriteria_Solution_Status	Status of the SLA criterion	undetermined	
KIX_TICKET_SLACriteria_Solution_TargetTime	Time stamp by when the solution must be made	2020-06-03 12:45	
KIX_TICKET_SLACriteria_Solution_TargetTimeDiff	Resolution time as specified when creating the SLA	2h 0m	
KIX_TICKET_SLACriteria_Solution_TimeDeviation	Deviation of the fulfilment time from the target time in minutes	1354	
KIX_TICKET_SLACriteria_Solution_Violation	Violation of the resolution time	No	

Placeholders for ticket templates and actions

The placeholders listed below can be used for the configuration of ticket templates as well as for dynamic ticket actions in templates. The placeholders are used on the "Input fields" page of the dialogs "New template", "Edit template", "New action", "Edit action".

If a template is selected by the user, the placeholders are used as follows:

- When editing, the replacement of placeholders with ticket reference refers to the source ticket
- In a new ticket, the fields are empty (for ticket placeholders)

The implementation refers to text input fields & drop-down fields.

Placeholder	Meaning	Example
KIX_CURRENT_UserID	Unique user ID for the user	123
KIX_CURRENT_Firstname	First name of the contact assigned to the user	John
KIX_CURRENT_Lastname	Last name of the contact assigned to the user	Doe
KIX_CURRENT_Email	Email address of the contact assigned to the user	john.doe@kix.cloud
KIX_CURRENT_Mobile	Cell phone number of the contact assigned to the user	+49 123 456 789
KIX_CURRENT_UserPhone	Phone number of the contact assigned to the user	+49 987 654 321
KIX_CURRENT_UserLogin	Login name of the user	jodo
KIX_CURRENT_UserLanguage	Language setting of the user	en
KIX_CURRENT_UserComment	Comment on user entry	

Placeholder	Meaning	Example
KIX_CURRENT_CreateTime	Time stamp indicating when the user entry was created	2020-01-01 00:00:00
KIX_CURRENT_ChangeTime	Time stamp indicating the last change to the user entry	2020-01-01 00:00:00
KIX_CURRENT_IsCustomer	Indicates whether the user is permitted to log in as the customer contact	0 | 1
KIX_CURRENT_IsAgent	Indicates whether the user is permitted to log in as the agent	0 | 1
KIX_CURRENT_Title	Title of the contact assigned to the user	Sir
KIX_TICKET_TicketNumber	Ticket number	202001014200001
KIX_TICKET_Title	Ticket title	Order
KIX_TICKET_TicketID	Unique ID of ticket	123
KIX_TICKET_State	Name of state	open
KIX_TICKET_StateID	ID of state	2
KIX_TICKET_StateType	Name of state type; of set state	open
KIX_TICKET_PendingTime	Pending time if status of type "pending reminder" or "pending auto" has been set	2020-05-23 09:00:00
KIX_TICKET_Priority	Name of priority	3 normal
KIX_TICKET_PriorityID	ID of priority	3
KIX_TICKET_Lock	Designation of lock identifier	lock

Placeholder	Meaning	Example
KIX_TICKET_LockID	ID of lock identifier	2
KIX_TICKET_Queue	Name of team/queue	Service desk
KIX_TICKET_QueueID	ID of team/queue	1
KIX_TICKET_Organisation	Name of organization saved in ticket	Example Ccompany Corp.
KIX_TICKET_OrganisationID	ID of contact/customer user saved in ticket	123
KIX_TICKET_Contact	Name of contact/customer user saved in ticket	John Doe
KIX_TICKET_ContactID	ID of contact/customer user saved in ticket	123
KIX_TICKET_Owner	Login name of owner entered in ticket	limu
KIX_TICKET_OwnerID	ID of owner entered in ticket	123
KIX_TICKET_Type	Name of ticket type	Incident
KIX_TICKET_TypeID	ID of ticket type	2
KIX_TICKET_Responsible	Login name of person responsible entered on ticket	limu
KIX_TICKET_ResponsibleID	ID of person responsible ID entered on ticket	123
KIX_TICKET_Created	Time stamp indicating when ticket was created	2020-01-01 09:00:00
KIX_TICKET_CreateBy	ID of the user who created the ticket	1

Placeholder	Meaning	Example
KIX_TICKET_Changed	Time stamp indicating the last change to the ticket	2020-04-01 09:00:00
KIX_TICKET_ChangedBy	ID of the user who made the last change	123
KIX_TICKET_TicketID	Unique ID of ticket	12345

18.3 Variables

You can define variables, for example to store the result values returned by macro actions in them. By calling this result variable, you can reference the values stored in it and process them further.

Variables are used in, among other things, jobs, report definitions and in the pre- and post-actions of ticket actions.

Content on this page:

18.3.1 Data types

Variables can hold different types of values. When implying variables, it is not necessary to explicitly specify the data type. It is determined by the macro action in which the variable is initialised.

However, the configuration of report definitions is an exception. For the variables resulting from the parameters in the SQL statement, the data type must be specified so that the values are used correctly.

The following data types are possible:

Data type	Description	Hints
String	Alphanumeric characters and strings	The characters and strings are always enclosed in inverted commas so that they are interpreted as text, e.g.: • `varText1 == "1"` • `varText2 == "The postcode of Chemnitz is: 09113"` Calculations with variables of the data type `string` are not possible, as these are pure text characters.
Numeric	Numeric values/numbers	Variables of the data type `number` are numerical values with which calculations are also possible. They must not be placed in inverted commas, otherwise they would be interpreted as text, e.g.: • `varNR1 == 1` • `varNR2 == 3.14159265359`

Data type	Description	Hints

Object	Objects	Variables of the data type `object` can hold several objects with different parameters. By using the filter "jq", parts of them can be extracted and used further (see variable filter below).
		
		Example:
		
		``
		varTest == {
		  "Objects":[
		    {
		      "id":"47441",
		      "value":"Moers",
		    },
		    {
		      "id":"47798",
		      "value":"Krefeld"
		    }
		  ]
		}
		``

Data type	Description	Hints
Date DateTime Time	Date values	Variables of the data type `date` can hold date values - without time information - in the format YYYY:MM:DD. The time specifications are set in inverted commas, e.g.: • `varDate == "2022:03:04"` Variables of the data type `DateTime` can hold date values including time information in the format YYYY:MM:DD hh:mm. The time information is set in inverted commas, e.g.: • `varDateTime == "2022:03:04 11:55"` Variables of the data type `Time` can hold time values - without date values - in the format hh:mm:ss. The time values are set in inverted commas, e.g.: • `varTime == "11:55:05"`

> ⓘ **Info**
>
> When initialising variables in KIX, it is not necessary to explicitly specify the data type. It depends on the macro action in which the variable is initialised. Exception: Configuration of report definitions. Please note: The variables are not initialised globally, but always locally. Their scope extends only to the current object (job/action/report definition). It is therefore not possible to further process a variable created in job A in job B.

18.3.2 Initialize Variables

A variable is initialized by specifying its variable identifier, e.g. in the "result identifiers" of a macro action. You can assign variable identifiers freely, but without special characters and without spaces. Develop your own, uniform naming convention for naming the variables (e.g. varTicketIDTechnology, varArticleIDSupport) to keep track of things.
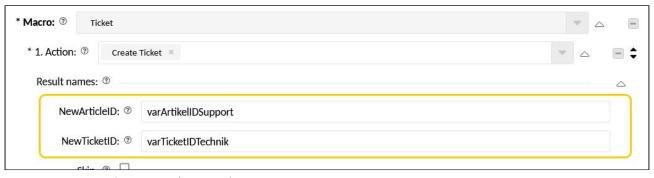
Fig.: Initialization of variables (example)

18.3.3 Referencing Variables

To access the values stored in a variable, the variable is referenced with `${VariableIdentifier}`. For example, if the variable identifier is "varTicketIDTechnik", the variable is called with `${varTicketIDTechnik}` and its values are processed.

You can use variables wherever you need their values. For example in the title and/or body of an article, in subsequent macro actions (e.g. for calculations) or to set the value in a dynamic field. However, the prerequisite is that the variable has been initialized before it is used. Since the macro actions of a job are processed in sequence - from top to bottom - the variable must already have been initialized in one of the previous macro actions.

Fig.: Putting the value stored in a variable in a dynamic field

Use Case: see Automating Reports

18.3.4 Arrays

Variables can hold several values and thus form an array. This can be used, for example, if the data to be processed contains a numbered attribute (Ticket1, Ticket2, ...) and all entries are to be processed in a loop.

The individual array values are separated by commas:

```
${Variable1, Variable2}
```

If one of the variables listed is already an array, the individual values are merged:

```
Variable1 = ['Test1.1','Test1.2']
Variable2 = ['Test2.1','Test2.2']
${Variable1, Variable2} = ['Test1.1','Test1.2','Test2.1','Test2.2']
```

18.3.5 Representation of Complex Objects

Variables can also represent complex objects, lists or mixed forms. The properties of the ResultObject are accessed using a period `(.)` . If the property is an array, you can use ":<index>" to access an element of the array.
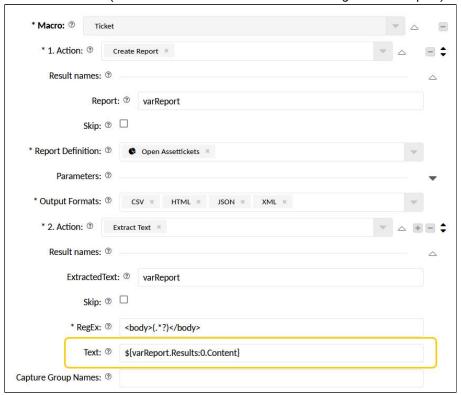
Sample report output file: `${varReport.Results:0.Content}`

- The `varReport` variable is created and returned in the "Create report" action.
  - The `varReport` variable has a `Results` property. `Results` is an array containing the reports in the respective result formats (HTML, CSV, JSON, XLSX, etc.).
    - By accessing `":0"` the first result format is referenced (here: HTML)

- The `Content` property is used by the first result format. This refers to the entire file content (here the entire HTML document of the generated report).

Use Case: see Automating Reports

18.3.6 Special Variables

The `RootObjectID` and `ObjectID` variables refer to triggering objects. They can be used like placeholders, for example to provide information about the affected asset in the ticket. Use cases can be:

If the license for a software asset expires, a job creates a ticket that shows the license to be renewed. The maintenance ticket generated by a job already contains various attributes of the asset to be maintained in the article text (e.g. name, location, address).

Variable	Description

`RootObjectID`	ID of the object that <u>initially triggered</u> the job, e.g:
	
	 • the triggering ticket on an event-based job
	   • Example: If a job is triggered as soon as a new ticket is created (Event: "Ticket Create"), the `RootObjectID` corresponds to the ID of this ticket.
	 • the object for which the job is being run
	   • Example: If a filter is stored in a (time-based or manually triggered) job, the `RootObjectID` corresponds to the ID of the object to which the filter conditions apply. Depending on the job type, this can be an asset or a ticket.
	
	 Note: The type of object (asset, ticket) - and thus the type of `RootObjectID` returned - depends on the selected job type. Based on the selected filter, it determines whether and what type of object is to be searched for. However, the job types Reporting and Synchronization do not provide any filters, so that no triggering object can be determined. `RootObjectID` is therefore empty.

`ObjectID`	ID of the <u>currently used</u> object. • corresponds to the `RootObjectID` for a non-interleaved job • corresponds to the ID of the relevant sub-macro in an interleaved job, e.g. with "execute macro" of the given `ObjectID` ; for "Loop" the value of the loop variable.

Use Case: see Periodic job "Licence renewal"

18.3.7 Variable filter

You can apply filters to variables to access their partial values. The variable must already have a value. The value can come from an object, a placeholder or the result of a previous macro action. The upper/lower case of the filters is irrelevant. The backend converts the information into lower case before processing.

Variable filters are indicated by the name of the variable to be processed, followed by a vertical bar "|".

Syntax:	`${VariableName|VariableFilter}`
Example:	`${varTest|jq(.Objects[] :: select(.id == "47441").value)}`

Several variable filters can be specified separated from each other by multiple specification of "|".

Syntax:	`${VariableName|VariableFilter1|VariableFilter2|VariableFilter3}`
Example:	`${varStart|DateUtil.bob|DateUtil.eob|DateUtil.unixTime}`

18.3.7.1 JSON filter "jq"

By using the filter "jq", the extracted part can be further processed as JSON. It replaces the former filter "FromJSON".

If, for example, a variable contains several objects with place names and their geo-position data as values, you can use the jq filter to extract the places or the individual geo-position data as JSON strings and then process them further.

Example: Values of an object

```
{
    "Objects": [
        {
            "id": "47441",
            "value": "Moers",
            "lattitude": 51.4463,
            "longitude": 6.6396
        },
```

```
        {
            "id": "47798",
            "value": "Krefeld",
            "lattitude": 51.3311,
            "longitude": 6.5616
        }
    ]
}
```
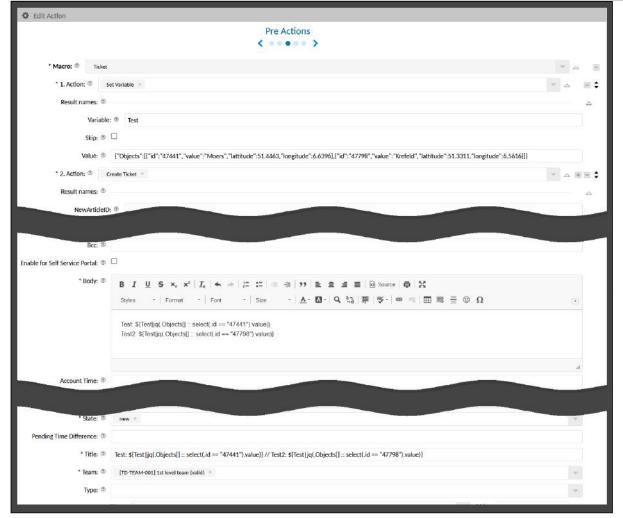

Fig.: Example of using variable filters

The example in the picture first initialises a variable "Test" (1st action). This variable are assigned 2 objects as values.

```
{"Objects":
[{"id":"47441","value":"Moers","lattitude":51.4463,"longitude":6.6396},
{"id":"47798","value":"Krefeld","lattitude":51.3311,"longitude":6.5616}]}
```

1. Object with the following values:

- ID: 47441
- Value: Moers
- and the geoposition data given under "lattitude" and "longitude".

2. Object with the following values:
- ID: 47798
- Value: Krefeld
- and the geoposition data specified under "lattitude" and "longitude

With the 2nd action, a new ticket is created. The variable values are output in its title as well as in the article text. For this purpose, partial values of the objects are extracted and output using the filter "jq".

Article text:

```
Test: ${Test|jq(.Objects[] :: select(.id == "47441").value)}
Test2: ${Test|jq(.Objects[] :: select(.id == "47798").value)}
```

Title:

```
Test: ${Test|jq(.Objects[] :: select(.id == "47441").value)} // Test2: $
{Test|jq(.Objects[] :: select(.id == "47798").value)}
```

ℹ Within jq, the pipes (|) must be replaced by double colons (::).

When the action is executed, the specified IDs are selected and their values defined under "value" are output in the title and in the article text of the new ticket.

The result is the following ticket:
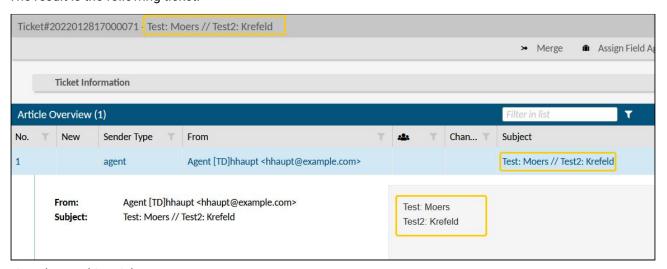
Fig.: The resulting ticket

Further information on jq can be found, for example, at:

- https://stedolan.github.io/jq/
- Editor: https://jqplay.org/

18.3.7.2   XML filter "XMLUtil.FromXML"

The variable filter "XMLUtil.FromXML" enables the use of XML structures as input values for macro actions (e.g. for XML transformation or macro actions with structured input parameters). A well-formed XML is required.

This means, for example, that the XML string received via the "Webhook extended" macro action or the content of an XML file in an email attachment can be further processed in a macro action.

Syntax example:	`${VariablenName|XMLUtil.FromXML}`

18.3.7.3   Date/Time Filter "DateUtil

With the "DateUtil" filter, date/time information can be extracted in order to use it for automated calculations. If the output values are based on dynamic fields of the type Date or DateTime, dates for resubmissions or for the next rule check of an asset, for example, can be calculated automatically (see also Mathematical Calculations with "Calculate" (KIX Pro)).

For the calculations, the filter supports the parameters listed in the following table. If the functions receive an invalid value (incorrect format or incorrect entries), the input value remains unchanged.

Syntax Example:	`${VariableName|DateUtil.bob}`

Parameter	Description	Notes
BOB	Calculates the "Begin of Businessday" based on the input value.	• Initial: 8:00 am (based on the default calendar in the SysConfig key "*TimeWorkingHours*"). • Accepts Timestamp and UnixTime as input values. • If the date is not a working day, the next working day is taken. • Result is a TimeStamp

Parameter	Description	Notes
EOB	Calculates the "End of Businessday" based on the input value.	• Initial: 21:00 (based on the default calendar in the SysConfig key "*TimeWorkingHours*"). • Accepts Timestamp and UnixTime as input values. • If the date is not a working day, the next working day is taken. • Result is a TimeStamp
UnixTime	Returns the entered TimeStamp as UnixTime (number of seconds).	• UnixTime means: number of seconds since the beginning of the epoch (e.g.: 1653388794). Depending on the operating system, the beginning of the epoch is 1 January 1970, 00:00 UTC (see also https://en.wikipedia.org/wiki/Unix_time).
TimeStamp	Returns the entered UnixTime as TimeStamp	• TimeStamp means: date/time in format: "`YYYY-MM-DD hh:mm:ss`" (e. g. "2022-04-21 12:00:00")

Parameter	Description	Notes

Calc	Calculates a new date or time based on the input value.	• Example: `${varDF1|DateUtil.bob|DateUtil.calc(+1Y -1M +1w -5d +2h)}`
		• The indication of the sign (+ or -) is mandatory! Without a sign, the value is ignored.
		   • Correct: " `+1d` "
		   • Incorrect: " `1d` "
		• Several entries (with spaces in between) can be made (e. g. " `+1d -4m +2Y` ")
		• Possible time units are:
		   • `Y` - Year
		   • `M` - Month
		   • `w` - Week
		   • `d` - Day
		   • `h` - Hour
		   • `m` - Minute
		   • `s` - Second
		   • without unit: s - second is assumed
		• The result is a TimeStamp.
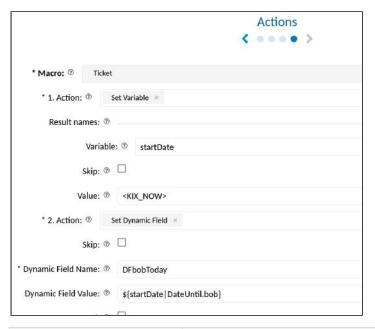
Placeholder for system times

You can use the following KIX placeholders to determine times. The placeholders are currently only available in the backend and serve to read out the system times. They cannot be used by the frontend, e.g. in text modules.

The values supplied by the placeholders must first be written into a variable in order to be able to calculate with them afterwards.
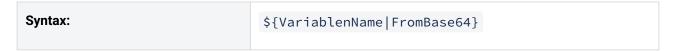
Placeholder	Description	Example
KIX_NOW	returns the current (local) time as TimeStamp	2022-05-30 13:28:08
KIX_NOW_DateTime	returns the current (local) date and time from the timestamp (analog KIX_NOW)	2022-05-30 13:28:08
KIX_NOW_Date	supplies only the current (local) date from the timestamp	2022-05-30
KIX_NOW_Time	supplies only the current (local) time from the time stamp	13:28:08

18.3.7.4   Filter "FromBase64"

The variable filter "FromBase64" returns the content of a Base64 encoded variable decoded in a macro action.

Example: A third party system delivers a PDF in the form of a Base64 string. The filter decodes the Base64 string so that a PDF is generated from it. The PDF can then be saved as an attachment to an article, for example.

Syntax:	`${VariablenName|FromBase64}`

18.3.7.5 Filter "ToBase64"

The variable filter "ToBase64" in a macro action encodes the content of a variable into a Base64 string. This makes it possible, for example, to transfer article attachments in binary form to third-party systems (e.g. DMS "D3").

Syntax:	`${VariablenName|ToBase64}`

18.4 Macro Actions

Macro Actions are a sequence of configured instructions that are processed in sequence from top to bottom. (1st action -> 2nd action -> 3rd action, etc.). The macro actions can be carried out on the triggering object (ticket) as well as on a referenced object (child ticket, parent ticket).

Macro Actions can be found

- when creating or editing a job in the 4th step (Actions)
- when creating or editing actions (KIX Pro) in the "Pre Actions" and "Post Actions".

All MacroActions stored in the SysConfig are available for selection (ke*y "Automation::MacroActionType::Ticket###[MacroActionName]". The following list provides an overview of this.
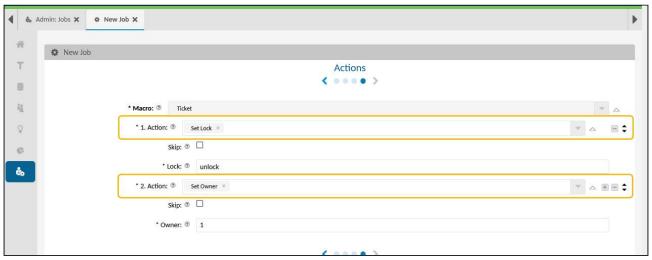
Fig .: 2 actions in one job

18.4.1 Notes on the use of Macro Actions

> ⚠️ **Important**
>
> - Macro Actions can be interleaved. Avoid endless loops!
> - The macro actions are only validated when the job is executed; any errors are recorded in the kix.log or job.log ("History" tab).

- You can work with variables and placeholders. Variables and placeholders always reference existing objects. Variables must therefore be declared in the same macro or in a preceding macro (within the job). For placeholder substitution, the dynamic fields must be created in the system.

- KIX placeholders are resolved in the default language. To prevent this, you can place an exclamation mark after the placeholder name: <KIX_TICKET_State! The exclamation mark explicitly suppresses the localisation of placeholders. For more information, see: Placeholders
  In the following illustration, the invalid status "new" (default language German) is prevented from being set instead of the status "new", for example. The same applies to the wait time.
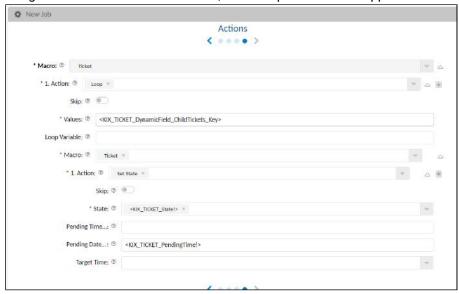
Fig.: The exclamation mark in the placeholder prevents localisation.

- Result variables are complex variables that can store the result of a macro action for further use.

- Regardless of their value, variables and placeholders are always replaced 1:1 BEFORE the job is executed. If a variable does not contain a value or is undefined due to a typing error, it will be calculated with "null" in calculations, for example, which will lead to an incorrect result. Example:
  - Formula: '1 + ${Variable}' results: '1 +'
  - Conditional: '${Variable} eq "test"' results: ' eq "test"'

- Macro Actions have the properties:
  - MacroID: the ID of the macro
  - ReferencedMacroID: the respective sub-macro - if available.

18.4.2 Overview of Macro Actions for KIX Start

- Assemble Object (see page 805)
- Conditional (see page 805)
- Create Article (see page 807)
- Create Report (see page 807)
- Create Ticket (see page 808)
- Create Ticket from System Template (see page 808)
- Delete Article (see page 809)
- Delete Ticket (see page 809)
- Execute Macro (see page 810)
- Extract Text (see page 810)
- Fetch Asset Attributes (see page 811)
- Loop (see page 811)
- Set Contact (see page 812)
- Set Dynamic Field (see page 812)
- Set Lock (see page 814)
- Set Organisation (see page 814)
- Set Owner (see page 814)
- Set Priority (see page 815)
- Set Responsible (see page 815)
- Set State (see page 815)
- Set Team (see page 817)
- Set Title (see page 817)
- Set Type (see page 817)
- Set Variable (see page 818)

18.4.2.1 Assemble Object

Merges several objects into a new object.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints

Object: Identifier of the result variable.
The specified identifier is a "text representation" of the object declared under "Definition".
Its attributes and sub-attributes can be accessed (`ObjectName.Definition.Attribute.SubAttribute`).
 	
The object has the properties:
 	
• **Type:** Format (data structure) of the object (JSON or YAML).
• **Definition**: Declaration of the object (Perl structure). You can use placeholders and variables. The generated object is made available as a result variable.
 	
Example Definition
 	
`{`
`"Filename":"dummy.csv",`
`"ContentType":"${Report.Results:0.ContentType}",`
`"Content":"${Report.Results:0.Content|JSON}`
`}`
	Can, for example, attach a table extracted from a report as an article attachment to a ticket.
	
	An attachment object created under "Definition" has 3 properties:
	
	• `"Filename"` : Name and file extension of the attachment object created (e.g. "`dummy.csv`")
	• `"ContentType"` : MIME Type of the content (e.g. `${Report.Results:0.ContentType}`)
	Information on this can also be found at: https://wiki.selfhtml.org/wiki/MIME-Type/Übersicht (German Website)
	"`Content`": Content of the attachment object with specification of the pipe (e.g. `${Report.Results:0.Content|JSON}`).
	The specification of the pipe with JSON is necessary to validly insert the content into the JSON format.
	
	For an application example, see Generate reports automatically .

18.4.2.2 Conditional

Checks whether a condition is true and thus enables the control flow in macros.

The macro is executed if the condition is true.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
If: Logical expression of the condition E.g.: `defined ${CalcResult}` (= Is the variable "CalcResult" defined or does it have a value?) You can use placeholders and the variables defined in previous macros. **Macro**: The selected macro is executed if the condition specified under "If" is true. The same job types as in step 1 (Job Information) are available for selection. However, the job types selected here as macros are completely independent of your selection in step 1. Depending on your selection, further actions and input fields are available. 	Possible operators include: • defined: Check for presence (true) • !defined: Check for non-existence (false) • &&: Logical AND • ||: Logical OR **Note:** Combined operators such as if-else or if-not are not possible. They must be split into individual statements (Condition1 = if-statement, Condition2 = else-statement). Conditional supports (restricted) Perl code. This means that a valid Perl expression must be used that results in a Boolean value. Note when making comparisons: • Texts are compared with "eq" (equals) • Numbers are compared with "=="

18.4.2.3   Create Article

Creates an article with predefined content.

Use/Job type: Ticket

Values	Hints
Parameters analogous to creating a new article. **NewArticleID:** Result variable individual name/identifier of a variable for storing the article ID (e.g. `myArticleID`). This variable can be used to reference the article in subsequent macro actions and to use its value. **DoNotSendEmail:** If activated, the system prevents the new item from being sent. For example, when linking 2 systems, it is possible to prevent the recipient system from sending the item again even though the sender system has already done so.	It is not possible to use dynamic fields. Possibility of using the TicketID / ArticleID see Variables (see page 786)

18.4.2.4   Create Report

Generates a report based on an existing report definition.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
Report definition: Based on the selected definition, the parameters of the report are included as input fields. Enter the corresponding values. The values are set by the job and the report is generated. The selection of the **output format** is also dependent on the report definition. The report delivers a result, which can also be used as a variable in follow-up actions. **Report**: Identifier of the result variable. The report provides a result that can also be used as a variable in follow-up actions.	You can find an application example under Generate reports automatically. (see page 726)

18.4.2.5   Create Ticket

Adds a new ticket with predefined values.

Use/Job type: Ticket

Values	Hints
Analogous to the ticket creation mask. Dynamic fields including their values can also be used. **Result names**: Enables the ticket ID to be stored in a variable in order to reference it in subsequent macro actions, for example (see Variables (see page 786)). ・ **NewArticleID**: individual name / identifier of a variable for storing the article ID (e.g. myArticleVariable) ・ **NewTicketID**: individual name / identifier of a variable for storing the ticket ID (e.g. myTicketVariable)	Please note: ・ Configure notification so that no new article is created. Otherwise this notification will appear as the first article. If nothing is set, the following standard values apply: ・ Lock status: "unlock" ・ Editor: the current user is set ・ Responsible: root (ID: 1) ・ Customer: 　・ if contact is found: its organization 　・ no contact found: contact as unknown contact ・ Type: Value from SysConfig key "Ticket :: Type :: Default"

18.4.2.6   Create Ticket from System Template

Creates a ticket based on a ticket template with usage context "System" (system template).

Use/Job type: Ticket

Values

The macro action returns the IDs of the created ticket and - if defined in the template - the ID of the created article.

NewArticleID: Result variable

Individual name/identifier of a variable for saving the article ID (e.g. myArticleVariable).

NewTicketID: Result variable

Individual name/identifier of a variable for saving the ticket ID (e.g. myTicketVariable)

Template: Selection of the ticket template on the basis of which the tickets are created. Only valid templates with the usage context "System" are available for selection. If necessary, create corresponding ticket templates in advance.

18.4.2.7   Delete Article

Deletes the article identified by the ArticleID. The history on the ticket receives the entry that the article has been deleted.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
ArticleID: ID of the article to be deleted. Placeholders can be used, e.g. <KIX_LAST_ArticleID>.	The deletion works in the background, i.e. for jobs with the permissions of the system user. The role permissions of the initiating user do not apply when the macro action is used.
	Tip: The action "ArticleDelete" also uses this macro action (see KIX Pro > Admin > Workflow > Actions).

18.4.2.8   Delete Ticket

Deletes tickets.

⚠ **Attention**: Action cannot be undone!

Use/Job type: Ticket

Values	Hints
none	This action ends the job. The following ticket-related actions will not be carried out (except for "Create ticket").
	It is not possible to delete specific tickets because the ticket number is not available as a filter criterion.

18.4.2.9 Execute Macro

Allows the execution of further macros within a MacroAction.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
ObjectID: Result variable The ID of the object for which the macro is to be executed (optional). **Macro**: The macro to be executed provides - depending on its type - the corresponding actions. The macros can also be of a different type and are not dependent on the selected job type.	You can find an application example under Generate reports automatically. (see page 726) <u>To differentiate:</u> "Execute Macro" can only be executed on 1 specific object. "Loop", on the other hand, can be executed on several values/arrays.

18.4.2.10 Extract Text

Extracts parts from an existing text.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints

ExtractedText: Result variable
Individual identifier of the result variable that holds the extracted text.
 	
RegEx: valid regular expression (e.g. <body>(.*?)</body>).
 	
Text: the text on which the regex is to be applied.
You can use variables and placeholders (e.g. $Report.Results:0.Content}
 	
Capture Group Names: optional naming of the individual capture groups of the regex.
If no names are given, the groups are numbered by index (1,..., n).
	Can, for example, extract the results table (or parts of it) generated in a report to make it available in the message text or in the article appendix.
	
	Use as a variable in other actions:
	• If no variable name is given, it is: `ExtractedText` .
	• Access to values of the variable is done by specifying the capture group, e.g.:
	  • `${ExtratcedText.1}` : no variable defined and also no named capture group.
	  • `${MyVariable.MyValue}:` named variable and named capture group
	
	For an application example, see Generate Reports Automatically. (see page 726)

18.4.2.11   Fetch Asset Attributes

Referenced to selected asset attributes, in order to display them, for example, in the ticket information.

Use/Job type: Ticket

Values	Hints
Asset Reference Dynamic Field: Reference to the asset whose attributes are to be retrieved. Enter the name of the dynamic field containing the ID of the asset to be used, e.g. AffectedAsset or RelatedAssets. Example: If attributes of the "affected asset" selected on the ticket are to be read, enter "AffectedAsset" here. **Attribute - Dynamic Field Mapping:** Defines which asset attribute is displayed in which dynamic field. Example: Reads out the manufacturer of the asset and displays it in the dynamic field "DFManufacturer". • Attribute: "Vendor • DynamicField: "DFManufacturer" (The value of the Dynamic Field "DFManufacturer" is displayed where it is included in the interface).	Practical examples: • KIX Start: Display asset attributes in ticket zoom view (see page 559) • KIX Pro: Displaying asset attribute in tickets (see page 863)

18.4.2.12   Loop

Generates multilevel (nested) actions

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
Values: List of values that are taken into account by the loop. Either a comma-separated list or an array generated by a placeholder (e.g. <KIX_TICKET_DynamicField_RelatedTickets_Key>). **Macro**: Possibility to configure further actions on the next lower level.	The action is also available on assets. How to use the action see Interleaving of Macro Actions with "Loop" (see page 840)

18.4.2.13 Set Contact

Puts the contact on a ticket.

Use/Job type: Ticket

Values	Hints
Contact: ID, Login or email address of the contact to be set. KIX placeholders are possible.	If the customer and contact are set at the same time, there is currently no check for • on the dependency between customer (organization) and contact • the existence of the set contact When a contact is set, its primary organization is assigned to the ticket. It is possible to set an unknown contact.

18.4.2.14 Set Dynamic Field

Sets a value in a dynamic field.

Use/Job type: Ticket

Values	Hints

Dynamic Field Name: Name of the dynamic field, e.g. "*DFMobileProcessingState*".

Dynamic Field Value: Value to be set in the dynamic field, e.g. "accepted".

Add:

The value specified under "Dynamic Field Value" is set in the dynamic field.
Values already present in the dynamic field are retained. The value is added.

The value specified under "Dynamic Field Value" is set as the new value in the dynamic field.
Values that already exist in the dynamic field are overwritten.

The value to be set depends on the type of dynamic field:

Field type	Value
Selection	"Key" value from the configuration of the dynamic field.
AssetReference	AssetID.
Date	Date in the format:YYYY-MM-DD 00:00:00
DateTime	Date-time specification in the format: YYYY-MM-DD hh:mm:ss
TicketReference	TicketID (only KIX Pro)
Checklist	JSON description of the checklist
Table	JSON description of the table
Text	The text to be set (single-line)
Textarea	The text to be set (multi-line)

18.4.2.15 Set Lock

Sets the lock status on a ticket.

Use/Job type: Ticket

Values
Lock: Exact specification of the lock status, e.g. "lock" or "unlock"

18.4.2.16 Set Organisation

Puts the organisation on a ticket.

Use/Job type: Ticket

Values	Hints
Organisation: Organisation ID or customer number. KIX placeholders are possible.	If client and contact are set at the same time, there is currently no check for • Dependency between customer (organisation) and contact • on the existence of the set organisation.

18.4.2.17 Set Owner

Places the owner on a ticket.

Use/Job type: Ticket

Values	Hints
Owner: ID or login name of the owner. KIX placeholders are possible (<KIX_CURRENT_UserID>)	There is no check as to whether the selected agent is authorized to process the ticket.

18.4.2.18   Set Priority

Sets the priority on a ticket.

Use/Job type: Ticket

Values	Hints
Priority: Exact specification of the priority	You must use the exact name of the attribute that is to be saved in the backend.

18.4.2.19   Set Responsible

Sets the responsible on a ticket.

Use/Job type: Ticket

Values	Hints
Responsible: ID or login name of the person responsible. KIX placeholders are possible.	There is no check as to whether the selected agent is authorized to process the ticket.

18.4.2.20   Set State

Sets the status on a ticket

Use/Job type: Ticket

Values	Hints

Status: Select the status to set (e.g. "new" | "open" | "pending reminder")	When setting a waiting status (status types "pending reminder" or "pending auto"), further parameters are possible. In addition, at least one indication of the waiting time must be defined:
	Pending Time Difference: Difference between the time to be set and the time of execution or relative waiting time in seconds. An entry in the form "60 * 60 * 24" is possible. Non-permissible values are interpreted as "0".
	Pending Date Time: Target time in the format "YYYY-MM-DD hh:mm:ss". Placeholders can also be used to obtain target times, e.g. from dynamic fields of the type Date or DateTime (e.g. KIX_TICKET_DynamicField_DateTime).
	Target Time: can take the value "EOB", "BOB" or <empty>. • "BOB" sets the "Begin Of Businessday" of the target time notified by PendingDiffTime or PendingDateTime. If the target time is outside the service time, the next service day is considered. • "EOB" sets the "End Of Businessday". If none of the options are used, the exact time is entered. PendingTimeOffset has priority over PendingDateTime. If a time is entered for PendingTimeOffset or PendingDateTime and EOB or BOB is also entered, the entry for Target Time has priority.

18.4.2.21 Set Team

Sets the team on a ticket.

Use/Job type: Ticket

Values	Hints
Team: Full name of the team to be set. KIX placeholders are possible.	To set subordinate teams, the following notation applies: Le `vel1::Level2::Level3` Example: 2 `nd-Level::On Site::Chemni` tz • Level 1 = 2nd level • Level 2 = On Site • Level 3 = Chemnitz

18.4.2.22 Set Title

Sets the title on a ticket.

Use/Job type: Ticket

Values	Hints
Title: The text of the title to be set. KIX placeholders are possible.	No placeholders can be used for dynamic fields of the "Checklist" and "Dropdown" types.

18.4.2.23 Set Type

Sets the type of a ticket.

Use/Job type: Ticket

Values
Type: Exact specification of the type to be set, e.g. "Incident".

18.4.2.24   Set Variable

Declares a variable for use in subsequent actions.

Use/Job type: Asset Reporting, Synchronisation, Ticket

Values	Hints
Variable: Individual identifier of the variable **Value**: Value with which the variable is to be predefined or variable identifier/placeholder for the value to be set.	The variable serves as a result variable. It can hold the following values: • the value set under "Value" • the value assigned to it when it is used, e.g. in calculations.

18.4.3 Overview of Macro Actions for KIX Pro

18.4.3.1 Account Time

Sets a value as a time booking in a ticket.

Use/Job type: Ticket

Values	Hints

Account Time:	By specifying a placeholder, you can, for example, set the value set in a dynamic field as a time entry.

• Time in minutes
• or KIX placeholders
  e.g.:
  <KIX_TICKET_DynamicField_someDynamicFieldName>
• or identifier of a result variable
  e.g. {ResultCalc}
	By specifying a result variable, the value of a previously performed calculation can be set as a time entry:

18.4.3.2   Article Attachment Add

Adds a file attachment to an existing article.

In conjunction with the Macro Action "Convert to PDF", PDF files can be generated immediately when an item is created and attached to it (e.g. printout of the work reports generated by the KIX Field Agent App).

Use/Job type: Ticket

Values	Hints
ArticleID: Defines the article to which the file attachment is to be created, e.g. the current article:<KIX_ARTICLE_ArticleID>. **Attachment**: Defines which file is attached, e.g. a PDF of a ticket or article created with the Macro Action "Convert to PDF".	

18.4.3.3   Calculate

Performs calculations of form fields, e.g:

• Calculating the values of two dynamic fields

- Determining the target times for resubmission

Can be used well in conjunction with the "Conditional" macro action.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values	Hints
Result: Result variable Identifier of the result variable (e.g. "ResultCalc"). The result variable only contains the result (value) of the calculation. The result variable can - be used and output in other macros - be set as a value in dynamic fields. **Expression:** mathematical expression of the calculation. Placeholders can be used, e.g.: DF_TotalN := <DF_TotalN> + <DF_NewValueN> **Example**: Determine the total price by multiplying the values of two Dynamic Fields ("Quantity" and "Price"); rounded to 2 decimal places. The result of the calculation is stored in the variable "CalcResult": - Result: ResultCalc - Expression: scale=2;<KIX_TICKET_DynamicField_DFQuantity>*<KIX_TICKET_DynamicField_DFPrice>	⚠️ Avoid loops: The result should not be included in the value of the calculation! Allowed operators: - Addition: + - Subtraction: - - Multiplication: * - integer division: / - Remainder for integer division: DIV + MOD - The following applies: Point calculation before dash calculation and possible bracketing. **Rounding** to decimal places is done by specifying `scale=n` (n = number of decimal places). If scale is not specified, rounding to the pre-decimal place takes place. The BC tool (Blue Bash Calculator) is used to calculate the expression. Information on this can be found at: - https://www.real-world-systems.com/docs/bc.1.html - https://www.geeksforgeeks.org/bc-command-linux-examples/ - https://www.gnu.org/software/bc/manual/html_mono/bc.html Empty results can be evaluated, for example, with the expression `'${<variable name>}'`. Use Case on the use of this macro action: Mathematical Calculations with "Calculate" (KIX Pro) (see page 851)

18.4.3.4   Convert to PDF

Description:

- Converts an item or ticket into a PDF file.
- Enables article or ticket content to be made available in automatically created tickets or in emails (e.g. creating and sending invoices or service reports).
- In combination with Webhooks, also enables the archiving of ticket processes as PDFs in a document management system.
- Can also be used in combination with the macro actions "Article Attachment Add" and "Assemble Object".
- Alternatively, PDF printing can also be done via the console (menu *System > Console*).
  To do this, use the console command: `Console::Command::Admin::HTMLToPDF::Convert` .
  This command can be used to convert an HTML document into a PDF.

Use/Job type: Ticket

Values:

- **File**: Result variable name
  Returns a file attachment as JSON containing "Content", "Filename" and "ContentType".
  For example, you can reference the variable in the "Article Attachment Add" macro action to set the generated PDF as an attachment to an article.

- **Template**: name of the template to use for the conversion (e.g. article or ticket)

- **Identifier Type**: Specifies the identifier on the basis of which the data for the print template is obtained, e.g. ArticleID.

- **Identifier**: Value that must be set based on the identifier type so that the data for the print template can be obtained,
  e.g. <KIX_ARTICLE_ArticleID> or ${RootObjectID}.

- **Filename**: Optional filename to be used instead of the default name,
  e.g.: Article_<KIX_ARTICLE_ArticleID>_<TIME_YYMMDD>.
  You can use the <TIME_YYMMDD> placeholder to add the PDF creation time to the file name.

- **Expands**: Determines which additional data should be obtained (optional),
  e.g. Dynamic field. The possible extensions depend on the template.

- **Filter**: Optional filters can be used to restrict certain extensions (format as JSON),
  e.g. if only the articles visible to the customer should be taken into account from a ticket:
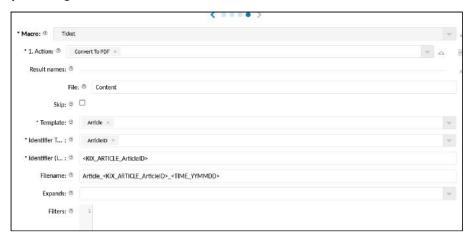
```
{
  "Article": {
      "AND": [{
        "Field": "CustomerVisible",
        "Type": "EQ",
        "Value": "1"
      }]
  }
}
```

- **Allows** (whitelist): Optional restriction of the table content (see notes opposite).
  Overwrites/manipulates what is stored in the print template for the given table.
  The use of regular expressions (RegEx) is possible.

```
{
  "InfoTableLeft": {
    "AccountedTime": "KEY",
    "DynamicField_DFChecklist": "KEY",
    "DynamicField_DFDropdown": "^regex$",
    "Created": "KEY",
    "CreateBy": "KEY",
    "ContactID": "KEY",
    "PendingTime": "KEY"
  }
}
```

- **Ignores** (Blacklist): Optional table content restriction (see notes opposite).
  Overwrites/manipulates what is stored in the print template for the given table.
  The structure is analogous to "Allows"; with the difference that the set values are ignored.

Example configuration:

Notes on the parameters "Allows" and "Ignores":

The layout of the PDF is based on an HTML print template. It defines, among other things, that the meta data of the ticket/article are listed in the page header (agent, ticket status, creation time, etc.). Tables are used to display this information. Tables are blocks of information defined in the template as type `Table`.

The parameters "Allows" and "Ignores" are used to restrict the information content of tables that are output in the PDF.

- **Allow** defines a whitelist. Only the named attributes are displayed.
- **Ignores** defines a blacklist. Attributes that made it through the whitelist will be removed. If no ignore is set, the final result corresponds to the result of the allow application.
- If both are emptied, all information from the objects is displayed.
- Specifying an empty hash clears the allow/ignore provided by the template:

```
{
   "ArticleMeta": {}
}
```

Structure of Allows/Ignores:

- In order to address the respective table, its reference ID is first required. This can be taken from the template structure.
  To do this, use the follow console command in the *System > Console* menu:
  `Console::Command::Admin::HTMLToPDF::Inspect`
  Currently, there are only the Ticket and Article templates.

- If the ID is known, a JSON can be constructed that contains the attributes to be restricted. There are two possibilities for the restriction:
  - 1.) By using the **KEY** abbreviation, the entire attribute is ignored or displayed without further checking.
  - 2.) By using a **RegEx,** the values of an attribute can be checked. If the respective value applies, the attribute is ignored or displayed.
    The RegEx is freely selectable. However, it must not contain "KEY", as "KEY" is the keyword for the entire attribute.

Syntax

```
{
    "Referenz-ID": {
        "Attribut": "KEY",
        "Attribut": "regex"
    }
}
```

Example

```
{
    "ArticleMeta": {
        "Channel": "KEY",
        "From": "kixdesk.com$"
    }
}
```

- For <u>Allows</u>, this means that the channel is displayed.
  However, From is only displayed if "kixdesk.com" is included at the end. All other possible attributes are not displayed.

- For <u>Ignores</u>, this means that the channel is directly ignored.
  However, From is only ignored if "kixdesk.com" is included at the end. All other attributes are displayeed.

18.4.3.5   Create FAQ Suggestion

Creates a new FAQ entry.

Use/Job type: Ticket

Values	Hints
NewFAQArticleID: Result variable If a variable identifier is specified, the ID of the FAQ article is stored in it for further use. **Title:** Title of the FAQ entry, KIX placeholders are possible, e.g. <KIX_FIRST_SUBJECT>. **Category:** FAQ category to which the FAQ entry is assigned. **Symptom, Cause, Solution**: Selection field. The selection determines with which ticket/article contents the FAQ entry is filled. Further fields analogous to creation mask for new FAQ entries.	Enables FAQ entries to be created from within a process. **Article of triggered event:** This selection value only works if the macro action was triggered by the event "Create article" or "Send article". Use in actions, in other events or in a time-controlled execution is currently not supported. **Example of use:** The initial job "Create FAQ Suggestion" is based on this macro action. An extension of the job by the dynamic field "Relevant Assets" with the value "KIX_TICKET_AffectedAsset_Key>" enables the linking of the FAQ with the assets from the ticket:

18.4.3.6   Create or update an asset

- Updates an existing asset (incl. new version) if an asset ID is specified and the new data represent changes to the current status.
- Creates a new asset (incl. version data) if no asset ID is specified.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values	Hints
AssetID: Result variable The ID of the asset to be updated. If no asset ID is specified, a new asset is created. **VersionID:** Result variable The ID of the new version. **Skip:** Allows temporary disabling of the action. **Asset JSON Object:** JSON editor for storing the configuration. Possible is the use of a "${XYZ}" notation for the object transfer or the specification of a JSON string which describes the object (like API). **Empty values leave the old values:** Determines how empty values are handled (analogous to importing assets). If selected, information not contained in previous versions is taken over.	The behaviour of the Macro Action corresponds to the CSV import of assets. Results of the Macro Action: AssetID and VersionID. When using <u>dates</u>, the "<u>DateTime</u>" field type must be used in the CI and in the dynamic field whose value is transferred to the asset. The field type "Date" does not work. The use of KIX placeholders for asset attributes is possible (see also Overview of KIX Placeholders (see page 750)). For more information, see Create or Update Assets (KIX Pro) (see page 854)

18.4.3.7   Create or update a contact

- Updates an existing contact if a contact ID **or** contact email address is specified and the new data represents changes to the current status.
- Creates a new contact if no contact ID is specified **and** the contact email address is unknown.

In conjunction with the add-on module Connect Database, the Macro Action enables synchronisation with an external database table/view to provide contact data in KIX. Agents can thus use the contact data from a CRM, ERP or similar without duplicate data maintenance in KIX.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values	Example: Contact (JSON Object)

<table>
<tr><td>

ContactID: Result variable for the return value / ID of the new contact or the contact to be updated.

- ContactID has a higher priority than contact email.
- An organisationID **or** organisation number can be transferred
    - only for PrimaryOrganisation
    - if an OrganisationID is specified, it is dominant
    - PrimaryOrganisationID can be determined from an optional parameter "PrimaryOrganisationNumber

Contact JSON object: JSON string of the input object. Allowed are:

- the use of a "${XYZ}" notation for the object transfer
- **or** the specification of a JSON string describing the object (like API).
- Analogous to the API, existing specifications are taken over as long as the corresponding key is not defined. As soon as a key is defined in the data, its value is also overwritten.

</td><td>

```
{
   "Email":
"johndoe@example.com",
   # oder "ContactID": 123,
   "Firstname": "John",
   "Lastname": "Doe",
   "Phone": null,
   "PrimaryOrganisationNumber":
"KNR0815",
   #OR "PrimaryOrganisationID":
456,
   "Street": "123
Downtownstreet",
   "City": "Downtown",
   "Comment": "some comment",
   "Country": "Great Britain",
   "Fax": null,
   "Mobile": null,
   "Title": null,
   "Zip": null,
   "DynamicFields": [{
      "Name": "Source",
      "Value": "created by
ticket
<KIX_TICKET_TicketNumber>"
   }]
}
```

</td></tr>
</table>

18.4.3.8 Create or update an organisation

- Updates an existing organisation if an organisation ID **or** organisation number is specified and the new data represent changes to the current status.
- Creates a new organisation if no organisation ID is specified **and** the organisation number is unknown

In conjunction with the add-on module Connect Database, the Macro Action enables synchronisation with an external database table/view to provide organisational data in KIX. Agents can thus use the organisational data from a CRM, ERP or similar without duplicate data maintenance in KIX.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values	Example: Organisation (JSON Object)

OrganisationID: Result variable for the return value / ID of the new or to be updated organisation.

- OrgID has a higher priority than organisation number.

Organisation JSON object: JSON string of the input object. Allowed are:

- the use of a "${XYZ}" notation for the object transfer
- **or** the specification of a JSON string describing the object (like API).
- Analogous to the API, existing specifications are taken over as long as the corresponding key is not defined. As soon as a key is defined in the data, its value is also overwritten.

```
{
  "City": "London",
  "Comment": "",
  "Country": "United Kingdom",
  "Name": "Harmony Shoal Ltd.",
  "Number": "HRMSHL",
  # OR "OrganisationID": 23
  "Street": "245 Walworth Rd.",
  "Url": "http://www.harmony-
shoal-example.co.uk",
  "Zip": "SE17 2AL",
  "DynamicFields": [{
      "Name": "Type",
      "Value": [
          "supplier/partner
(external)",
          "customer"
      ]
  }]
}
```

18.4.3.9   History Cleanup

Deletes the history of tickets for selected agents or after previous ticket actions.

Use/Job type: Ticket

Values
Agent: ID or login name of the agent
Action: Name of the action (RegEx possible)
New Comment: You can add a comment to the history entry.

18.4.3.10   LDAP to Contact

Synchronizes the contact and user data in KIX Pro with data from the LDAP / Active Directory.

Use/Job type: Synchronisation

Values
s. Admin Manual KIX Pro: LDAP/AD-Synchronisation

18.4.3.11   Set Article Email Attributes

Set email attributes (Bcc, Cc, To, From).

Use/Job type: Ticket

Values	Hints
Valid email addresses. KIX placeholders are possible.	For anonymization

18.4.3.12   Set Asset Name

Changes the name of an asset without generating a new version. The current version of the asset is updated (name and time of change).

For example, to automatically generate the asset name based on attribute values, such as:

- Name of the computer - consisting of manufacturer, user and IP address.
- Name of a room - consisting of name of parent location and room type

Use/Job type: Asset

Values	Hints
Skip: Allows you to temporarily disable the action. **Asset Name:** The new name of the asset. KIX placeholders can be used, e.g.: - <KIX_ASSET_Name - <KIX_ASSET_ParentAttribute_0_SubAttribute_0>. Further placeholders: see also Overview of placeholders > section "Asset-related placeholder areas (see page 750) ".	The Macro Action can only be used in jobs. ⚠️ **Important:** Restrict the assets to be changed using **filters** (step 3 in the job configuration)! Otherwise **all** assets will be renamed.

18.4.3.13   Set Asset Number

Replaces the asset numbers assigned during import by KIX with own identifiers.

Use/Job type: Asset

Values	Hints
Asset number: The new identifier of the asset. KIX placeholders can be used, e.g: • <KIX_ASSET_Name> • <KIX_ASSET_ParentAttribute_0_SubAttribute_0>	Use Case: see: Admin Manual KIX Pro: Change asset numbers during import

18.4.3.14 Set CreateBy and ChangeBy

Puts a specific agent in a ticket.

The agent is stored in the fields "Created by" and "Changed by".

Use/Job type: Ticket

Values	Hints
Agent: Agent ID or login name. KIX placeholders are possible.	For anonymization

18.4.3.15 Set CreateBy and ChangeBy for articles

Puts a specific agent in an article.

The agent is stored in the fields "Created by" and "Changed by"

Use/Job type: Ticket

Values	Hints
Agent: Agent ID or login name. KIX placeholders are possible.	For anonymization

18.4.3.16 Set Fulfillment Time

Sets the fulfillment time for an SLA criterion.

Use/Job type: Ticket

Values	Hints
SLA criterion: ID or exact name of the SLA **Force if not null**: 0 | 1	For SLA criteria Value "1" forces the fulfillment time to be overwritten.

18.4.3.17 Set incident state of affected asset

Sets the incident status on assets.

Use/Job type: Ticket

Values
Asset / AssetID: Number or ID of the asset. Multiple assets can be specified separated by commas. KIX placeholders are possible. **Force**: 0 | 1 The incident status "Incident" can only be overridden if there are no open incident tickets for the asset. Set "1" to force the status regardless of the situation. **Incident status**: Enter the name of the incident status to be applied, e.g. "Incident".

18.4.3.18   Set SLA

Sets an SLA criterion in the ticket.

Leave the field blank if you want to remove an SLA from the ticket.

Use/Job type: Ticket

Values
SLA: ID or name of the SLA. KIX placeholders are possible.

18.4.3.19   Ticket Merge

Combines 2 tickets, e.g. to attach a misdirected email (source) to a ticket (destination).

Use/Job type: Ticket

Values	Hints
Target ticket: The ticket number of the tartget ticket or the name of the dynamic field containing the ID of the target ticket. **Ticket Attributes:** Comma-separated list of ticket attributes. The list contains the attributes of the source ticket that will be written to the target ticket. 🛈 Translations of the attribute names are not possible. **Dynamic fields:** Comma-separated list of the names of the dynamic fields that will be transferred to the target ticket. 🛈 Translations of the names are not possible. **Create values for dynamic fields:** 🔵 Overwrites the values of the listed dynamic fields in the destination ticket with the values of the source ticket. ⚪ The value of a dynamic field of the source ticket will only be set on the destination ticket if such a value does not yet exist there.	Further information: • Job "TicketMerge" (see Jobs - Advanced Functions) • Merge action (see Actions)

18.4.3.20   Wait

Pauses the execution of the job for the specified time.

Can be used, for example, if a webhook or Baramundi request delivers a timeout and the job should repeat the request after a specified waiting time.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values
Wait: Specify how many seconds the execution of the job should pause.   The job waits the specified number of seconds before continuing with the execution of the subsequent macro actions.

18.4.3.21   Webhook

Sends a predefined event message to an external system (ERP, server, etc.)
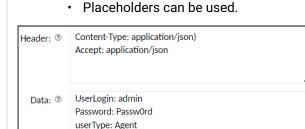
For example, sending the ticket number and the sum of the booked expenses to an ERP system.

Use/Job type: Ticket

Values	Hints

URL: character string (endpoint URL for an API call)
 	
Method: Transmission method (protocol)
An asynchronous GET / POST request is sent.
 	
• GET: Converts the key-value pairs into URL parameters and adds them to the named URL. The separation is done with "?", or with a semicolon, if the URL has a "?" included.
• POST: Sends the key-value pairs as a JSON request (encoded)
 	
Header: Request header
 	
    • The specified values are used as HTTP headers
    • One header is to be specified per line as a key-value pair.
 	
Data: Request Data
 	
    • Data that the request should transfer.
    • A key-value pair must be specified per line.
    • Placeholders can be used.
 	
Header: ⑦   Content-Type: application/json)
            Accept: application/json
 	
Data: ⑦   UserLogin: admin
          Password: Passw0rd
          userType: Agent
          TicketID: <KIX_TICKET_TicketID>
	• Is available for ticket and article actions.
	• The basic SysConfig settings (proxy, timeout, etc.) of WebUserAgent are used as a basis.
	• There is no processing of results ("fire & forget").
	• No structures and value mappings can be defined.
	• No attachments can be sent.
	• Changes to KIX objects are not possible.
	• The response status is documented in the ticket history ("Webhook called - response xxx received")

18.4.3.22 While

Repeats the execution of the macro as long as a configured condition is fulfilled.

<u>Usage example:</u>
A webhook request delivers that further data can be retrieved (paging for APIs). The job should trigger further requests until all data has been retrieved and processed. A waiting time can be configured between the requests to prevent the appearance of a DoS attack in the case of requests that follow in quick succession.

Use/Job type: Asset, Reporting, Synchronisation, Ticket

Values
While: Insert the condition that must be fulfilled for the macro action specified under "Macro" to be executed. The macro action is executed as long as the condition specified here applies. **Macro**: Configure the macro action to be executed here. Select the object type and the actions to be executed and configure them as required. **Maximum Iteration**: To avoid continuous loops, you can specify the maximum number of runs. If no number is specified, the process is cancelled after 1000 runs.

18.4.4 Save ticket ID and article ID in one variable

Some macro actions provide one or more return values (e.g. the ID of the created ticket or article) as a result of the execution. You can save these return values as result variables in order to use them in subsequent macro actions.

For example, the configurations of the macro actions "Create ticket" and "Create article" contain the input fields "NewArticleID" and "NewTicketID". Enter a freely definable identifier in these fields. The identifier is the name of the result variable in which the return value (e.g. the ID) is stored. By calling up the identifier, you can reference the return value stored in the variable, e.g. to enter the ID

- in a dynamic field
- in the ticket subject/title
- in the article text

In this way, for example, the ID of a newly created ticket can be stored temporarily in order to save the ticket as a child ticket on the triggering ticket based on its ID.

The variable is integrated by `${identifier}`. If your variable identifier is, for example, "TicketIDTechnik", this will be integrated with `${TicketIDTechnik}`.
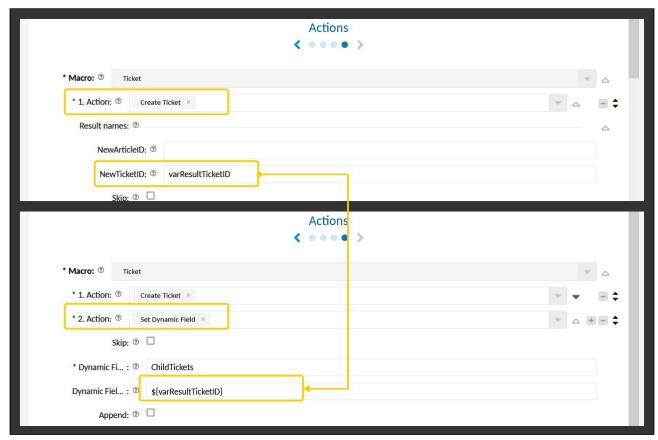
Fig .: Use of an individual variable to output the ticket ID in a dynamic field.

If the job is triggered or the action is applied in KIX Pro, the variable is read out and replaced by the real value of the ID.

For more information on the use of variables, see: Variables (see page 786)

> ⓘ **Info**
>
> If the ID is saved in a dynamic field, you can define whether the ID replaces existing IDs or whether it is added. This is done by activating/deactivating the "Append" checkbox in the job configuration (see Action in "Set dynamic field").
> If the job is triggered or the action is applied in KIX Pro, the variable is read out and replaced by the real value of the ID.

18.4.5 Interleaving of Macro Actions with "Loop"

The macro action "Loop" can be used to define a referenced macro and to execute it for <u>different</u> objects. For example, to pass on the workaround documented in a ticket to all child tickets or to make changes to a parent ticket also to all child tickets (e.g. "area disruption"). To differentiate: The macro action "Execute macro" can only be executed on 1 specific object.

With the macro action "Loop", actions can be nested as desired. In this way, complex, multi-stage workflows can be mapped and references between several tickets that refer to each other can be used for automatisms. However, be careful <u>not to create endless loops</u>!

- 1 job can consist of n macros (also the pre and post actions of the actions in KIX Pro)
- 1 macro can consist of n actions
- 1 action can reference n macros.

The following actions can be performed on n tickets that are referenced using <KIX_TICKET_DynamicField_ParentTickets> or <KIX_TICKET_DynamicField_ChildTickets> or dynamic fields of the "TicketReference" type:

- Set dynamic field
- to write an article
- Make contact
- Set customer (organization)
- Set agent
- Set priority
- Put those responsible
- Set Status
- Team
- Set title
- Type
- Set lock status.
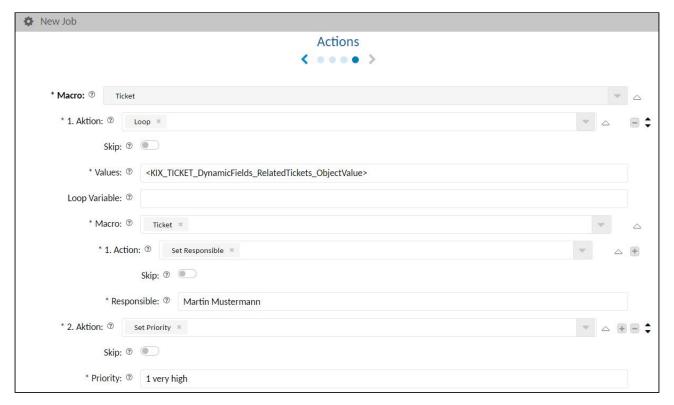
All other KIX_TICKET placeholders are replaced with details of the ticket that triggered the ticket.

Example:

Job: Set the person responsible "Martin Mustermann" (username) and the priority to "1 very high" on all "RelatedTickets":

The use of " `_Key` " instead of " `_ObjectValue` " is possible until further notice. However, we recommend the use of " `_ObjectValue` " (see also Overview of KIX Placeholders).

18.4.6 Time Calculations with Macro Actions

If form field values are based on dynamic fields of the type "Date" or "DateTime", target times can be calculated automatically with a formula. In this way, for example, dates for resubmissions or for the next rule check of an asset can be determined and further used.

The execution of these calculations is possible with jobs as well as with actions (only KIX Pro) and is carried out using KIX time placeholders and variable filters (see also Variables (see page 786)).

If necessary, asset class definitions must be supplemented with further attributes (form fields) so that these values can be used for a calculation, e.g. information on the maintenance interval. (see also: Class Definition (see page 126)).

18.4.6.1 Scenario

In the following example, the next maintenance date of an asset is to be determined. This is based on the calculation of the maintenance interval stored for the asset concerned and the date of the last maintenance noted on the ticket (date of last maintenance + maintenance interval = next maintenance date).

Status*	🖶 open ✖			⊗ ▾
Plan Begin	TT . MM . JJJJ	🗓	-- : --	⊖
Plan End	TT . MM . JJJJ	🗓	-- : --	⊖
Last Maintenance	17 . 01 . 2023	🗓	14 : 30	
Maintenance Period (months)	6			
Next Maintenance	17 . 07 . 2023	🗓	08 : 00	
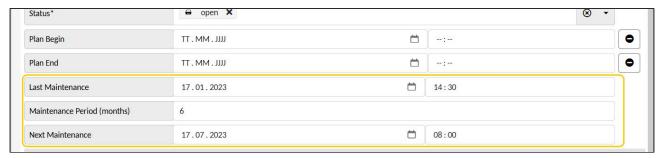
Fig.: Information on the maintenance interval in the ticket

The following steps are required for this:

- Create 3 dynamic fields (date of last maintenance, maintenance interval, date of next maintenance).
- Integration of the dynamic fields into the dialogues "new ticket" and "edit ticket
- Extension of the asset class definition by the attribute (form field) "Maintenance interval | MaintenancePeriod"
- Deposit of a pattern for the new input field (optional)
- Job configuration for reading out the maintenance interval
- Job configuration for calculating the next maintenance date

18.4.6.2 Procedure

1. Create Dynamic Fields

For the implementation of the example, 3 dynamic fields are required. They are created in the menu *System > Dynamic Fields* with the following basic configuration:

Dynamic Field Name	Function in the ticket	Label	Field type	Object type	Count (min.| max.| def.)	Other parameters
DFMaintenanceDateTime	Date of last maintenance	Last Maintenance	Date/Time	Ticket	1 | 1 | 1	
DFMaintenancePeriod	Maintenance period in months	Maintenance Period (months)	Text	Ticket	1 | 1 | 1	RegEx: `^[0-9]*$` RegEx ErrorMessage: "Please enter only numerical values"
DFNextMaintenance	Date of the next maintenance appointment	Next Maintenance	Date/Time	Ticket	1 | 1 | 1	Date Restriction: DisablePastDates

All other configuration parameters are optional.
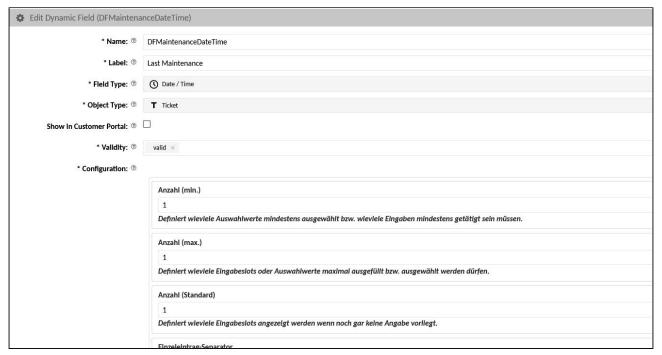
Fig.: Configuration of dynamic field "DFMaintenanceDateTime"

2. Integrate Dynamic Fields

In order for the maintenance information to be available on the ticket, the dynamic fields must be provided in the dialogues "New ticket" and "Edit ticket". Alternatively, you can also provide them in other templates or actions.

- **KIX Start**: Include the Dynamic Fields in the SysConfig keys "*ticket-new-form-group-data*" (new ticket dialogue) and "*ticket-edit-form-group-data*" (edit ticket dialogue). Proceed as described in Integrating a Dynamic Field (see page 423) .
- **KIX Pro:**
    - Dialog "New ticket": Navigate to the menu *Workflow > Template*s. Open the template "Default - New Ticket Template" and add the previously created 3 dynamic fields under "Input fields".
    - Dialogue "Edit ticket": Navigate to the menu *Workflow > Actions*. Open the action "Ticket Edit" and add the 3 previously created dynamic fields under "Input fields".
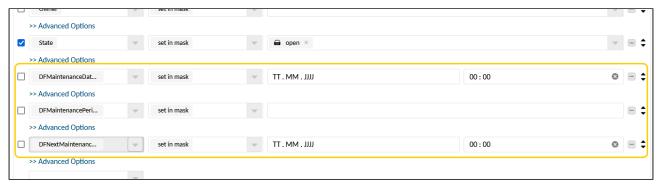
Fig.: Integration of the dynamic fields into the dialogue "New ticket" (KIX Pro: Template "Default - New Ticket Template")

3. Extension of the asset class

The dialogues for creating new assets do not contain an input field for entering the maintenance interval by default. To integrate this into the dialogue, the corresponding asset class must be extended by an additional input field. This is done by modifying the class definition.

For this example, the asset class "Computer" is extended by the field "MaintenancePeriod". Then, when creating/editing an asset of the class "Computer", the maintenance interval can be stored in months.
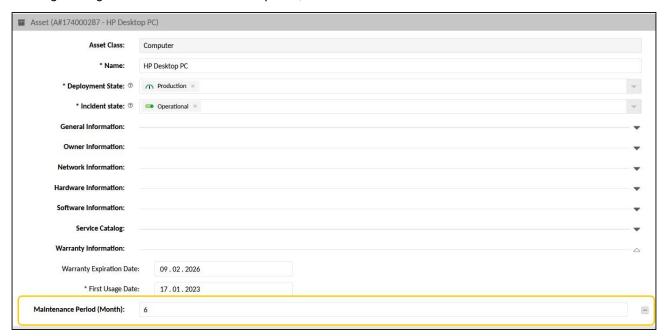
Fig.: Extended asset class "Computer"

The modification of the asset class is done in the menu *Assets > Asset Classes*. Open the class "Computer" to edit it. Alternatively, you can select another asset class.

Insert the following code in the class definition. Place it in the desired position. The placement in the code defines the placement of the input field in the form.

Alternatively, you can name the input field "MaintenancePeriod" differently, but you must consider this in the further course of the example.

For a functioning calculation, it must be ensured that only numbers can be noted in the input field. This is done by specifying a regular expression (RegEx).

```
{
    'CountDefault' => 0,
    'CountMax' => 1,
    'CountMin' => 0,
    'CustomerVisible' => 0,
    'Input' => {
      'Required' => 0,
      'Type' => 'Text',
      'RegEx' => '^[0-9]*$',
          'RegExErrorMessage' => 'Please enter numeric values',
    },
    'Key' => 'MaintenancePeriod',
    'Name' => 'MaintenancePeriod',
    'Searchable' => 1
}
```
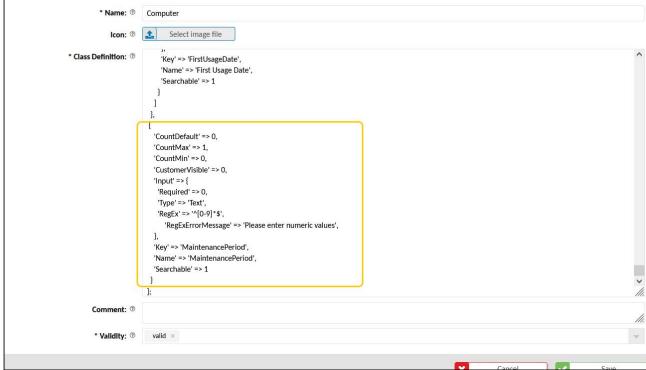

Fig.: Extension of asset class "Computer" by the input field "MaintenancePeriod" (at the end of the class definition)

Further information on asset classes and their modification can be found in the Admin Manual of KIX Start:
Asset Classes (see page 120) .

4. Create pattern

The name of the input field (MaintenancePeriod) is freely selectable. You may have chosen a different name.
In order for KIX to be able to translate this name in the interfaces, a pattern can be stored in KIX. This is done
in the menu *Internationalisation > Translations*. Create a new pattern.

Fig.: Pattern for the "MaintenancePeriod" input field

5. Job configuration for reading out the maintenance interval

If an affected asset is selected on the ticket, its maintenance interval should be read out and stored on the
ticket. This is done automatically by a job.

The job reads the value of the attribute "MaintenancePeriod" from the affected asset and sets this value into
the dynamic field "DFMaintenancePeriod", which was already integrated into the interfaces in step 2, when
the ticket is saved.
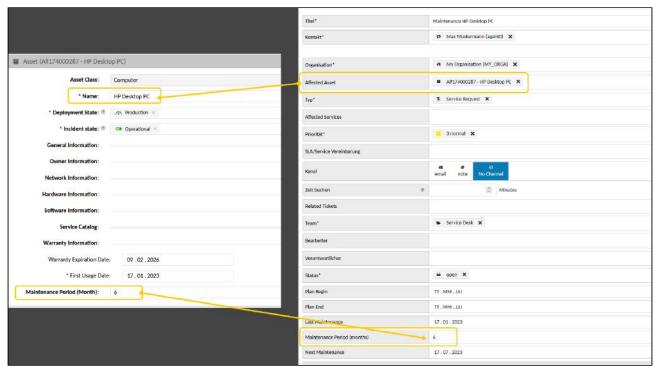
Fig.: Entering the maintenance interval of the affected asset in the ticket

The job is created and configured in the *Automation > Jobs* menu. Configure the job as follows:

- Job Information
    - Type of Job: Ticket
    - Name and Comment: any
    - Validity: valid
- Execution Plan
    - Time Based Execution: n. A.
    - Event Based Execution: TicketDynamicFieldUpdate_AffectedAsset, TicketCreate (optionally other events)
    - Filter: n. A.
    - Actions
        - 1. Action: Fetch Asset Attributes
        - Asset Reference Dynamic Field: AffectedAsset
        - Force: checkmark if the value should always be updated when the job is executed (optional).
        - Attribute - Dynamic Field Mapping: Attribute: MaintenancePeriod
        - Dynamic Field: DFMaintenancePeriod

6. Job configuration for calculating the next maintenance date.

If the date of the last maintenance is specified on the ticket, the date of the next maintenance can be calculated in connection with the maintenance interval. This is also done automatically by a job.

Fig.: Calculation of the next maintenance date

The job reads the date of the last maintenance stored on the ticket and saves it in the variable "varCurrentMaintenance" (1st action). If no date is specified for the last maintenance, the job automatically sets the current time as the date of the last maintenance so that a calculation is possible (2nd action).

Furthermore, the job reads the maintenance interval of the asset concerned from the dynamic field "DFMaintenanceDateTime" and stores this value in the variable "varMaintenancePeriod". (3rd action). If the maintenance interval is assigned a value, the calculation can be performed (4th action). The result of the calculation is written into the dynamic field "DFNextMaintenance" and thus made available on the ticket (action 4.1).

The following formula is used for the calculation:

```
${varCurrentMaintenance|DateUtil.bob|DateUtil.calc(+${varMaintenancePeriod}
M)|DateUtil.unixTime|DateUtil.timeStamp}
```

- From the date of the last maintenance, the start of the business day (BOB) is extracted using the filter " `DateUtil.bob` " (8:00 am - depending on the calendar).
- Using the filter " `DateUtil.calc` ", this value is added to the number of months.
- The number of months corresponds to the value of the maintenance interval, which is stored in the variable " `${varMaintenancePeriod}` ". Thus, the expression (`+${varMaintenancePeriod}M`) corresponds to the expression (`+6M`), i.e. 6 months.
- The filters " `DateUtil.unixTim` e" and " `DateUtil.timeStamp` " convert the result into a Unix/ time stamp so that the date can be written into the dynamic field "DFNextMaintenance" (date of the next maintenance) and made available on the ticket.

The job configuration in detail:

- Job Information
    - Type of Job: Ticket

- Name and Comment: any
- Validity: valid

- Execution Plan
    - Time Based Execution: n. A.
    - Event Based Execution: TicketDynamicFieldUpdate_AffectedAsset, TicketDynamicFieldUpdate_DFMaintenanceDateTime (optionally further events)

- Filter: none

- Actions:
    - 1. Action: Set Variable
        - Variable: varCurrentMaintenance
        - Value: <KIX_TICKET_DynamicField_DFMaintenanceDateTime_ObjectValue>
    - 2. Action: Conditional
        - If: `!defined ${varCurrentMaintenance}`
        - Macro: Ticket
            - 1. Action: Set Dynamic Field
                - Dynamic Field Name: DFMaintenanceDateTime
                - Dynamic Field Value: <KIX_NOW>
    - 3. Action: Set Variable
        - Variable: varMaintenancePeriod
        - Value: <KIX_TICKET_DynamicField_DFMaintenancePeriod>
    - 4. Action: Conditional
        - If: `defined ${varMaintenancePeriod}`
        - Macro: Ticket
            - 1. Action: Set Dynamic Field
                - Dynamic Field Name: DFNextMaintenance
                Dynamic Field Value: `${varCurrentMaintenance|DateUtil.bob| DateUtil.calc(+${varMaintenancePeriod}M)| DateUtil.unixTime|DateUtil.timeStamp}`

18.4.7 Mathematical Calculations with "Calculate" (KIX Pro)

The "Calculate" macro action is only available in KIX Pro. It allows the automatic calculation of form field values, e.g. to calculate prices, sums and the like from individual values. The point (e.g. 3.147) is used as the decimal separator.

Note: Variables and placeholders are replaced 1:1 without validation, BEFORE the job is executed. If a variable does not contain a value or is undefined due to a typing error, it is calculated with "null", which leads to an incorrect result. Therefore, make sure to restrict dynamic fields of the type "Text" to the input of numeric characters via RegEx.

18.4.7.1 Scenario

To calculate the total price, there are two form fields on a ticket for entering the (individual) price and quantity. In a third field in the form, the total price should be calculated automatically. If the calculated total price exceeds a threshold value, the ticket is presented to management for a decision.

Proceed:

1. Creation of 3 dynamic fields (DF price, DF amount, DF total price) with the following configuration:
   - Field type: text
   - Object type: ticket
   - CountMin, CountMax, CountDefault: 1 each
   - RegEx List: `^[0-9]*$` (only allows numbers and digits in the input field)
   - RegExErrorMessage: "Only numbers and digits allowed"

2. Integration of the dynamic fields into the GUI
   - DFQuantity and DFPrice
     - e.g. in the dialog "Ticket new" (see chapter Integrating a Dynamic Field)
     - Menu: *System > SysConfig > Key: "ticket-new-form-group-data"*
     - e.g. in a dialog-based ticket action.
   - DFPriceTotal in the ticket details .
3. Configuration of a job to carry out the calculation and use of the total price for automatic ticket routing.
   The configuration of the job could be as follows:
   - Job information
     - Job Type: Ticket
     - Name: Name of the job
     - Validity: valid
   - Execution plan
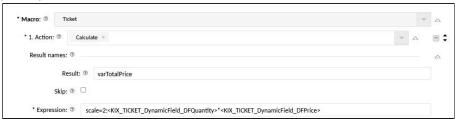
- Events: TicketDynamicFieldUpdate_DFPrice, TicketDynamicFieldUpdate_DFQuantity
- Filters: none or as required
- Actions
  - 1st action: Calculate

    (= Execution of the calculation)
    - Result: Identifier of the result variable (e.g. varTotalPrice)
    - Expression:

      `scale=2;<KIX_TICKET_DynamicField_DFQuantity>*<KIX_TICKET_DynamicField_DFPrice>`

      (= Multiplication of the values in the dynamic fields, rounded to 2 decimal places)

  - 2nd action: conditional
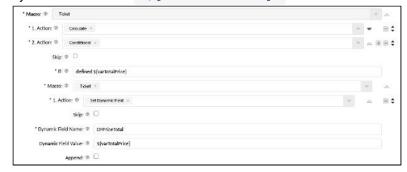
    (= Condition for the following processing: calculation provided a result)
    - If: Specification of the condition (e.g. `defined ${varTotalPrice}` or `${varTotalPrice} && ${varTotalPrice}> 0)`
    - Macro: ticket

      (= If condition is met: put total price in DFPriceTotal)
      - 1st action: Set dynamic field
        - Dynamic Field Name: DFPriceTotal
        - Dynamic Field Value: `${varTotalPrice}`

      - 2nd action: conditional

        (= Condition for the following processing: total price> 10000)
        - If: `${varTotalPrice} > 10000`
        - Macro: ticket

          (= If total price> 10,000, then pass the ticket to "Management", change status and priority)
          - 1st action: set team

- Team: management
  - 2. Action: Set State
    - Status: Approval required (create status (see page 614) beforehand if necessary)
  - 3. Action: Set priority
    - Priority: 1 very high

> ⓘ **Note on error handling**
>
> If a specific calculation cannot be carried out, it is recommended to obtain the placeholders / fields used in the calculation or their values from the specific ticket history and to carry out the calculation using bc on the command line. In this way it can be narrowed down whether the calculation itself or the values used in it are causing the error.

18.4.8 Create or Update Assets (KIX Pro)

The macro action "Create or update an asset | Create or Update an Asset" is only available in KIX Pro. You can use the Macro Action, for example,

- to use the processed return of a webhook call to create or update an asset
- to update the referenced asset (s) with information from the ticket (or absolute values) when a ticket status is reached
- to apply the automatism to any dynamic fields of the type "AssetReference" for certain events or time-controlled.

Specific use cases would be e.g. B .:

- **With KIXConnect**: A periodic request to a web service interface returns information about an asset. This data should be able to be used to update the entry in KIX asset management (=> Baramundi inventory; => IDoIT inventory; etc.)
- **Without KIXConnect:**
    - The delivery of N assets on a ticket is documented in a dynamic field of the Date / Time type. The day of commissioning is to be set with this value for the assets concerned and the usage status is to be set to "Production".
    - Time units are booked on a ticket for a service contract. The budget used is recorded in the asset service contract. When the ticket is closed, the budget used for the asset should be increased by the sum of the time spent on the ticket.
    - The mileage of a vehicle is recorded on a ticket. The vehicle is recorded as an asset and includes the last odometer reading. This should be automatically written from the ticket to the asset.
    - The old device and the new device are entered on a ticket for exchanging a workstation computer. When the processing status "exchanged" is reached, the user assignment of the old device is removed and the user is entered on the replacement device. The user is the customer contact of the ticket.

18.4.8.1 Basics of this Macro Action

The macro action expects a configuration in JSON format. The configuration defines which asset is addressed by which asset class and which attributes of this asset are assigned which values.

For example, a PC with Asset ID 4 in the Computer asset class. The incident status and the usage status can be set on this PC via the macro action, as well as selected attribute values on the asset, e.g. the date of commissioning.

The basic structure of the JSON is based on a POST request for ConfigItems / Assets (without images).

All other attributes of the MacroAction to be specified can be found in the Overview of Macro Actions for KIX Pro .

Basic configuration	Attribute	Description	Hint

{
  "AssetID": 1,
  "ClassID": 4,
  "Version": {
    "Name":
"some name",
 	
"DeplStateID":
16,
 	
"InciStateID":
1,
    "Data": {}
  }
}
	AssetID	Asset ID of the asset to be updated
		The specification determines whether the asset is updated or created.
			• Update: if Asset ID is given
			• Create: if no Asset ID is given

	ClassID	Asset class ID, e.g. "4" for the computer asset class.	• Mandatory at Create • Optional with Update
	Name	Name of the Assets	• Mandatory at Create • Optional with Update

	DeplStateID	Usage status ID	• If not specified:
			  • with Create: Default to production
			  • with Update: values of the previous version

	InciStateID	Incident status ID	• If not specified:
			  • with Create: Default to Operational
			  • with update: values of the previous version

	Data {...}	class-specific attributes	see below section "Data attributes"

ℹ You can find the ID of the asset or the asset class in the address line of your browser.

18.4.8.2   Data attributes

The attributes specified under Data {...} address the asset attributes in the class definition. This defines which values are set for which attribute of the class definition (see configuration example (see page 859) below).

The information can be defined in 3 ways:

Way	Description	Example
| 1 | Direct (value after the attribute key) | ```json
{
  "Version": {
    "Data": {
      "SomeAttribute": "Value"
    }
  }
}
``` |
| 2 | Directly (value list (array) after the attribute key) | ```json
{
  "Version": {
    "Data": {
      "SomeAttribute": [
        "Value1",
        "Value2"
      ]
    }
  }
}
``` |

Way	Description	Example
3	As an object list (array of "hashes") after the attribute key. Whereby the attribute key itself is contained and this in turn has to receive the value directly. This variant must be used if you want to specify child / sub-attributes. The child / sub-attributes can then use one of the variants.	```{ "Version": { "Data": { "SomeAttribute": [{ "SomeAttribut": "Value1", "SomeChildAttribute": "ChildValue" }, { "SomeAttribut": "Value2" }] } } }```

18.4.8.3 General information

- It is possible to combine the aforementioned variants 2 and 3: If in variant 3 an object / hash is specified for all values (directly in the list, without children), a "Value2" can be used instead of the hash for the 2nd value. - as in the 2nd variant - can be specified.
- Attachments are possible. As usual in the API, the value is: Object with "Filename", "ContentType" and base64-encoded "Content".
- The content of data does not have to include the complete version definition; a single attribute is sufficient. Attribute values must - if necessary - be IDs (as with POST). There is no "resolution" of the values (e.g. for GeneralCatalog attributes).
- When using dates, the "DateTime" field type must be used in the CI and in the dynamic field whose value is transferred to the asset. The field type "Date" does not work.
- Attributes not specified are ignored; their values remain unchanged.
- The code expects at least one information in the "Version" object (Name, DeplState or InciState). The "Data" entries are optional.

- The use of KIX placeholders is possible anywhere. Placeholders that cannot be resolved are replaced with an empty string.
- Instead of a "JSON-Object" string, a "Result" placeholder can also be used (`$ {SomeAssembledObject}`). Although this is marked as incorrect by the editor (since there is no valid JSON), it is still accepted.
- The result variables are set:
  - "AssetID" - ID of the created / updated asset
  - "VersionID" - ID of the new version

18.4.8.4   Dealing with blank values

The option "Empty values leave the old values" determines how empty attribute values (null, empty strings) should be handled (analogous to CSV import of assets). Attribute values that are not specified are generally ignored, i. H. they have no effect on old values. By default, the checkbox in the front end is checked (active).

Example:

- Variant A: The asset already has 2 values in the attribute
- Variant B: The asset already has 5 values in the attribute

The following JSON is used:

```
{
    "SomeAttribute": [
        null,
        null,
        null,
        "NewValue"
    ]
}
```

☑ **If active:** New values are "appended" or overwritten at the same value position.

- in variant A: "NewValue" is stored as the 3rd value. The 1st and 2nd values are retained because the blank value leaves the old values. There was no third value, so this space is free (avoid empty value spaces).
- in variant B: value 4 is overwritten with the new value, the rest remains.

☐ **If inactive:** New values overwrite existing values; Empty values (null, empty string) overwrite / remove existing values.

- in variant A: The new value is the only one available. 1st and 2nd value have been emptied and are therefore free. The new value moves to the first free place.
- in variant B: The attribute now has 2 values: the new value in position 1 and the value in position 5 is now in position 2 (moved up).

Special features of the behavior when "inactive" and sub / child attributes are available:

- The behavior to positions that have become free only applies if the attribute has no sub-attributes (values). Only the value is removed, not its subs / children. On the above Example are e.g. B. Position 2 is given values for its sub-attributes.
    - Since position 1 loses its value and has no subs, this position is free. This moves the subs of position 2 up to 1 in order to avoid empty values. Position 2 has thus become free, as the value has also been removed (null).
    - The new value is set accordingly at position 2.
    - At B, the value 5 also moves to position 3.
- The CSV import of assets is basically the same (however, the subs are set to "empty"). Would the above Example also set empty values for the sub-attributes, one would bypass or remove the "sub-restriction":

```
{
    "SomeAttribute": [
        null,
        {
            "SomeAttribute": null,
            "SomeChildAttribute": [
                null,
                null,
                ...
            ],
        },
        null,
        "NewValue"
    ]
}
```

18.4.8.5 Configuration example

The following configuration example creates a new asset of the class Computer with the following parameters:

- Title of the ticket (type ticket job) as a note
- an image as an attachment
- a dynamic field (date field for commissioning)
- since the asset ID is not set in the input field of the macro action, "ClassID" and "Name" must be specified in the version
- "DeplStateID" is included ("Production")
- "InciStateID" is not included (the default is set to "Operational").

```
{
  "ClassID": 4,
  "Version": {
    "Name": "Created by Job",
    "DeplStateID": 16,
    "Data": {
      "SectionGeneral": [
        {
          "Note": "<KIX_TICKET_Title> and some text",
          "Attachment": [
            {
              "ContentType": "image/jpeg",
              "Filename": "test.jpg",
              "Content": "<base64-String des Bildes> "
            }
          ]
        }
      ],
      "SectionWarranty": [
        {
          "WarrantyExpirationDate": "2021-08-11 12:00:42",
          "FirstUsageDate": "<KIX_TICKET_DynamicField_PlanBegin_ObjectValue>"
        }
      ]
    }
  }
}
```

The created computer will be adapted in the further course (pay attention to the correct "AssetID"!).

- The "AssetID" must be specified in the input field of the Macro Action, otherwise a Create is attempted. However, this would fail because "ClassID" is missing (irrelevant for update).
- The name of the asset is changed
- The incident status is changed ("Fault")
- "Data" attributes are not specified. the existing values therefore remain unaffected.

```
{
  "AssetID": 42,
  "Version": {
    "Name": "Created by Job – updated",
    "InciStateID": 3
  }
}
```

The created computer is adjusted again:

- Only one FQDN is stored, everything else remains unaffected.
  - No further attributes necessary.

```json
{
  "AssetID": 42,
  "Version": {
    "Data": {
      "SectionNetwork": [
        {
          "FQDN": "some.test-computer.com",
        }
      ]
    }
  }
}
```

The created computer is adjusted for the 3rd time:

- Only one additional FQDN is stored, the previous one should be retained.
  - "EmptyValuesLeaveTheOldValues" is active (checked).

```json
{
  "AssetID": 42,
  "Version": {
    "Data": {
      "SectionNetwork": [
        {
          "FQDN": [
            null,
            "another.test-computer.com"
          ]
        }
      ]
    }
  }
}
```

The created computer is adjusted for the 4th time.

- The 2nd FQDN is overwritten and a 3rd FQDN is added:
  - "EmptyValuesLeaveTheOldValues" is active (checked)

```json
{
  "AssetID": 42,
  "Version": {
    "Data": {
      "SectionNetwork": [
        {
          "FQDN": [
            null,
            "changed-another.test-computer.com",
```

```
                    "yet-another.test-computer.com"
                ]
            }
        ]
    }
  }
}
```

The created computer is adjusted for the 5th time:

- The old FQDN will be replaced by a new one. Then there is only one.
  - If there are already more than 3 FQDNs, the list should contain more "null" elements. But we know there are only 3 FQDN. So 2 are enough, which empty the 1st and 2nd FQDN and replace the 3rd. This in turn moves up because the places in front of it have become free.
  - "EmptyValuesLeaveTheOldValues" is inactive (not checked). Thus, empty values replace the old values.

```
{
  "AssetID": 42,
  "Version": {
    "Data": {
      "SectionNetwork": [
        {
          "FQDN": [
            null,
            null,
            "one-and-only.test-computer.com"
          ]
        }
      ]
    }
  }
}
```

18.4.9 Displaying Asset attribute in Tickets (KIX Pro)

You can create time or event-based jobs that reference asset attributes. The macro action "Retrieve asset attributes" is used for this purpose. For example, working time specifications or service prices can be copied to a ticket if a service is selected as "Affected asset" in the ticket. This information can be used for reporting or workflow control. Using dynamic fields of the type "AssetReference", you can also display the asset attributes in the ticket details.

Application examples:

18.4.9.1 Show transfer price

Starting point:

The transfer price for the requested service is to be listed on a service request ticket. The billing price is to be set by selecting a service in the ticket.
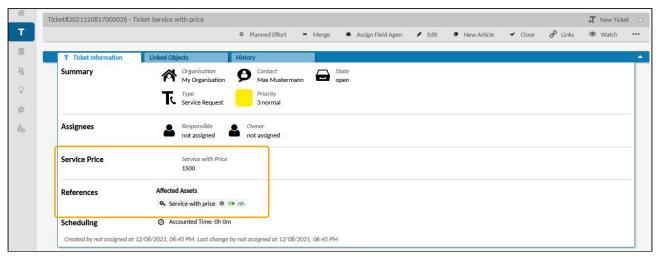
Fig.: Asset attributes in the ticket zoom view

Requirements:

The asset class "Service". This is delivered with the module "Service Catalogue" (KIX Pro). In KIX Start, a separate asset class "Service" must be created manually.
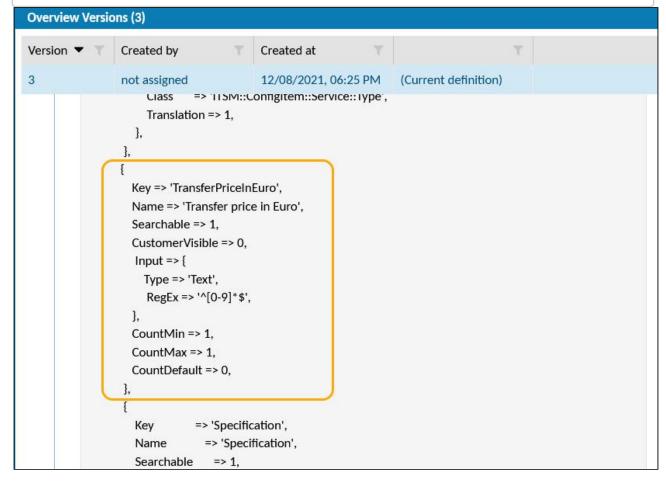
Procedure:

1. Navigate to *Assets > Asset classes*. Extend the asset class "Service" by the attribute (input field) "Transfer price in euro". Insert the following code block into the class definition:
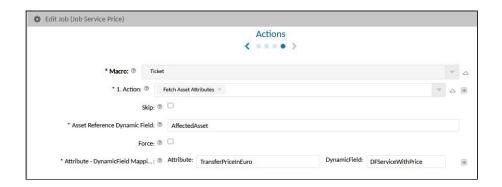
```
{
    Key => 'TransferPriceInEuro',
    Name => 'Transfer price in Euro',
    Searchable => 1,
    CustomerVisible => 0,
    Input => {
        Type => 'Text',
        RegEx => '^[0-9]*$',
    },
    CountMin => 1,
    CountMax => 1,
    CountDefault => 0,
},
```

2. Click on the green plus button and create a new asset of the class "Service" (e.g. "Service with price") and enter the transfer price in EUR, e.g. 1500.

3. Navigate to *System > Dynamic Fields*. Create a dynamic field of the field type "Text" and object type "Ticket", e.g. "DFServiceWithPrice".

4. Navigate to *Automation > Jobs*. Create a job of the type "Ticket" with the action "Fetch Asset Attributes". Use the following configuration:

- Job Information:
  - Job Type: Ticket
  - Name: Name of the job (z. B. "Job Service Price")
  - Validity: valid

- Execution Plan:
  - Events: TicketCreate

- Filter:
  - Filter: Type - contained in - Service Request

- Actions:
  - 1. Action: Fetch Asset Attributes
  - Asset Reference Dynamic Field: AffectedAsset
  - Attribute – DynamicField Mapping:
    - Attribut: TransferPriceInEuro (=Key-Attribute of the asset class - see step 1)
    - DynamicField: "DFServiceWithPrice" (=Name of the dynamic field - see step 3)

5. Include the dynamic field "DFServiceWithPrice" in the ticket detail view:

- Navigate to *System > SysConfig*.
- Open the configuration key *"ticket-details-info-card"*.
- Copy the following code block into the zoom view of a ticket. Use a JSON editor if necessary. The placement of the code block in the configuration determines where the Dynamic Field is displayed in the ticket detail view.

```
{
  "title": "Translatable#Service Price",
  "separator": true,
  "values": [
    [
      {
        "componentId": "object-avatar-label",
        "componentData": {
          "property": "DynamicFields.DFServiceWithPrice"
        },
        "conditions": [
          {
            "property": "DynamicFields.DFServiceWithPrice",
            "operator": "NE",
            "value": null
          }
        ]
      }
    ]
  ]
},
```

6. Save the change to the configuration key and reload the frontend.

7. Create a new ticket and select the asset "Service with price" under "Affected Asset". Under "Type", select the option "Service Request". The type must correspond to the type set in the filter of the job.

8. After saving the ticket, the asset "Service with price" will be displayed in the ticket detail view as the affected asset and the field "Service with price" will show the transfer price with a value of 1500 (EUR).

18.4.9.2   Specify working time in the ticket

Starting point:

A working time specification for the elimination of a incident is to be kept on an incident ticket. The working time is to serve as a default value and orientation on the ticket. The working time specification should be set by selecting a service in the ticket.

If the actual processing time required (account time) is also recorded on the ticket, both values can be compared and evaluated.

.

Abb.: Asset attributes in the ticket zoom view.

Requirements:

- The asset class "Service". This is supplied with the "Service Catalog" module (KIX Pro).
  A separate asset class "Service" must be created manually in KIX Start.

Procedure:

Navigate to *Assets > Asset Classes*. Extend the asset class "Service" with the attribute (input field) "WorkingTimeTargetInMinutes". Insert the following code block into the class definition:

```
{
    Key => 'WorkingTimeTargetInMinutes',
    Name => 'Working time target in minutes',
    Searchable => 1,
    CustomerVisible => 0,
    Input => {
        Type => 'Text',
        RegEx => '^[0-9]*$',
    },
    CountMin => 1,
    CountMax => 1,
    CountDefault => 0,
},
```

2. Click on the green plus button and create an asset of the class "Service" (e.g. "Service working time") and store the predefined working time in minutes, e.g. 120.

3. Navigate to *System > Dynamic Fields*. Create a dynamic field of the field type "Text" and object type "Ticket", e.g. "DFWorkingTimeTarget".

4. Navigate to *Automation > Jobs*. Create a job of the type "Ticket" with the action "Fetch Asset Attributes". Use the following configuration:

- Job Information:
    - Job Type: Ticket
    - Name: Name of the job (z. B. "Job Working time target")
    - Validity: valid
- Execution Plan:
    - Events: TicketCreate
- Filter:
    - Filter: Type - contained in - Incident
- Actions:
    - 1. Action: Fetch Asset Attributes
    - Asset Reference Dynamic Field: AffectetAsset
    - Attribute – DynamicField Mapping
        - Attribute: WorkingTimeTargetInMinutes (=key attribute of the asset class - see step 1)

- DynamicField: "DFWorkingTimeTarget" (=name of the dynamic field - see step 3)

5. Include the dynamic field "DFWorkingTimeTarget" in the ticket zoom view:

- Navigate to *System > SysConfig.*
- Open the configuration key "*ticket-details-info-card*".
- Copy the following code block into the ticket details configuration. Use a JSON editor if necessary. The placement of the code block in the configuration determines where the dynamic field is displayed in the ticket details view.

```
{
    "id": "ticket-details-info-card",
    "name": "Ticket Info Widget",
    "type": "Widget",
    "widgetId": "object-information-card-widget",
    "title": "Translatable#Ticket Information",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
        "avatar": [],
        "rows": [
            {⬚},
            {⬚},
            {⬚},
            {⬚},
            {
                "title": "Translatable#Working Time Specification",
                "separator": true,
                "values": [
                    [
                        {
                            "componentId": "dynamic-field-value",
                            "componentData": {
                                "name": "DFArbeitszeitvorgabe"
                            },
                            "conditions": [
                                {
                                    "property": "DynamicFields.DFArbeitszeitvorgabe",
                                    "operator": "NE",
                                    "value": null
                                }
                            ]
                        }
                    ]
                ]
            },
            {
                "title": "Translatable#Assignees",
```

```
{
  "title": "Translatable#Working Time Specification",
  "separator": true,
  "values": [
    [
      {
          "componentId": "dynamic-field-value",
          "componentData": {
              "name": "DFWorkingTimeTarget"
          },
          "conditions": [
            {
              "property": "DynamicFields.DFWorkingTimeTarget",
              "operator": "NE",
              "value": null
            }
          ]
      }
    ]
  ]
},
```

6. Save the changes to the SysConfig key and reload the frontend.

7. Create a new ticket and select the asset "Service working time" under "Affected asset". Under "Type", select the option "Incident". The type must correspond to the type set in the filter of the job.

8. After saving the ticket, the ticket detail view will show the service "Service working time" as the affected asset and the value 120 in the field "Working time in minutes".

18.5 KIX Objects

Below you will find a list of potential OBJECTS that can be used in placeholders. OBJECTS are always written in uppercase letters in placeholders and set as the 2nd argument in the placeholder `<KIX_OBJECT_attributes>.`

Example of OBJECT in a placeholder:	"Dear Mr. <KIX_**CONTACT**_UserLastname>, thank you […]"

Overview of KIX objects:

OBJEKT	Refers to:
AGENT	The last article of a ticket of the type Agent
ARTICLE	The referenced article of a ticket, e.g. for forwarding
CONFIGITEM	An asset
CONTACT	The contact person for a ticket
CURRENT	The person who is currently logged in
CUSTOMER	The last article of a ticket of the type Customer
FIRST	The first article in the ticket
LAST	The last article in the ticket
ORG	The customer of a ticket
OWNER	The owner of the ticket
QUEUE	The team folder containing the ticket
RESPONSIBLE	The person responsible for the ticket
TICKET	The ticket itself

OBJEKT	Refers to:
TICKETOWNER	The ticket owner
TICKETRESPONSIBLE	The person responsible for the ticket

18.6 KIX Attributes

Attributes are - to put it simply - the input and selection fields in a ticket or in an article. They contain the attribute values, i.e. the concrete data of a ticket.

Attributes are used:

- When **configuring the user interface**.
  For example, if you want to insert another column in a dashboard table. Specifying the attribute defines which data (attribute values) are displayed in the table column.

- in **KIX placeholders:**
  An attribute specifies the OBJECT to use in a placeholder. Attributes in placeholders are written in lower case and used as the 3rd argument in the placeholder:
  <KIX_OBJECT_**Attribute**>.

Example of an attribute in a placeholder:	"Dear Mr. <KIX_CONTACT_**Lastname**>, thank you very much […] "

Below you will find the overviews of the KIX attributes. Please understand that we cannot list all possible attributes due to the scope. However, our support will be happy to help you.

18.6.1 Ticket Attributes

- Always refer to the assigned properties of the referenced ticket or (if applicable) of the currently selected team.
- Can be used on the objects: CURRENT, OWNER, RESPONSIBLE, NOTIFICATION_RECIPIEN

Attributes (KIX Start)	Description
TicketNumber	Ticket nummer
Title	Ticket titel
TicketID	Ticket ID (number)
State	Ticket state (name)
StateID	Ticket state ID (number)

Attributes (KIX Start)	Description
StateType	State type (new | open | closed | pending […], etc.)
Priority	Piority (name)
PriorityID	Priority ID (number)
Lock	Locked (Yes/No)
LockID	Lock state ID (number)
Queue	Team (name)
QueueID	Team ID (number)
Organisation	Organization (Customers' name)
OrganisationID	Customer ID (number)
ContactID	Contact ID (number)
Owner	Ticket owner (name)
OwnerID	Ticket owner ID (number)
Type	Ticket type (designation)
TypeID	Ticket type ID (number)
Responsible	Person responsible for ticket (name)
ResponsibleID	ID of person responsible for ticket (number)
Age	Ticket Age
ArchiveFlag	Archiving indicator

Attributes (KIX Start)	Description
Created	Created on/at (date, time)
CreateBy	Created by (user ID)
Changed	Changed on/at (date, time)
ChangedBy	Changed by (user ID)
DynamicField_XYZ	Dynamic field with field name "xyz"

Additional ticket attributes for KIX Pro:

Attribute (KIX Pro)	Beschreibung
AccountedTime	Legible representation of the time required.
SLA	Name of the SLAs
SLAID	ID of the SLA
SLACriteria_FirstResponse_BusinessTimeDeviation	Deviation of the fulfillment time from the target time
SLACriteria_FirstResponse_Calendar	Calendar on which the SLA of the response time is based
SLACriteria_FirstResponse_FirstReminderTime	Time stamp on which the reminder notification for the criterion is sent
SLACriteria_FirstResponse_FulfillmentTime	Timestamp at which the criterion is fulfilled
SLACriteria_FirstResponse_Name	Name of the SLA criterion
SLACriteria_FirstResponse_SLAID	ID of the SLA from which the response time originates

Attribute (KIX Pro)	Beschreibung
SLACriteria_FirstResponse_StartTime	Timestamp of the start of the reaction time runtime
SLACriteria_FirstResponse_Status	Status of the SLA criterion
SLACriteria_FirstResponse_TargetTime	Time stamp by when the reaction must be made
SLACriteria_FirstResponse_TargetTimeDiff	Response time as specified when creating the SLA
SLACriteria_FirstResponse_TimeDeviation	Deviation of the fulfilment time from the target time in minutes
SLACriteria_FirstResponse_Violation	Violation of the response time
SLACriteria_Solution_BusinessTimeDeviation	Deviation of the fulfilment time from the target time
SLACriteria_Solution_Calendar	Calendar underlying the SLA of the resolution time
SLACriteria_Solution_FirstReminderTime	Timestamp at which the reminder notification for the criterion is sent out
SLACriteria_Solution_FulfillmentTime	Timestamp at which the criterion is fulfilled
SLACriteria_Solution_Name	Name of the SLA criterion
SLACriteria_FirstResponse_SLAID	ID of the SLA from which the response time originates
SLACriteria_Solution_StartTime	Timestamp of the start of the solution time runtime
SLACriteria_Solution_Status	Status of the SLA criterion

Attribute (KIX Pro)	Beschreibung
SLACriteria_Solution_TargetTime	Time stamp by when the solution must be made
SLACriteria_Solution_TargetTimeDiff	Resolution time as specified when creating the SLA
SLACriteria_Solution_TimeDeviation	Deviation of the fulfilment time from the target time in minutes
SLACriteria_Solution_Violation	Violation of the resolution time
DynamicField_ParentTickets	List of parent tickets
DynamicField_ChildTickets	List of child tickets
DynamicField_RelatedTickets	List of ticket references
DynamicField_XYZ	Readable representation of the DF value (text)
DynamicField_XYZ_Key	internal representation of the DF content (text, may contain IDs)
DynamicField_XYZ_Value	Readable representation of the DF value (text)
DynamicField_XYZ_Short	Readable representation of the DF value (text, includes a brief information)
DynamicField_XYZ_HTML	Readable formatted representation of the DF value (HTML)

18.6.2 Article Attributes

- Always relate to one or more specific articles
- Can be used on the following objects: TICKET, CUSTOMER, AGENT, ARTICLE

Attributes	Description
ID	ID of artikel

Attributes	Description
TicketID	ID of associated ticket
From	Sender of article
To	Recipient of article
CC	Recipient of duplicate articles
Subject	Article subject line
Body	Article text (content)
ContentType	Type of content (email | note)
CreatedBy	Created by (user ID)
CreateTime	Created (time)
ChangedBy	Changed on (date)
ChangeTime	Changed (time)
DynamicField_XYZ	Value of the dynamic field XYZ corresponds to the name of the dynamic field

18.6.3 Agent Attributes (User Data)

- Always relate to the assigned properties of the referenced ticket or (if applicable) of the team that is currently selected.
- Can be used on the following objects: CURRENT, OWNER, RESPONSIBLE, NOTIFICATION_RECIPIENT

Attributes	Description
UserID	Agent´s ID

Attributes	Description
UserLogin	Agent's username in the system (login)
UserLanguage	Agent´s Language in the system
Password	Agent's user password
Firstname	Agent's first name
Lastname	Agent's last name
Email	Agent's email address
Mobile	Agent's cell phone number
Phone	Agent's telephone number
Valid	Validity values defined in the system
UserComment	Comment
CreateTime	Created on/at (date, time)
ChangeTime	Geändert am (Datum, Zeit)

18.6.4 Contact Attributes

- Relate to the information about customer contacts, and always relate to the property Contact/ ContactID that is assigned to the referenced ticket.
- The internal identifiers of the contact attributes are used as variables.
- Can be used on the following object: CONTACT

Attributes	Description
Login	Username (login)

Attributes	Description
Password	Password
PrimaryOrganisationID	Organization
OrganisastionIDs	Other organizations
Firstname	First name
Lastname	Last name
Title	Title
Email	Email address (communication)
Email1 | Email2| Email3| Email4| Email5	Additional Email addresses of the contact
Phone	Telephone number (communication)
Mobile	Cell phone number (communication)
Fax	Fax (Communication)
Street	Street (address)
City	City (address)
Zip	ZIP code (Address)
Country	Country (Address)
Comment	Comment
Valid	Validity values defined in the system
CreateTime	Created on/at (date, time)

Attributes	Description
ChangeTime	Changed on/at (date, time)

18.6.5 Customer Attributes

- Relate to the information about organizations (customers), and always relate to the property Customer/OrgID that is assigned to the referenced ticket.
- The internal identifiers of the organization attributes are used as variables
- Can be used on the following object: ORG

Attributes	Description
ID	Customer ID
Number	Customer number
Name	Customer name
Url	Customer URL
Street	Customer street
City	Customer city
Zip	Customer ZIP code
Country	Customer country
Comment	Saved comment
Valid	Validity values defined in the system
CreateTime	Created at
ChangeTime	Changed at

18.7 Overview of the search operators

Search operators are used both in the complex search and in filters (e.g. for jobs or actions). The following is an overview of the search operators in KIX and their use:

Legend:

Indicator	Description
*	Attribute is pre-selected
bold	Selected by default
**	Available only in KIX Pro

For each attribute, all available (valid & invalid) values are always displayed to also allow access to tickets with invalid values.

> ⓘ **The following generally applies to dynamic fields:**
>
> Only valid dynamic fields can be selected.

18.7.1 General

Operator	Data type	Descripton
equals	Strings, Arrays, Numerical values	The value to be found is **exactly** the same as the value entered, e.g.: • Name of the team: "Purchasing • is found via the input: "Purchasing
starts with	Strings	The value to be found starts with the exact value entered, e.g.: • Ticket title: "Problem with printer in meeting room". • is found by entering "Problem
ends with	Strings	The value to be found ends with the exact value entered, e.g.: • Ticket title: "Problem with printer in meeting room". • is found via the entry "Meeting room
contains	Strings	The entered value is included in the value to be found, i.e. wildcards (*) are automatically added to the right and left of the search word, e.g.: • Ticket title: "Problem with printer in meeting room". • is found by entering "printer

Operator	Data type	Descripton
Wildcard search *(like)*	Strings	Similar to the search with "contains", but here the user can replace unknown parts at any position of the search phrase with wildcards (*). If no (*) are used, this search operator corresponds to "Is equal to", e.g.: • Ticket title: "Problem with printer in meeting room". • is found by entering "Problem * in the meeting room".
contains in *(in)*	Arrays	The value to be found is contained in a concrete set of the selected values, e.g.: • The user knows that the ticket being searched for is in one of the teams "Purchasing", "Sales" or "Marketing", but not in which one. • is found by selecting the teams "Purchasing", "Sales" and "Marketing
not contains in	Arrays	The value to be found is not contained in a specific set of the selected values, e.g: • The user knows that the ticket they are looking for is not in any of the "Purchasing", "Sales" or "Marketing" teams • is found by selecting the "Purchasing", "Sales" and "Marketing" teams

18.7.1.1   Search for times

Operator	Data type	Description
bevor/less than *(less_than)*	Date, DateTime*, Numerical values	Numerical values: < / less than All tickets before the entered date/time are found. The date itself is not included. It is possible to search for a date only or date & time, e.g.: • The user wants to know how many tickets have been received for a service before today. • will be found by selecting the corresponding service as search attribute and "Created" "before" the desired date as another search attribute.
to/equal or less *(less_than_or_equal)*	Date, DateTime*, Numerical values	Numerical values: <= / less than or equal All tickets up to and including the date/time entered will be found. It is possible to search only a date or date & time, e.g.: • The user wants to know how many tickets have been received in relation to a service by the end of a previous month. • will be found by selecting the corresponding service as search attribute and "Created" "Until" the desired date as another search attribute.

Operator	Data type	Description
after/greater than *(greater_than)*	Date, DateTime*, Numerical values	Numerical values: > / greater than All tickets from the date/time entered will be found. The date itself is not included. It is possible to search only for a date or date & time, e.g.: • The user wants to know how many tickets have accrued after 24.12. in the Purchasing team. • is found by selecting the team "Purchasing" as the search attribute and "Created" "after" 24.12. as a further search attribute.
since/equal or greater *(greater_than_or_equal)*	Date, DateTime*, Numerical values	Numerical values: >= / greater than or equal All tickets from including the entered date/time will be found. It is possible to search for a date only or date & time, e.g.: • The user wants to know how many tickets have been accumulated in the Purchasing team since the beginning of the month. • will be found by selecting the team "Purchasing" as the search attribute and "Created" "Since" the first of the month as another search attribute.

Operator	Data type	Description
between *(less_than_or_equal & greater_than_or_equal)*	2x Date + DateTime*, Numerical values	All tickets between an entered start date/ time and end date/time will be found. It is possible to search only a date or date & time, e.g.: • The user wants to display all tickets created by a specific customer in the month of July. • will be found by selecting the corresponding customer as search attribute and "Created" & "between" 1.7. and 31.7. as further search attribute.
within the last	Numerical values	Natural number + Time unit • The user would like to have all tickets displayed that were created by a certain customer within the last 24 hours. • is found by selecting the corresponding customer as search attribute and "Created" & "within the last" & "24" & "hours" as further search attribute.
within the next	Numerical values	Natural number + Time unit • The user wants to display all tickets of a certain customer that have to be completed within the next week. • is found by selecting the corresponding customer as search attribute and "plan end" & "within the next" & "1" & "week(s)" as further search attribute.

Operator	Data type	Description
within	2x Numerical values	Type + Natural number + Time unit Searches/filters within two relative time values. If a numerical value is missing, the filter is not applied. • The user wants to display all assets whose warranty expires in the last 1 day to 2 months. • Is found by selecting the class Software as search attribute and "expiry date "[1] & "within" & "the last" & "1" & "days" to "the next" & "2" & "months" as further search attribute. The 1st value (left) is stored with "gte - greater than equals" and should always contain the smaller value or use the type "the last". The 2nd value (right) is saved with "lte - lower than equals". It should always contain the larger value or use the type "the next". If this is not the case, the values are swapped for a sensible logic. If both [types] are the same: • for type "the last", the 1st numerical value plus unit (left) must be greater. • for type "the next" the 2nd numerical value plus unit (right) must be greater. Otherwise the values are swapped. [1] If necessary, the date field must be implemented manually in the class definition (see also: Editing a Class Definition (see page 130)).

Operator	Data type	Description
more than ago	Numerical values	Natural number + Time unit • The user wants to display all tickets from a specific customer that were created more than 10 days ago. • is found by selecting the corresponding customer as search attribute and "Created" & "More than" & "10" & "Day(s)" as further search attribute.
in more than	Numerical values	Natural number + Time unit • The user wants to display all tickets of a certain customer that have to be completed in more than four weeks. • is found by selecting the corresponding customer as search attribute and "plan end" & "in more than" & "4" & "week(s)" as further search attribute.
in less ago	Numerical values	Natural number + Time unit • The user wants to display all tickets from a specific customer that were created less than 10 days ago. • is found by selecting the corresponding customer as search attribute and "Created" & "Less than" & "10" & "Day(s)" as further search attribute.
in less than	Numerical values	Natural number + Time unit • The user wants to display all tickets from a certain customer that have to be completed in less than 8 hours. • is found by selecting the corresponding customer as search attribute and "plan end" & "in less than" & "8" & "hour(s)" as further search attribute.

> ⓘ **Note on searching for times**
>
> If no time is specified in a DateTime field, default values (before/until: 00: 00:00; after/as of:
> 23:59:59) are automatically entered in this field.

18.7.2 Use of logical operators

18.7.2.1 When searching for contacts

Supported:

- Full text search contacts in the complex search
- all form fields "Assign contact" within the application; these always perform a contact full-text search

The following special characters are supported:

Special characters	Description	Example
& resp. +	Combine terms (AND operation)	Search for "John&Doe" or for "John+Doe" searches for "John" and "Doe" and finds the contact John Doe
|	Separates terms to find anything that contains "either one or the other" (OR operation).	Search for "Hark|Potts" searches for "Hark" OR "Potts" and finds the contacts "Bill Potts" and "Jack Harkness".
*	see Wildcard search (see page 886)	Search for "J*Yves" shows the contact "Jacque-Yves Cousteau".

18.7.2.2 Full text search

For the attribute "full text search" of the complex search, wildcards (see page 886) are also supported for the operator "contains". The search fields in the upper dashboard area (customer dashboard, asset dashboard, admin/translations, admin/SysConfig) also support wildcards.

18.7.3 Ticket search

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Close code	**contained in**	Compare dynamic field type "Selection"
Sender type	**contained in**	Dropdown: Agent, System, extern
~~Age~~	~~equals~~ ~~before/less than~~ ~~after/greater than~~ ~~until/equal or less~~ ~~since/equal or greater~~ ~~between~~	~~Free text~~ ~~Usage note:~~ ~~The age search~~ ~~expects the number in~~ ~~seconds.~~ ℹ️**The search attribute "Age" has been dropped.** Instead, use a relative search based on the creation date/time. In future, the ticket age will be determined from the difference between ticket creation and NOW. Sorting by ticket age is based on the time of ticket creation.
To	**contains** starts with ends with equals Wildcard Search	Free text
Owner	**contained in**	Multiselect, dropdown
Watch User	**contained in**	

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Subject	**contains** starts with ends with equals Wildcard Search	Free text
Affected Asset	**contained in**	Compare dynamic field type "AssetReference"
Cc	**contains** starts with ends with equals Wildcard Search	Free text
Parent tickets**	**contained in**	Compare dynamic field type "TicketReference"
Created on	**before/less than** after/greater than until/equal or less since/equal or greater between	DateTime
Created with User	**contained in**	
Created with Status	**contained in**	
Created with Team	**contained in**	
Created with Type	**contained in**	
Created with Priority	**contained in**	

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Created by	**contained in**	Multiselect, Dropdown of all agents
Articel Create Time	**before/less than** after/greater than until/equal or less since/equal or greater between	DateTime
Changed at	**before/less than** after/greater than until/equal or less since/equal or greater between	DateTime
Changed by	**contained in**	Multiselect, Dropdown of all agents
Risk Assumption Remark	**contains** starts with ends with equals Wildcard Search	Compare dynamic field type "TextArea"
Closed at	**before/less than** after/greater than until/equal or less since/equal or greater between	DateTime
Channel	**contained in**	Dropdown, Multiselect
Child Tickets**	**contained in**	Compare dynamic field type "TicketReference"

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Organisation (the ID and the name are searched for)	**contained in**	Autocomplete Multiselect
Contact (the ID and the name are searched for)	**contained in**	Autocomplete Multiselect
Mobile Processing	**contained in**	Compare dynamic field type "Selection"
Body	**contains** starts with ends with equals Wildcard Search	Free text
Attachment Name	**contains** starts with ends with equals Wildcard Search	Free text
Plan Begin	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
Plan End	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Planned Effort (min) **	**equals** before/less than after/greater than until/equal or less since/equal or greater between	Free text
Priority (the ID or the name are searched for)	**contained in**	Multiselect, Dropdown
SLA Deviation**	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
SLA Deviation (Service Time)**	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
SLA Fulfillment Time**	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
SLA Criterion**	**contained in**	Dropdown, Multiselect
SLA Criterion - SLA/Service Agreement**	**contained in**	Dropdown, Multiselect

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
SLA Status**	**contained in**	Dropdown, Multiselect
SLA Violation**	**equals**	Dropdown, Multiselect
SLA Target Time**	**equals**	DateTime
SLA/Service Agreement**	**contained in**	Dropdown, Multiselect
Service via DF "Affected Asset"	*Compare DF-Type "Asset Reference"*	
Lock State (the ID is searched for)	**equals**	Singleselect
Status (the ID is searched for)	**contained in**	Multiselect, dropdown
Status type (the ID or the name are searched for)	**contained in**	Multiselect, dropdown
Viewable Status	**contained in**	Multiselect, dropdown *The statuses in "Open" correspond to those stored in "ViewableStateTypes" in SysConfig and include tickets that are still being processed in some way or are waiting for (automatic) completion..*
Team* (the ID or the name are searched for)	**contained in**	Multiselect, dropdown

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Anonymise Ticket**	**contained in**	Compare dynamic field type "Selection"
Ticket#	**contains** starts with ends with equals Wildcard Search	Free text
Titel*	**contains** starts with ends with equals Wildcard Search	Free text
Type (the ID is searched for)	**contained in**	Multiselect, Dropdown
Responsible (the ID is searched for)	**contained in**	Autocomplete, Multiselect
Related Tickets**	**contained in**	Compare dynamic field type "TicketReference"
Full Text*	**contains**	Free text on: Number, Title, Article content, From, To, Cc
From	**contains** starts with ends with equals Wildcard Search	Free text

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Pending until	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
Accounted Time	**equals** before/less than after/greater than until/equal or less since/equal or greater between	Free text
Last changed at	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
Dynamic Field Type "Text"	**contains** starts with ends with equals Wildcard Search	Free text
Dynamic Field Type "TextArea"	**contains** starts with ends with equals Wildcard Search	Free text

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Dynamic Field Type "Date"	**equals** before/less than after/greater than until/equal or less since/equal or greater between	Date
Dynamic Field Type "DateTime"	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
Dynamic Field Type "Selection"	**contained in**	Dropdown, Multiselect
Dynamic Field Type "AssetReference"	**contained in**	Dropdown, Autocomplete, searches via number and name
Dynamic Field Type "TicketReference"	**contained in**	Dropdown, Autocomplete, searches via number and name

18.7.4 Asset Suche

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Full text*	**contains**	Free text on: Asset name & Asset number

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Name*	**contains** starts with ends with equals Wildcard Search	Free text
A# *	**contains** starts with ends with equals Wildcard Search	Free text
Current incident state*	**contained in**	Autocomplete, Multiselect
Current deployment state*	**contained in**	Autocomplete, Multiselect
Changed on	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
Changed by	**contained in**	Multiselect, Dropdown of all agents
Class*	**contained in**	Multiselect (all valid asset classes)
GeneralCatalog Types		
CIClassReference	**contained in**	Autocomplete, Multiselect
Organisation	**contained in**	Autocomplete, Multiselect

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Contact	**contained in**	Autocomplete, Multiselect
Date	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
DateTime	**equals** before/less than after/greater than until/equal or less since/equal or greater between	DateTime
GeneralCatalog	**contained in**	Autocomplete, Multiselect
TextArea	**contains** starts with ends with equals Wildcard Search	Free text
Text	**contains** starts with ends with equals Wildcard Search	Free text

18.7.5 Organisation search

Attribute	Operator (2nd colunm)	Field type (3rd column) / Comment
Full text*	**contains**	Free text on: KNR, Name, URL, Street, City, Country, ZIP
Name*	**contains** starts with ends with equals Wildcard Search	Free text
CNO*	**contains** starts with ends with equals Wildcard Search	Free text
URL	**contains** starts with ends with equals Wildcard Search	Free text
Street	**contains** starts with ends with equals Wildcard Search	Free text
ZIP	**contains** starts with ends with equals Wildcard Search	Free text

Attribute	Operator (2nd colunm)	Field type (3rd column) / Comment
City	**contains** starts with ends with equals Wildcard Search	Free text
Country	**contains** starts with ends with equals Wildcard Search	Free text
Validity	**contained in**	Multiselect, Dropdown

18.7.6 Contact search

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Full text*	contains	Free text on: Login Name, Email, First Name, Last Name, Title, Phone, Fax, Mobile, Street, City, County, ZIP
First Name*	contains starts with ends with equals Wildcard Search	Free text
Last Name*	contains starts with ends with equals Wildcard Search	Free text
Email*	contains starts with ends with equals Wildcard Search	Free text
Login Name*	contains starts with ends with equals Wildcard Search	Free text
Organisation	**contained in**	Autocomplete, Multiselect

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Phone	**contains** starts with ends with equals Wildcard Search	Free text
Mobile	**contains** starts with ends with equals Wildcard Search	Free text
Fax	**contains** starts with ends with equals Wildcard Search	Free text
Street	**contains** starts with ends with equals Wildcard Search	Free text
ZIP	**contains** starts with ends with equals Wildcard Search	Free text
City	**contains** starts with ends with equals Wildcard Search	Free text

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Country	**contains** starts with ends with equals Wildcard Search	Free text
Validity	**contained in**	Multiselect, Dropdown

18.7.7 FAQ Search

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Full Text*	contains	Free text on: Approved, Changed on, Changed by, Created on, Created by, Symptom, Cause, Solution, Comment, Tags, Languae, Title, FAQ#, Validity, Enable for Self Service Portal
FAQ#	contains starts with ends with equals Wildcard Search	Free text
Title*	**contains** starts with ends with equals Wildcard Search	Free text
Category*	**contained in**	Multiselect, Dropdown

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Enable for Self Service Portal	**contained in**	Multiselect
Language	**contained in**	Multiselect, Dropdown
Symptom	**contains** starts with ends with equals Wildcard Search	Free text
Cause	**contains** starts with ends with equals Wildcard Search	Free text
Solution	**contains** starts with ends with equals Wildcard Search	Free text
Comment	**contains** starts with ends with equals Wildcard Search	Free text
Tags	**contained in**	Multiselect, Dropdown
Created at	**equals** before/less than after/greater than until/equal or less since/equal or greater between	Date, Date/Time

Attribute	Operator (2nd column)	Field type (3rd column) / Comment
Created by	**contained in**	Multiselect, Dropdown of all agents
Changed at	**equals** before/less than after/greater than until/equal or less since/equal or greater between	Date, Date/Time
Changed by	**contained in**	Multiselect, Dropdown of all agents
Validity*	**contained in**	Multiselect, Dropdown
Related Assets	**contained in**	Multiselect, Dropdown

19   Release Information

Below you will find a brief overview of the contents of the latest releases of KIX 18. You can find more information in the forum: https://forum.kixdesk.com/index.php. In the forum you will also find information on past releases.

Content on this page:

19.1   Release KIX18 v31

(Build: 4438-0.1868-0 | SSP: 151-0 | released in February 24th 2024)

Functionality	Affected Product	Brief Description	Described in the Manual under

Ticket: Set conditional automatic visibility in the Self Service Portal (SSP)	KIX Start KIX Pro KIX Cloud SSP	SSP visibility is no longer determined automatically in the frontend. Instead, it is set by the central configuration in SysConfig. This is active by default, but can be deactivated. When an article is created in the "Email" channel, a check is made to see whether one of the mail recipients (To, Cc, Bcc) matches the ticket contact. If this is the case, the visibility for the Self Service Portal is activated. This is regardless of what the agent has set via the GUI.	• Admin Manual
			   • KIX Pro > Self Service Portal > Control visibilities in the SSP
			• User Manual
			   • KIX Start > Ticket Dashboard > How to Create and Edit a Ticket

Functionality	Affected Product	Brief Description	Described in the Manual under
Complex search: Asset complex search also possible in older versions	KIX Start KIX Pro KIX Cloud	The Asset complex search can now also be used to search in older versions.	• User Manual • KIX Start > Toolbar > Complex search
Macro Actions: "While" and "Wait"	KIX Start KIX Pro KIX Cloud	The "While" macro action generates a loop in which a configured macro is executed as long as a condition applies. The "Wait" macro action forces the job to be paused for a defined period of time. After this period, the execution of the next macro action is continued. This makes it possible to react more specifically to responses to requests and, if necessary, to trigger a delayed repetition of the request.	• Admin Manual • KIX Start > Appendix > Macro Actions > Overview of Macro Actions for KIX Pro
Asset: Asset export - Name instead of IDs for asset references	KIX Start KIX Pro KIX Cloud	For the asset export, the names of the referenced assets (CI class reference) can be exported instead of the asset numbers/IDs in order to reuse the export data outside of KIX. This can be stored in the asset class definition.	• Admin Manual • KIX Start > Assets > Asset Classes > Attribut types in the class definition

Functionality	Affected Product	Brief Description	Described in the Manual under

KIX Connect web service: Macro Action "SetVariable" extended by Append	KIX Pro KIX Cloud Connect	Variables can hold multiple values and therefore form arrays. For KIX Connect Webservice, there are additional variable filters that can be used to influence the formation of the arrays.	• Admin Manual
			  • KIX Start > Appendix > Variables
			  • KIX Pro > Add-ons > Connect > Connect Webservice

Communication: PostMaster - Controlling the visibility of incoming e-mails in the SSP	KIX Start KIX Pro KIX Cloud SSP	Incoming emails are only visible in the SSP under certain conditions. The initial behaviour can be modified in various SysConfig keys.	• Admin Manual • KIX Pro > Self-service Portal > Control visibilities in the SSP
Automation: Duplicate jobs	KIX Pro KIX Cloud	Jobs can now be duplicated. The duplicate is created on the basis of an existing job.	• Admin Manual • KIX Pro > Jobs - Enhanced Functions

Functionality	Affected Product	Brief Description	Described in the Manual under
Communication: Resolution of contacts based on URL parameters	KIX Start KIX Pro KIX Cloud SSP	A contact can be determined using the parameters transferred in an URL. If KIX is linked to a telephone system or a telephone client, the caller information can be added as a parameter to the KIX URL. KIX can use this information to open the ticket creation screen and store the caller as a contact in the ticket. The prerequisite is that the telephone system/the telephone client supports calling URLs.	• Admin Handbuch • KIX Start > Interfaces > Use URL of the browser
Workflow Rule Sets: Check for empty/ any/contains	KIX Pro KIX Cloud SSP	Workflow rule sets can be used to check whether a (dynamic) field contains one, no or certain values.	• Admin Handbuch • KIX Pro > Workflow > Rule Sets
Placeholder: Placeholder suffix "_ObjectValue"	KIX Start KIX Pro KIX Cloud SSP	The placeholder suffix "_ObjectValue" now provides a comma-separated array of IDs and no longer just the first entry. The placeholder suffix "_Key" can still be used, but we recommend using "_ObjectValue".	• Admin Manual • KIX Start > Appendix > Placeholders > Overview of KIX Placeholders

Functionality	Affected Product	Brief Description	Described in the Manual under

Sorting: Tables or lists take the selected sorting into account	KIX Start KIX Pro KIX Cloud SSP	The display of tables and restricted lists takes a requested sorting into account.	• Admin Manual
			  • KIX Start > System settings > Configuration of User Interface > Basics of GUI Configuration > The configuration of widgets > The table widget
			• User Manual
			  • KIX Start > Notes on Working with KIX 18 > General Function Description

Contact: Automatic allocation of the organisation	KIX Start KIX Pro KIX Cloud SSP	To save manual maintenance effort for agents, automatically created contacts can be assigned to an existing organisation based on the email sending domain. The organisation can also be assigned automatically if no organisation is assigned to a contact when creating/ editing a contact or when creating a new ticket.	• Admin Manual
			  • KIX Start > System settings > Settings for organisations and contacts > Automatic assignment of the organisation to a contact
			  • KIX Pro > Self Service Portal > Control visibilities in the SSP
			• User Manual
			  • KIX Start >
			  • Organisation Dashboard > How to Create an Organisation and Edit this Entry

Functionality	Affected Product	Brief Description	Described in the Manual under

Link: Parent-child link of FAQ with tickets initially inactive	KIX Start KIX Pro KIX Cloud SSP	If FAQs are linked to tickets, the "Parent-child" link type will not be available in future. This is for the sake of clarity and better data maintenance. If required, the admin can set the key to active so that the link type can be used again.	• Admin Manual
			  • KIX Start > System settings > Settings for FAQ > Linking FAQs and Tickets
			• User Manual
			  • KIX Start > Notes on Working with KIX 18 > General Function Descriptions

Macro Actions: ArticleCreate - Suppress sending of e-mails	KIX Start KIX Pro KIX Cloud SSP	When using the "Create article" macro action, the sending of emails can be prevented, for example to avoid duplicate emails in linked systems.	• Admin Manual • KIX Start > Appendix > Macro Actions > Overview of Macro Actions for KIX Start

Templates/Actions: Help/hint texts for input fields	KIX Pro KIX Cloud SSP	Customised information texts on the use of input and selection fields can be maintained in templates and actions.
		
		Users of the agent or self-service portal can thus be provided with information on the details to be entered or how to use the field.
			• Admin Manual
			  • KIX Pro > Workflow > Actions > Create and configure actions
			  • KIX Pro > Workflow > Templates

Functionality	Affected Product	Brief Description	Described in the Manual under
System: KIX Field Agent App - Editing dynamic fields	KIX Start KIX Pro KIX Cloud FAA	Selected dynamic fields of a ticket can be edited directly in the app. The system admin defines which dynamic fields of a ticket can be displayed in the app and which can be edited.	• KIX Field Agent App • KIX Field Agent > Administration
Placeholder: Abbreviation/ extension of the quoted text	KIX Start KIX Pro KIX Cloud SSP	In future, the admin will be able to configure how many lines of content should be transferred when quoting.	• Admin Manual • KIX Start > Appendix > Placeholder > Overview of KIX Placeholders

Placeholder: Prevent localisation	KIX Start KIX Pro KIX Cloud	KIX placeholders are resolved in the default language. To prevent unwanted side effects, it is now possible to add an exclamation mark to the placeholder name to prevent localisation.	• Admin Manual
			  • KIX Start > Appendix > Placeholders
			  • KIX Start > Appendix > Macro Actions

LDAP/AD authentication: Extension of synchronisation to process "mail" array attributes + AD array attribute values are written to contact attribute	KIX Start KIX Pro KIX Cloud SSP	Each KIX attribute can be composed of n LDAP attributes and separator strings. This means that array attributes in LDAP can be written to a single target attribute as a concatenated character string. This allows, for example, all stored emails to the AD to be stored (max. 6 values per array).	• Admin Manual
			  • KIX Start > User management > Users > Authentication/ Authorisation and Connection to Active Directory
			  • KIX Pro > Jobs - Enhanced Functions > LDAP/AD Synchronisation

Functionality	Affected Product	Brief Description	Described in the Manual under
System: Console command for system data overview	KIX Start KIX Pro KIX Cloud SSP	KIX enables the display of important data and variables in the system via console command and thus the detection of possible sources of error. After consultation with our support team, this information can also be sent (automatically) to our support team.	• Admin Manual • KIX Start > System settings > Display of support information
KIX Connect: Extension with OAuth2 authentication	KIX Pro KIX Cloud Connect	A web service can be configured which authenticates itself against the provider via OAuth2.	• Admin Manual • KIX Pro >Add-Ons > Connect
KIX Connect Webservice: Use MFA (TOTP) Auth in WebhookExtended	KIX Pro KIX Cloud Connect	When calling up a WebhookExtended, MFA authentication - based on TOTP (time based one time password) and another fixed password - can be used.	• Admin Manual • KIX Pro >Add-Ons > Connect
KIX Connect Webservice: Provision of specific endpoints	KIX Pro KIX Cloud Connect	Web services can now also be set up as data sources in order to integrate remote systems into KIX.	• Admin Manual • KIX Pro >Add-Ons > Connect > Connect Webservice

Functionality	Affected Product	Brief Description	Described in the Manual under

Assets: Asset details - Viewing the "Affected Services" in the "Linked Objects" tab	KIX Start KIX Pro KIX Cloud	In the detailed view of an asset of the Service class, the linked tickets are now also visible in the "Linked Objects" tab.	• Admin Manual
			    • KIX Start > Assets - The Asset Device Database > Configuration settings
			• User Manual
			    • KIX Start > Asset Dashboard > How to Create and Edit an Asset (Config Item)

Macro Actions: Variable filter XML	KIX Start KIX Pro KIX Cloud	With the variable filter "XMLUtil.FromXML", XML can be used as an input data structure for XSL transformations or other macro actions with structured input parameters in order to process this directly.	• Admin Manual • KIX Start > Appendix > Variables

Functionality	Affected Product	Brief Description	Described in the Manual under
Performance analysis: FE request logs	KIX Start KIX Pro KIX Cloud	KIX logs the HTTP frontend requests and saves this information once a day in a log file. The log file can be viewed and downloaded in the log file management. In addition, KIX Pro provides you with a support tool that gives you an overview of the current HTTP requests on the system. The metrics of the request logs for the front end are analysed and presented graphically for you. You can select and analyse individual or multiple log files.	• Admin Manual • KIX Pro > System - Enhanced Functions > Analyses > FE-Request Logs

19.2   Release KIX18 v30

(Build: 4346-0.1800-0 | SSP2: 135-0 | released in November 2nd 2023)

> ⚠ **Disclaimer**
>
> As of release v31, the Self Service Portal 1 (SSP1) will be discontinued. Please switch to using the SSP2 until then.

> ⓘ **Update note**
>
> - This update makes the SSP2 available in the KIX.Cloud as the Self Service Portal.
>   - The URL https://yourkix-ssp.kix.cloud points to the Self Service Portal 2.
>   - The SSP1 will be accessible at https://yourkix-ssp1.kix.cloud until its complete removal with v31 .
>   - Further information for users of KIX as an on-premises solution can be found in our forum article: https://forum.kixdesk.com/index.php?topic=12079.msg17752#new.
> - The execution of system commands via admin configuration will now be prevented. If configurations already exist, they will no longer work.

Functionality	Affected Product	Brief Description	Described in the Manual under
Placeholder: Macro Action "Set Asset Name"	KIX Pro KIX Cloud KIX MP	The macro action "Set Asset Name" makes it possible to change the name of an asset per job without changing the version of the asset. It is mainly used by the add-on "KIX Maintenance Plan", but can also be used in own jobs.	• Admin Manual • KIX Start > Appendix > Overview of Macro Actions for KIX Pro

Functionality	Affected Product	Brief Description	Described in the Manual under
LDAP/AD authentication: Restriction via X-Forwarded-For header	KIX Start KIX Pro KIX Cloud SSP	For logging in to the agent or self-service portal, the valid-user authentication backend also restricts users to specific client IP addresses based on the X-Forwarded-For header.	• Admin Manual • KIX Start > User Management > Users > Authentication/ Authorisation and Connection to Active Directory

UI/UX: Use of icon libraries on objects	KIX Start KIX Pro KIX Cloud SSP	As an alternative to uploading a graphic file, icons from icon libraries (e.g. FontAwesome) can now be selected for icon-supporting objects (e.g. ticket types, statuses, teams, etc.).	• Admin Manual
			   • KIX Start > Ticket > Priorities
			   • KIX Start > Ticket > States
			   • KIX Start > Ticket > Teams
			   • KIX Start > Ticket > Types
			   • KIX Start > Assets > Asset classes > Create an asset class
			   • KIX Start > Knowledge Database
			   • KIX Start > Assets > General Catalog

Functionality	Affected Product	Brief Description	Described in the Manual under

Automation: Job at contacts	KIX Start KIX Pro KIX Cloud	Jobs can now be executed on contacts, e.g. to trigger a process when a contact reaches a certain date. Example: A date is stored on a contact that states when the contact will leave the company. As soon as the date is reached, the off-boarding process should be started automatically.	• Admin Manual
			  • KIX Start > Automation > Jobs > Create or edit a job
			  • KIX Start > System settings > Configuration of User Interface > Configuration examples of GUI configuration >
			    • Integrating a Dynamic field
			    • Displaying Values of Dynamic Fields

Password entry: Switch to plain text	KIX Start KIX Pro KIX Cloud SSP	It is possible to switch to plain text when entering passwords in order to avoid incorrect entries.	• Admin Manual
			  • KIX Start > User Management > Users > Creating and Editing a User and assigning roles
			• User Manual
			  • KIX Start > Login
			  • KIX Start > Toolbar > Special Features
			• SSP Manual
			  • Self Service Portal 2 > SSP2: Login

Functionality	Affected Product	Brief Description	Described in the Manual under
Permissions: Enable ticket creation without update permission	KIX Start KIX Pro KIX Cloud	Agents can create tickets with certain properties (especially in Team X) without having further update permissions on the ticket after it has been created. The same applies to assets.	• Admin Manual • KIX Start > User Management > Rolles > Ticket creation without update permission
Dynamic field types: DF Type "Contact" - extension of the placeholders	KIX Start KIX Pro KIX Cloud SSP	In jobs and macros, the user data of a value in a dynamic field of the type ContactReference can be accessed, e.g. in order to assign a ticket to another agent or responsible. The values of the specified objects can also be accessed in other referencing dynamic fields (type AssetReference, organisation reference, etc.).	• Admin Manual • KIX Start > Appendix > Placeholders > Overview of Placeholders
Search: Supplement sort	KIX Start KIX Pro KIX Cloud SSP	When filtering and restricting lists, the selected sorting is taken into account over the complete result list.	• User Manual • KIX Start > Notes on working with KIX 18 > General Function Descriptions

Functionality	Affected Product	Brief Description	Described in the Manual under
Communication: Notification in the event of an error when sending an email	KIX Start KIX Pro KIX Cloud SSP	If an item could not be sent by email, the agent receives a notification and the item is marked accordingly. The notice is also displayed in the toolbar in the "Notifications" button so that the agent can trigger a resend.	• User Manual • KIX Start > Ticket Dashboard > Ticket Zoom View
Authorisation: Restrict administration ("Mini-admin")	KIX Start KIX Pro KIX Cloud	Users can be assigned limited administrator roles. These can only view and configure some of the admin functions that can be administered (Admin text modules, Admin FAQ).	• Admin Manual • KIX Start > User management > Roles/ Permissions > Overview of preconfigured roles
Authorisation: Restrict access to admin area despite update rights ("Mini-admin")	KIX Start KIX Pro KIX Cloud	In the configuration of roles, there is a switch to show or hide the admin area for role holders. For example, so that agent users with update rights to certain objects cannot view the admin area.	• Admin Manual • KIX Start > User management > Roles/ Permissions > Configure and assign a role

Functionality	Affected Product	Brief Description	Described in the Manual under
Workflow Rules: Use of names instead of just "IDs"	KIX Pro KIX Cloud SSP	The respective object IDs can be different for each system. If one system is transferred to the other, this can lead to incorrect assignments in workflow rules. Therefore, the conditions and the parameter set support the unique names of objects in addition to the IDs. Another advantage: The IDs of the objects do not have to be determined in advance to create the rulesets.	• Admin Manual • KIX Pro > Workflow > Rulesets
Communication: Inbox with POP3S_OAuth2	KIX Pro KIX Cloud SSP	POP3 SSL/TLS (POP3S) with OAuth2 authentication is supported for connecting Office365 mailboxes.	• Admin Manual • KIX Start > Communication > EMail > Inbox

UI/UX: More compact display of the CKEditor and deactivation of SCAYT	KIX Start KIX Pro KIX Cloud SSP	KIX delivers the CKEditor with a compact function bar and with the SCAYT plug-in deactivated. Administrators can, if necessary and on their own responsibility, activate the SCAYT plugin and adapt the available buttons in the editor.	• Admin Manual
			  • KIX Start > System settings > Settings for tickets > Configure CKEditor
			• User Manual
			  • KIX Start > Notes on Working with KIX 18 > General Function Descriptions

Functionality	Affected Product	Brief Description	Described in the Manual under
Migration: Checklists KIX17 > KIX18	KIX Start KIX Pro KIX Cloud SSP	When migrating from KIX17 to KIX18, the checklists from old tickets are transferred to the new system. The checklist entries per ticket are retained.	• Admin Manual • KIX Start > Installation > Migration KIX17>KIX18
Customer Management: History of organisations and contacts	KIX Pro KIX Cloud	Analogous to the tab "History" for tickets, this information now also exists for organisations and contacts. By clicking on the "History" you can see which steps or edits of a contact or an organisation have taken place when and by whom.	• User Manual • KIX Pro > Organisation - Enhanced Function
Automation: Jobs - Filter conditions OR-linked	KIX Start KIX Pro KIX Cloud	The filters within a job configuration now also support the specification of several condition groups in a logical OR link. Thus, combined filter conditions of AND and OR links are possible.	• Admin Manual • KIX Start > Automation > Jobs > Creating or editing a job

Communication: Deposit team at email address	KIX Start KIX Pro KIX Cloud	A team can be stored at the system addresses. This enables the mailbox to independently determine the relevant team based on the recipient. The type of distribution can be defined on the e-mail account.	• Admin Manual
			  • KIX Start >
			  • KIX Pro >
			• User Manual
			  • KIX Start >
			  • KIX Pro >

Functionality	Affected Product	Brief Description	Described in the Manual under

SSP: Accessibility of the SSP2 via own FQDN	KIX Start KIX Pro KIX Cloud SSP	The new version of the Self Service Portal, SSP2 for short, can be accessed in the same way as the previous SSP1. SSP1 is still available as a fallback.	• Admin Manual
			   • KIX Start >
			   • KIX Pro >
			• SSP Manual
			   • Self Service Portal 2 > SSP2: Login
			   Self Service Portal 1 > SSP1: Login

Functionality	Affected Product	Brief Description	Described in the Manual under

KIX Connect Webservice	KIX Pro KIX Cloud Connect	The add-on KIX Connect Webservice has been extended by further macro actions:
		
		• **API call:** Direct access to API functions without explicitly sending a request to the backend.
		• **Post Attachment as Form Data:** Transfer of attachments to other web services.
		
		In addition, XSLT returns array and hash structures. If specific XSLT functions are called that return such complex data structures, the individual elements of the structures can be accessed.
		
		Furthermore, a data type can be specified in XSLT as which an attribute is represented in the JSON (numeric, string, boolean). Thus, incoming and outgoing data can be processed in XSLT despite non-XML-compliant attribute names.
			• Admin Manual
			  • KIX Pro > Add-ons > Connect
			  • KIX Pro > Add-ons > Connect > Connect Webservice

Workflow Rules	KIX Pro KIX Cloud SSP	The evaluation of workflows is only executed if valid rule sets are defined and workflow evaluation is activated in SysConfig.	• Admin Manual • KIX Pro > Workflow > Rule Sets

19.3   Release KIX18 v29 KIX MP

(released in September 2023)

Functionality	Affected Product	Brief Description	Described in the Manual under

Add-on KIX Maintenance Plan	KIX Pro KIX Cloud KIX MP	The add-on "KIX Maintenance Plan" supports you with recurring maintenance tasks. It allows you to manage recurring tasks for the devices, contracts or equipment documented in the asset management.	• Admin Manual
			  • KIX Pro > Add-ons > Maintenance Plan
			• User Manual
			  • KIX Start > Asset Dashboard > How to Create and Edit an Asset (Config Item)
			  • KIX Pro > Practice > Maintenance Plan

19.4   Release KIX18 v29

(Build:4276-0.1758-0 | SSP2: 116 | released in July 15th 2023)

Functionality	Affected Product	Brief Description	Described in the Manual under
Templates: Maintenance of system templates	KIX Pro KIX Cloud SSP	Templates can be created that are used exclusively by the system. These templates are used, for example, by the add-on "KIX Maintenance Plan" to automatically create tickets for upcoming maintenance tasks. There is also a macro action "Create Ticket from System Template". This can be used to create tickets based on a system ticket template and a reference article/ticket.	• Admin Manual • KIX Start > Appendix > Macro Actions > Overview of Macro Actions KIX Pro

Functionality	Affected Product	Brief Description	Described in the Manual under
Templates/Actions: Asset Placeholders in Jobs and Templates/Actions	KIX Start KIX Pro KIX Cloud SSP	Asset placeholders have been added to the KIX placeholders. Thus, placeholders for asset attributes can be used in an asset job or in the macro action "Create or Update Asset", e.g. in order to maintain data automatically. In the definition of ticket templates/actions, class-specific information of assets can be used, e.g. in the article text, in the title and in dynamic fields, in order to be able to create maintenance plan tasks automatically.	• Admin Manual • KIX Start > Appendix > Placeholders > Overview of Placeholders • KIX Start > Appendix > Macro Actions > Overview of Macro Actions KIX Pro
Dynamic fields: Type "FAQ Article Reference"	KIX Pro KIX Cloud SSP	The dynamic fields have been extended by the type "FAQ Article Reference" in order to be able to refer to individual FAQ entries at tickets, organisations or contacts. Future add-on "KIX Maintenance Plan": In maintenance plan and asset assignments, the relevant FAQ entries can be integrated into system ticket templates so that they are automatically available on the ticket when it is created.	• Admin Manual • KIX Start > System > Dynamic Fields > Object and field types of dynamic fields

Functionality	Affected Product	Brief Description	Described in the Manual under
SSP2: Template search for keywords	KIX Pro KIX Cloud SSP	In the future, the template search in SSP2 will support the search for keywords. That means: If keywords are stored in ticket templates, they can be searched for in the template search in SSP.	• KIX Self Service Portal • Self Service Portal 2 > SSP2: Tickets > SSP2:New Ticket
Change tab "Release Information" to "Welcome"	KIX Start KIX Pro KIX Cloud	The welcome page in KIX 18 now contains an information block on the latest releases with links to the corresponding forum posts.	• Admin Manual • KIX Start > Initial Setup

Ticket: Selection of agent/ responsible person in the drop-down menu	KIX Start KIX Pro KIX Cloud SSP	On opening the selection field "Agent" or "Responsible person", ten potential agents/responsible persons are displayed. No search pattern needs to be entered.	• Admin Manual
			   • KIX Start > Ticket > Teams > Creating or editing a team
			 • User Manual
			   • KIX Start > Ticket Dashboard > How to Create and Edit a Ticket

Ticket: Removal Ticket attribute "Age	KIX Start KIX Pro KIX Cloud SSP	The ticket attribute "Age" has been removed and is therefore no longer available in searches. In future, the ticket age will be determined from the difference between the time of ticket creation and the current time (NOW). Sorting by ticket age will in future be based on the time of ticket creation.	• Admin Manual
			   • KIX Start > Appendix > Overview of the search operators
			 • User Manual
			   • KIX Start > Tool Bar > Complex search

Functionality	Affected Product	Brief Description	Described in the Manual under
PDF printing: Contact data for ticket/article printing Print templates with contact data	KIX Start KIX Pro KIX Cloud SSP	As an agent user, I would like to print a ticket or an article as a PDF in the agent frontend, whereby the print result should also contain contact data (currently not the case).	• Admin Manual • KIX Start > System settings > Settings for tickets > Configuration PDF printing
Tickets: LoadingOptions for ticket table in ticket dashboard	KIX Start KIX Pro KIX Cloud SSP	LoadingOptions can be defined for the table in the ticket dashboard as well as for other contexts in order to pre-sort the tables and to be able to determine the maximum number to be loaded.	• Admin Manual • KIX Start > System settings > Configuration of User Interface (GUI) > Basics of GUI Configuration > The configuration of widgets > The table widget
FAQ: Delete item	KIX Start KIX Pro KIX Cloud	With the corresponding authorisation assigned by the admin, the "Delete" function is available to the agent.	• User Manual • KIX Start > FAQ Dashboard > How to Create an FAQ Entry and Edit it?
Service contract: Intuitive arrangement of the input fields	KIX Pro KIX Cloud	The new arrangement of the input fields when creating a service contract enables a better understanding of the correlations as well as a more intuitive input behaviour.	• User Manual • KIX Pro > Service Catalog Management > Service Contracts - Create and Edit

Functionality	Affected Product	Brief Description	Described in the Manual under
Ticket templates: Maintenance of keywords	KIX Pro KIX Cloud SSP	Ticket templates can now be provided with keywords. These are displayed in the Self Service Portal at the respective template. Users of the Self Service Portal can search for templates using these keywords.	• Admin Manual • KIX Pro > Workflow > Templates
Ticket templates: Usage context "System"	KIX Pro KIX Cloud	Ticket templates with usage context "System" can be created. These templates are used by the system to create tickets automatically (e.g. by the future add-on "KIX Maintenance Plan").	• Admin Manual • KIX Pro > Workflow > Templates

Templates/Actions: Relative Initialisation Date/Time Fields + Wait Time	KIX Pro KIX Cloud SSP	When defining templates and actions, time-based placeholders can be used. This makes it possible
		
		• to set relative time differences on dynamic fields of type Date or DateTime and wait times (at the time of use,
		• to create tickets that have a defined relative wait time or schedule start/end dates.
			• Admin Manual
			   • KIX Pro > Workflow > Actions > Create and configure actions
			   • KIX Pro > Workflow > Templates

Functionality	Affected Product	Brief Description	Described in the Manual under
Assets: Jumps (clickable links) to organisations, contacts and other assets in asset details possible	KIX Start KIX Pro KIX Cloud	In the version information of assets, various data are displayed as links. After clicking on this link, a new tab opens with the respective detailed view. This concerns • the assigned contact • the assigned organisation • references to other asset	• User Manual • KIX Start > Asset Dashboard > Asset Zoom View
Tickets: Change article sorting in communication history	KIX Start KIX Pro KIX Cloud	An the communication history, a button is available to sort the articles in ascending or descending order with one click.	• User Manual • KIX Start > Ticket Dashboard > Ticket Zoom View
Use email filter in SysMon processing	KIX Start KIX Pro KIX Cloud	SysMon messages can also be distributed by email filter without using the default value of SysMon processing.	• Admin Manual • KIX Start > Interfaces > System Monitoring
Organisations (e.g. objects): Configurable Display Values for Selection Values	KIX Start KIX Pro KIX Cloud SSP	The structure of the display values of objects (organisations, tickets, contacts) can be configured to provide individual information on the object at selection fields or in title lines.	• Admin Manual • KIX Start > System settings > Configuration of User Interface (GUI) > Configuration examples of GUI configuration > Individualize display values for objects

Functionality	Affected Product	Brief Description	Described in the Manual under
Templates/Actions: Either-Or for Context of Use	KIX Pro KIX Cloud SSP	To avoid misconfigurations, templates or actions can now only be assigned to one context of use, i.e. either Agent Portal or Self Service Portal.	• Admin Manual • KIX Pro > Workflow > Actions > Create and configure actions • KIX Pro > Workflow > Templates
Contacts: Removal of obsolete actions in detail views	KIX Start KIX Pro KIX Cloud	The detail view of contacts and organisations has been cleaned from obsolete actions.	• User Manual • KIX Start > Organistaions Dashboard > Organisation/Contact Zoom View
Personal settings: Visibility of teams based on basic permissions	KIX Start KIX Pro KIX Cloud	Agents are only shown in the personal settings under "My teams" the teams in which they can actively participate and process tickets according to their basic permissions.	• User Manual • KIX Start > Toolbar > Special Features
Bulk action for tickets	KIX Start KIX Pro KIX Cloud	Tickets that have already been combined cannot be combined again and are therefore not available in the selection for the collective action.	• User Manual • KIX Start > Ticket Dashboard

Functionality	Affected Product	Brief Description	Described in the Manual under
Notifications: Evaluation of Base/ Object Permissions	KIX Start KIX Pro KIX Cloud	When sending automated notifications, the base permissions are checked. Thus, agents only receive a notification if they have UPDATE (READ+WRITE) permission or READ permission on the specific ticket.	• Admin Manual • KIX Start > Automation > Notifications > Creating or Editing a Ticket Notification
AssignedConfigItem Mapping: Restriction to configurable deployment states	KIX Start KIX Pro KIX Cloud SSP	So that SSP users do not have to distinguish between current data and old data themselves, a static search criterion can be defined based on the deployment state. This allows you to specify, for example, that only assets with pre- or productive usage status are displayed or selectable in the SSP.	• Admin Manual • KIX Start > Assets - The Asset Device Database > Configuration settings

Functionality	Affected Product	Brief Description	Described in the Manual under

Contacts: Multiple email addresses for a contact	KIX Start KIX Pro KIX Cloud	An option, up to 5 additional email addresses can be stored on the contact. Thus, several email addresses of a contact can be synchronised from one LDAP/AD attribute via LDAP2Contact or AuthSync.	• Admin Manual
			  • KIX Start > User Management > Users > Creating and editing a user and assigning roles
			  • KIX Start > User Management > Users > Authentication/ Authorisation and Connection to Active Directory
			  • KIX Start > Appendix > Placeholders > Overview of KIX Placeholders
			  • KIX Start > Appendix > KIX Attributes
			  • KIX Pro > Jobs - Enhanced Functions > LDAP/AD-Synchronisation
			• User Manual
			  • KIX Start > Organistaions Dashboard > How to Create a New Contact and Edit Existing Ones

Functionality	Affected Product	Brief Description	Described in the Manual under

Contacts: Organisation requirement for unknown contacts no longer applies	KIX Start KIX Pro KIX Cloud	The organisation requirement for contacts will no longer apply in the future.
		
		For incoming emails, no separate organisation is created for an unknown contact (e.g. mails from private individuals/tenants). Therefore, the specification of an organisation is also obligatory for tickets. Even when migrating from other systems, contacts do not necessarily have to be assigned an organisation.
			• Admin Manual
			  • KIX Start > Installation > Migration KIX17>KIX18
			• User Manual
			  • KIX Start > Ticket Dashboard > How to Create and Edit a Ticket
			  • KIX Start > Organisations Dashboard > How to Create an Organisation and Edit this Entry
			  • KIX Start > Organisations Dashboard > How to Create a New Contact and Edit Existing Ones

Migration KIX 17 > KIX 18: root@localhost becomes admin	KIX Start KIX Pro KIX Cloud	When migrating a KIX17 environment to KIX18, the system user from KIX17 (UserID 1, usually "root@localhost") is not migrated. Instead, all its assignments are transferred to the system user of the KIX18 environment (login "admin").	• Admin Manual • KIX Start > Installation > Migration KIX 17 > KIX 18

20 Liability Disclaimer for KIX Start

20.1 Liability for Contents

The contents of our pages and documents have been prepared with the utmost care. Nevertheless, no liability can be accepted for any technical or editorial errors or omissions in this document. This also applies to any incidental or consequential damages that may arise from the provision, function or use of this material.

Please feel free to send any comments regarding design, additions or possible errors to our support team (https://forum.kixdesk.com) at any time. We will gladly take up and implement sensible suggestions and improvements.

All data, features and descriptions given in this work are subject to change at any time and without notice. Personal names and company names are fictitious. Any coincidences with real persons and companies are purely coincidental.

As a service provider, we are responsible for our own content on these pages in accordance with the general laws pursuant to § 7 para. 1 German Teleservices Act. However, according to §§ 8 to 10 German Teleservices Act, we are not obliged as a service provider to monitor transmitted or stored third-party information or to investigate circumstances that indicate illegal activity.

Obligations to remove or block the use of information in accordance with general laws remain unaffected by this. However, liability in this respect is only possible from the point in time at which a concrete infringement of the law becomes known. If we become aware of any such infringements, we will remove the relevant content immediately.

20.2 Liability for Links

Our website contains links to external websites of third parties over whose content we have no influence. Therefore, we do not assume any liability for these external contents. The respective provider or operator of the pages is always responsible for the content of the linked pages. The linked pages were checked for possible legal violations at the time of linking. Illegal contents were not recognisable at the time of linking.

A permanent control of the contents of the linked pages is not reasonable without concrete evidence of a violation of the law. If we become aware of any infringements of the law, we will remove such links immediately.

20.3 Copyright

The contents created by the site operators are subject to German copyright law. Duplication, processing, distribution, or any form of commercialization of such material beyond the scope of the copyright law shall require the prior written consent of its respective author or creator.

Insofar as the content on this site was not created by the operator, the copyrights of third parties are respected. In particular, third-party content is identified as such. Should you nevertheless become aware of a copyright infringement, please inform us accordingly. If we become aware of any infringements, we will remove such content immediately.

21 Purpose for which the use of KIX Start is intended within a medical context

KIX Start is not suitable, intended or approved for the identification, prevention, monitoring, treatment, relief or compensation of illnesses, injuries and disabilities. KIX Start is also not intended for the examination, replacement or modification of the anatomical structure or of a physiological process. KIX Start must also not be used to directly control a diagnostic or therapeutic product. Furthermore, KIX Start is not designed to be used in conjunction with a medical device with the intention of enabling the medical device to fulfill its intended purpose.

In a medical context, KIX Start is exclusively intended for carrying out administration and database-related tasks. If the above restrictions are complied with, KIX Start is suitable for use in a medical environment

- purely for documentation purposes, such as:
  - the general management of equipment in the form of managing and cataloging device data (device meta data) such as names, IP addresses, series numbers, persons responsible, guarantee periods, service providers, operating documents, license information, cost centers, as well as the management/organisation of users, device instructions;
  - the central documentation of all activities and changes in the IT such as due to executed maintenance activities or other service activities (e.g. medical device log book);
  - for compiling a knowledge database.
- for automating and simplifying general management processes, such as:
  - in service and technical customer service, for example in IT service (errors, changes, maintenance);
  - in building services (errors, changes, cleaning) or medical device technology.
- for monitoring purposes and calendar functions, such as:
  - for central IT services (network, email, data servers, SAP,...);
  - and for error and requirement notifications for the IT team, building services, medical device technology;
  - for the planning of regular maintenance works and reminders for replacing wear parts;
  - for the organisation of regular orders and planning the deployment of service technicians.

KIX Start is not designed for enabling or guaranteeing the functioning of medical devices and must therefore not be used for these purposes. If in the context of the aforementioned functions KIX Start also allows data exchange via an interface, please note that KIX Start must not be used for data modification or for any type of data control for medical or therapeutic purposes.

KIX Start may only be used in a medical context within the approved parameters mentioned above. KIX Service Software GmbH assumes no liability for any use that goes beyond or deviates from the approved parameters.