



KIX Pro

KIX 18 Admin Manual - EN

Issued on: 26.02.2024



Table of contents

1	Information on Using this Manual	8
2	Installing KIX Pro	9
2.1	Use the KIX.Cloud environment	9
2.2	Using KIX Pro On-Premises	9
2.3	Installing add-ons	11
3	Jobs - Enhanced Functions	12
3.1	Jobs initially delivered in KIX Pro	13
3.2	Duplicating a job	16
3.3	Change Asset Numbers during Import	18
3.4	Anonymisation	20
3.4.1	Adjust the "Anonymization" job	20
3.5	LDAP/AD-Synchronisation	22
3.5.1	Use synchronisation script or daemon	22
3.5.1.1	Automatic deactivation of contacts that are no longer available	23
3.5.2	Set up synchronization job	23
3.5.2.1	Parameters	25
3.5.2.2	Configuration notes	35
3.6	Automating Reports	36
3.6.1	Preconditions	36
3.6.2	Example configuration of the job	36
3.6.3	Configuration of the actions	37
3.6.3.1	1. Action "Create report"	37
3.6.3.2	2. Action "Extract Text"	37
3.6.3.3	3. Action "Execute Macro"	38
4	System - Enhanced Functions	41
4.1	Analyses	42
4.1.1	FE - Request Logs	43
4.1.1.1	Selection of log files	43
4.1.1.2	Metrics	43
4.1.1.3	Charts	44
4.1.1.4	Table	44
4.2	Config-Transfer	46
4.2.1	Export configurations	46



- 4.2.2 Import configurations 48
- 4.2.3 Notes on import and export 50
- 4.3 Dynamic Fields in KIX Pro 52
- 4.4 GUI Configuration..... 59
 - 4.4.1 The Explorer 60
 - 4.4.2 The JSON Editor..... 61
- 4.5 Icons 64
- 4.6 Migration KIX 17 65
 - 4.6.1 Perform the migration 66
 - 4.6.1.1 Step 1: Activate PSK mode in KIX 17..... 66
 - 4.6.1.2 Step 2: Start the migration 67
 - 4.6.1.3 Step 3: complete the migration..... 68
 - 4.6.2 Notes on migration 69
- 4.7 Plugins 70
- 4.8 Single Sign On (SSO) with Kerberos 71
 - 4.8.1 Connection 71
 - 4.8.2 Activate SSO in the frontend 73
 - 4.8.3 The keytab file..... 73
 - 4.8.4 Notes on Base64 coding 73
 - 4.8.5 Use / client-side requirements 74
 - 4.8.6 Frequently occurring error causes 74
 - 4.8.7 References 74
- 5 Services and SLA 75**
 - 5.1 Service Contracts..... 76
 - 5.1.1 Updating the service tree..... 76
 - 5.1.2 Steering of the "Affected Services" selection..... 76
 - 5.1.3 Affected Assets vs. Affected Services..... 78
 - 5.1.4 Criticality..... 79
 - 5.1.5 Import and export service contracts 79
 - 5.1.6 Use of post-productive services 80
 - 5.2 Service Level Agreements (SLAs)..... 81
 - 5.2.1 SLA criteria and attributes..... 81
 - 5.2.2 Scheme of an SLA..... 84
 - 5.2.3 Create or Edit an SLA 85
 - 5.2.4 SLA to Ticket 86
 - 5.2.5 SLA to Assets / Device SLA 86
 - 5.2.6 Notifications for SLA 87



- 5.2.7 Use of SLA in KIX placeholders..... 87
- 5.2.8 Configuration of the fulfillment time 88
- 5.2.9 Calendar configuration 89
- 5.3 Set response time when creating the ticket..... 90
 - 5.3.1 Scenario:..... 90
 - 5.3.2 Procedure: 90
 - 5.3.3 The configurations in detail..... 90
 - 5.3.3.1 1. Create dynamic field 90
 - 5.3.3.2 2. Create translation 91
 - 5.3.3.3 3. Extend template 91
 - 5.3.3.4 4. Create job 92
- 6 Self-service Portal 93**
 - 6.1 Accessibility of the Self Service Portal 94
 - 6.2 User settings for the SSP 96
 - 6.2.1 Set up user login 96
 - 6.2.2 Roles and permissions 97
 - 6.3 Control visibilities in the SSP 98
 - 6.3.1 Basic settings for visibilities 99
 - 6.3.2 Visibility of tickets and articles 101
 - 6.3.3 Visibility of follow-ups 102
 - 6.3.4 Asset visibility 104
 - 6.3.5 Visibility of FAQ articles 105
 - 6.3.6 Visibility of Dynamic Fields 106
 - 6.3.7 Visibility of news 107
 - 6.4 Ticket settings..... 109
 - 6.4.1 Fallback for ticket title 109
 - 6.4.2 Enable editing of ticket header attributes (permission) 109
 - 6.5 Templates, Actions, Rule Sets..... 111
 - 6.5.1 Provision of ticket actions..... 111
 - 6.5.2 Provision of article actions..... 112
 - 6.5.3 Provision of templates 112
 - 6.5.4 Workflow Rulesets 114
 - 6.6 GUI configuration of the Self Service Portal..... 115
 - 6.6.1 Dashboard Configurations 117
 - 6.6.1.1 Home Dashboard 117
 - 6.6.1.2 Ticket, Asset, FAQ Dashboard..... 118
 - 6.6.2 Configuration of the zoom views 118



- 6.6.2.1 Ticket Details..... 118
- 6.6.2.2 Asset Details 119
- 6.6.2.3 FAQ details 120
- 6.6.3 New ticket..... 121
- 6.6.4 Personal Preferences 121
 - 6.6.4.1 Enable/disable password change..... 122
 - 6.6.4.2 Enable/disable language selection..... 122
 - 6.6.4.3 Configuration of the Personal Data 123
- 6.6.5 Configuration examples 123
 - 6.6.5.1 Object-information-card-widget 123
 - 6.6.5.2 Table widget 124
 - 6.6.5.3 Implementing your own widget..... 125
- 6.7 Layout configuration..... 128
 - 6.7.1 Attributes in the layout configuration 128
 - 6.7.2 Organisation-specific layouts 130
 - 6.7.3 Example layout configuration 131
- 7 News 134**
- 8 Workflow 136**
 - 8.1 Actions..... 137
 - 8.1.1 Types of Actions 137
 - 8.1.1.1 Article actions 138
 - 8.1.2 Actions initially delivered in KIX Pro 139
 - 8.1.3 Create and configure actions..... 145
 - 8.1.3.1 Create, edit, duplicate and delete an action 145
 - 8.1.3.2 Configuration of an action..... 146
 - 8.2 Rule Sets..... 161
 - 8.2.1 Structure 163
 - 8.2.2 Sequence of processing 164
 - 8.2.2.1 Enable/disable processing of rulesets 165
 - 8.2.3 Create or edit a Ruleset 166
 - 8.2.4 Rulebook..... 167
 - 8.2.5 Overview of the commands 169
 - 8.2.5.1 Properties of the transaction object..... 171
 - 8.2.5.2 Restricting the context of use 172
 - 8.2.5.3 Operators..... 172
 - 8.2.5.4 Conditions (if)..... 173
 - 8.2.5.5 Statements (then) 177
 - 8.2.6 Functions..... 186
 - 8.2.6.1 DynamicFields.contains 186



- 8.3 Template groups 187
 - 8.3.1 Duplicating groups of templates..... 189
- 8.4 Templates..... 191
 - 8.4.1 Configure a Template 193
 - 8.4.1.1 Notes on field selection..... 203
 - 8.4.1.2 Notes on field options 204
 - 8.4.2 The initial standard template "Default - Ticket New Template" 207
 - 8.4.2.1 Initial configuration of the standard template 207
- 9 Additional configuration options 214**
 - 9.1 Configuring the Team View Mode 215
 - 9.1.1 List view..... 217
 - 9.1.2 Kanban view 217
 - 9.1.3 Calendar view 219
 - 9.1.4 Map view 220
 - 9.1.4.1 Updating geo-position data 223
 - 9.1.5 Enabling/disabling team view modes 224
 - 9.2 Time Recording 226
 - 9.2.1 Configuration example 228
 - 9.3 KIX Pro REST API 231
 - 9.4 Show child ticket tab in ticket details 232
 - 9.4.1 Integrate dynamic field "ChildTickets" into the dialogue "new ticket" 233
 - 9.4.2 Integrating the dynamic field "ChildTickets" into the "Edit Ticket" dialogue box 233
 - 9.4.3 Activate SysConfig key 234
 - 9.5 Configuring object history 235
- 10 Add-Ons 237**
 - 10.1 Connect 238
 - 10.1.1 Installation..... 239
 - 10.1.2 Common functional scope of the Connect Add-ons 239
 - 10.1.3 Field type "Data Source" 240
 - 10.1.3.1 Parameter..... 241
 - 10.1.4 Advanced Macro Actions 242
 - 10.1.4.1 XSL Transformation..... 242
 - 10.1.4.2 GetObjectData 252
 - 10.1.4.3 Get Item List From Data Source..... 253
 - 10.1.4.4 Get Item From Data Source..... 255
 - 10.1.5 Connect Baramundi 256
 - 10.1.5.1 Requirements 256



- 10.1.5.2 Use 257
- 10.1.5.3 Adjustments 260
- 10.1.6 Connect Database 263
 - 10.1.6.1 Requirements 263
 - 10.1.6.2 Establishment 263
 - 10.1.6.3 Use of Data Sources 272
 - 10.1.6.4 Use of the KIX Database..... 294
- 10.1.7 Connect Opsi 295
 - 10.1.7.1 Requirements 295
 - 10.1.7.2 Use 295
 - 10.1.7.3 Adjustments 298
- 10.1.8 Connect Webservice 299
 - 10.1.8.1 KIX as a provider - specific endpoints 299
 - 10.1.8.2 KIX as a requester - use of webhooks 308
- 10.2 Add-on "ITIL Practices" 321**
 - 10.2.1 Video 321
 - 10.2.2 Activate/Deactivate "ITIL Practices" 322
 - 10.2.2.1 Prerequisite 322
 - 10.2.2.2 Installation/System Update..... 322
 - 10.2.2.3 Activate the "ITIL Practices" add-on 322
 - 10.2.2.4 Reset the ITIL Practices add-on 324
 - 10.2.3 Implementation of ITIL practices by Means of Processes 325
 - 10.2.3.1 Ticket Templates 325
 - 10.2.3.2 Ticket Actions 327
 - 10.2.3.3 Permissions..... 329
 - 10.2.3.4 Process 1: Service Request..... 330
 - 10.2.3.5 Process 2: Incident 334
 - 10.2.3.6 Process 3: Problem..... 338
 - 10.2.3.7 Process 4: Change 341
 - 10.2.3.8 Process 5: Business Continuity Preparation 345
 - 10.2.4 Advanced Configuration 347
 - 10.2.4.1 Actions for "ITIL Practices" 348
 - 10.2.4.2 Assets for "ITIL Practices" 350
 - 10.2.4.3 Dynamic Fields for "ITIL Practices" 351
 - 10.2.4.4 FAQ for "ITIL Practices" 352
 - 10.2.4.5 Jobs for "ITIL Practices" 353
 - 10.2.4.6 Reference Data for "ITIL Practices" 354
 - 10.2.4.7 Report Definitions for "ITIL Practices" 355
 - 10.2.4.8 Templates for "ITIL Practices" 356
- 10.3 Maintenance Plan 357**
 - 10.3.1 Prerequisite 357



10.3.2	Fundamentals	358
10.3.2.1	Maintenance service.....	359
10.3.2.2	Maintenance asset	359
10.3.3	Maintenance plan	359
10.3.3.1	Mapping.....	359
10.3.3.2	Scheduling.....	359
10.3.3.3	Ticket templates	360
10.3.4	Maintenance tasks.....	360
10.3.5	Maintenance ticket	361
10.3.6	Administration.....	361
10.3.6.1	Installation/system update	362
10.3.6.2	Authorisation roles	362
10.3.6.3	Ticket templates	362
10.3.6.4	Update maintenance tasks.....	364
10.3.6.5	Functional enhancements.....	364
11	Practice	365
11.1	Periodic job "Licence renewal"	366
11.1.1	Preparation.....	366
11.1.2	Configure job.....	367
11.1.3	Execute job.....	369
11.2	Use of checklists.....	370
11.2.1	Checklists in practice	372
11.2.2	Checklists in KIX Pro.....	373
11.2.3	Configuration of checklists	373
11.2.4	References	374
11.2.5	Configuring and providing checklists	376
11.2.5.1	1. Create a checklist	376
11.2.5.2	2. Provide checklist.....	380
11.2.5.3	3. Show checklist status.....	386
11.2.5.4	References:.....	395
11.2.6	Data structure of checklists	397
12	Liability Disclaimer KIX Pro	399
12.1	Liability for Contents.....	399
12.2	Liability for Links	399
12.3	Copyright	400
13	Purpose for which the use of KIX Pro is intended within a medical context	401



1 Information on Using this Manual

To ensure transparency and ease of use, only settings and applications specific to **KIX Pro version 18** are explained in this manual. Please refer to the KIX Start manual for all basic functions.

The manual for the agents and KIX-users can be found at <https://docs.kixdesk.com/> . Here you will also find all other documentation on KIX versions 17 and 18.

We invite you to become part of our KIX community and to support us with helpful tips in the further development of KIX and user information.

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

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 Installing KIX Pro

KIX Pro is an add-on to KIX Start with an extended range of functions. You can choose between different ways of using KIX 18:

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

2.1 Use the KIX.Cloud environment

No installation is required to use a KIX.Cloud environment. KIX.Cloud is therefore also suitable as a test environment.

Apply for the provision of KIX.Cloud at www.kixdesk.com¹. We will send you the link to your KIX portal and the access data for the initial user (admin) by email. Open the link and log in with the access data you received from us.

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

2.2 Using KIX Pro On-Premises

To use KIX Pro locally, you only need to replace the public Docker registry with your individual repository ID in the `environment` file. We will send you the repository ID by email after you have subscribed to KIX Pro.

Attention!

The repository ID sent to you is intended exclusively for use by you or your organisation. Therefore, do not pass on the repository ID and keep it safe from unauthorised access. If unauthorised persons come into possession of your repository ID, they could gain access to your system and your data.

To use KIX Pro, proceed as follows:

1. First install KIX Start - if it is not already installed. You will find instructions on how to do this in the Start manual under Installation.
2. Open the `environment` file.
3. Put a hash (#) at the beginning of the line "`REGISTRY=docker-registry.kixdesk.com/public`"

This comments out the reference to the public registry so that this information is ignored when the programme is started.

¹ <http://www.kixdesk.com>



- In the next but one line, replace < YOURREPOSITORYHERE > with your individual repository ID and remove the hash (#) at the beginning of the line.
By removing the hash, this line will be taken into account when the programme starts.
- The result looks like this:

```
Specification of the repository ID in the environment file
1 # -----
2 # basic configuration
3 # -----
4
5 # the docker registry to use
6 #REGISTRY=docker-registry.kixdesk.com/public
7
8 # the following applies for KIX Pro customers
9 REGISTRY=docker-registry.kixdesk.com/customers/
10 4d11xo22xxxxxxxxxxxxxxxxxxxxff29
11 # the image tag to use for all application images
12 IMAGE_TAG=stable
13 ...
```

- Run an update (see also Admin Manual of KIX Start: Installation)
- The next time you start KIX you can use KIX Pro.
If necessary, use the Setup Assistant to set a new admin password and a "Super User" and to carry out the basic configuration of KIX.

Alternatively, you can initialise KIX Pro via the console. To do this, call up the following command:

```
Console command for initialising KIX Pro
root@dockerhost:/opt/kix-on-premise/deploy/linux/# sed -i.orig 's/REGISTRY=docker-
registry.kixdesk.com/public/REGISTRY=docker-registry.kixdesk.com\
public\nREGISTRY=docker-registry.kixdesk.com/customers\/<YOUR_REPOSITORY_ID_HERE>/g'
environment
```

Further information on installing KIX Pro can also be found on GitHub at:

- Linux: <https://github.com/kix-service-software/kix-on-premise/blob/master/deploy/linux/README.md>
- Windows: <https://github.com/kix-service-software/kix-on-premise/blob/master/deploy/windows/README.md>



2.3 Installing add-ons

The additional modules and extensions of KIX are assembled in the customer-specific registry and are directly available. As a rule, this takes place one working day after the order has been received and processed. The updated images are installed in the same way as a version update:

```
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./stop.sh
```

```
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./update.sh
```

Beyond that, no further general steps are required. Depending on the add-on, new setting options are offered in the configuration wizard in the admin area.

The provision of AddOns in the KIX Cloud is carried out by our support team.

3 Jobs - Enhanced Functions

Menu	KIX > Automation > Jobs
-------------	-------------------------

KIX Pro contains a range of preconfigured jobs and offers you the option of creating jobs with extended functions. KIX Pro also offers the option of duplicating jobs (see below).

Job types in KIX Pro

In addition to ticket-related jobs, you can also create asset-related jobs and synchronisation jobs in KIX Pro. This is defined in the *Automation > Jobs* menu by selecting the job type.

Asset

The job type "Asset" offers you the possibility to create jobs on assets. You can use this job type, for example, to adapt the asset numbers assigned by KIX to your own number range after importing assets.

Contact

The "Contact" job type is applied to contacts. You can use this, for example, to create a new ticket as soon as a certain date is reached on the contact (start of an offboarding process when an employee leaves).

Synchronisation

With the help of the job type "Synchronisation" you can synchronise both user and contact data from LDAP/AD directories with KIX. In doing so, KIX fetches the data from the specified directories, stores it in KIX and synchronises it via an event- or time-based job. You can read how to set up the job under "[Synchronisation \(see page 22\)](#)".

Reports

With the help of the job type "Reports", you can automatically generate reports and then send them in a new ticket. KIX Pro offers the possibility to create reports in different output formats such as CSV, JSON, HTML, Excel (XLSX), AtomFeed, XML and PDF. It is thus possible to read these reports into other (external) systems and process them further. KIX Pro also offers additional MacroActions to extract the content of an automated report and send it directly in the message text of a ticket. An example of this can be found here, in the [Admin Manual of KIX Pro \(see page 36\)](#). The basic functions for creating report definitions and reports as well as for the automated sending of a report as a ticket attachment can be found in the admin manual of KIX Start (chapter KIX Modules).

Ticket

With the job type "Ticket", you can create jobs on tickets and their articles. This standard function is described in the KIX Start manual in the chapter Jobs.

3.1 Jobs initially delivered in KIX Pro

Job	Description	Notes
Anonymisation	With this job, you can have tickets anonymised from a predefined period of time, e.g. in order to archive them in compliance with data protection regulations. You can specify at which time or event the anonymisation is to be carried out and which data is to be anonymised. This job is already prepared for you, but is set to "invalid". If you want to use the job, set it to "valid" and adjust the configuration further if necessary.	See also chapter " Anonymisation (see page 20) "
Auto Set Planned Effort (Incident)	Sets a default value as the default for the planned fulfilment time (target time). Applies to tickets of the type "Incident". Initial value: 30 minutes	<p>The job automatically sets the target time defined in the job in the dynamic field "PlannedEffort".</p> <p>You can change the configuration of the job by, for example</p> <ul style="list-style-type: none"> • changing the value of the target time • defining further events that trigger the job • using filters to define the conditions under which the job should set the target time • and much more.
Auto Set Planned Effort (Service Request)	Sets a default value as the default for the planned fulfilment time (target time). Applies to tickets of the type "Service request". Initial value: 60 minutes	

Job	Description	Notes
<p>Create FAQ Suggestion</p>	<p>The job creates a new FAQ entry from an article if the dynamic field "CreateFAQSuggestion" is set to "yes" (= 1st action).</p> <p>The 2nd action resets the value of the dynamic field "CreateFAQSuggestion" for a new use (empty value).</p> <p>Info: The field "CreateFAQSuggestion" is only available in the dialogues "Edit ticket" and "Close ticket" if the channel "Note" or "E-mail" is selected.</p>	<p>If necessary, you can change the default configurations in the 1st action to add other values to the FAQ entry or assign the FAQ entry to a different category.</p> <p>The following parameters, among others, are initially set in the job:</p> <ul style="list-style-type: none"> • Title: Subject of the first article • Category: Misc • Cause: Content of the first article on the ticket • Solution: Content of the item triggering the job. • Dynamic fields: Related tickets → ID of the ticket (establishes the link between ticket and FAQ).
<p>KIX Field Agent - Mobile Processing Rejected</p>	<p>Sets the agent in the ticket and the blocking status in the ticket if the ticket is rejected in the Field Agent app.</p>	<p>You can set the agent to be set and add further macro actions.</p>
<p>Put affected assets back into operational state</p>	<p>This job automatically sets the incident status "Operational" for all affected assets of an existing ticket if:</p> <ul style="list-style-type: none"> • the ticket of the type "Incident" is closed • or the solution SLA is fulfilled ("satisfied") • AND no further open/unresolved incident tickets exist for this asset. 	<p>Do not change the configuration! System-relevant functions are linked to it.</p>



Job	Description	Notes
Put affected assets into incident state	<p>This job automatically sets the incident status "Incident" for all affected assets of an existing ticket when:</p> <ul style="list-style-type: none">• a ticket of the type "incident" is created• the solution SLA is not yet fulfilled ("satisfied")• and one or more assets are entered in "Affected Asset".	<p>Do not change the configuration! System-relevant functions are linked to it.</p>
SLA First Response Time Fulfillment	<p>This job sets a timestamp in a ticket when an item is created that is visible in the Self Service Portal. It thus defines the time of the first response for SLA fulfilment. You can reconfigure the job as needed.</p>	<p>The action "Set fulfilment time" should not be removed, otherwise the time stamp will not be set.</p>
SLA Solution time fulfillment	<p>This job sets a timestamp in a ticket when a ticket receives the status "removed", "closed" or "summarised". It thus defines the time of resolution for SLA fulfilment. You can reconfigure the job as needed.</p>	<p>The action "Set fulfilment time" should not be removed, otherwise the time stamp will not be set.</p>

Job	Description	Notes
TicketMerge	<p>This job is required for the initial action "Merge". This opens a dialogue to merge 2 tickets. The job is triggered when the action is saved.</p> <p>Thereby</p> <ul style="list-style-type: none"> • the source ticket is set to the status "merged" • the event "TicketMerge" is triggered on the source ticket • the properties configured in the action are transferred <p>If the channel "e-mail" or "note" is selected in the action, the item will also be written to the target ticket.</p> <p>If a ticket has the status "merged", no admin-configurable ticket and item actions will be displayed on this ticket.</p> <p>The initial job configuration provides for the source tickets to be appended to the target ticket. In doing so, the values of the target ticket remain and the source ticket loses its values (such as priority, agent, etc.).</p>	<p>You can reconfigure the action and decide which attributes of the source ticket are merged with the target ticket:</p> <p>1st action: initial "Ticket Merge" (may not be changed).</p> <p>Target ticket: The ticket number of the target ticket or the name of the dynamic field containing the ID of the target ticket (initially the corresponding dynamic field "MergeToTicket").</p> <p>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. (Attention: Translations of the attribute names are not possible).</p> <p>Dynamic fields: Comma-separated list of the names of the dynamic fields that will be transferred to the target ticket (Attention: Translations of the names are not possible).</p> <p>Create values for dynamic fields:</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Overwrites the values of the listed dynamic fields in the destination ticket with the values of the source ticket. <input type="checkbox"/> 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.

3.2 Duplicating a job

KIX Pro offers the option of duplicating jobs. This allows you to use existing jobs as a template for new jobs, e.g. to execute different actions under the same conditions.



In the zoom view of a job, you will find the "Duplicate" action. After clicking on the button, a new job is created based on the current job. This opens a form with the name "Copy of [name of duplicated job]". You can change the name of the duplicate.

The form fields of the duplicate are filled with the values of the duplicated job and can be changed by you as required.

The Macro Actions are not referenced, but newly created. This means that you can also change the macro actions in the duplicate individually.

When the duplicate is saved, an independent new job is created without references to the original job. This means that a change in the duplicate does not change anything in the duplicated job.

3.3 Change Asset Numbers during Import

When importing, assets are stored under an asset number generated by KIX. KIX Pro offers the option of replacing the asset numbers with your own identifiers when importing. This is done with an event-based job of the type "Asset".

To configure a job to change the asset numbers:

1. In Explorer, navigate to *Automation > Jobs*. A table is opened in the content area, which lists all jobs created in the system.
2. Click on "New Job" in the table. A form dialog opens in which you can create the synchronization job step by step. Use the small blue arrow or dot buttons to go to the next step or to switch between steps.
 - **Job information** (step 1):
 - Select the option "Asset" under "JobType"
 - Give the job a meaningful name.
 - Describe the job in the "Comment" field (optional)
 - Set the job to "valid".
 - **Execution plan** (step 2):
 - Select the "VersionCreate" event under "Event Based Execution".
This specifies that the job replaces the number whenever the version of an asset changes.
3. **Filter** (step 3):
 - Set filters to determine the conditions under which the job is executed (optional). The job is only applied to the assets that meet all of the filter criteria.
If no filters are set, the job is carried out for all imported assets.
The asset attributes of the selected asset class are available for selection.

• **Actions** (step 4):

- Select "Set asset number" as the first action.
- Enter the identifier to be set under "Asset number". You can use letters and numbers as well as their combination.

Placeholders are also possible:

Placeholder	Description
<KIX_ASSET_CurrentVersion_[AttributeName]>	Reference to attributes from the asset version (e.g. "Name")
<KIX_ASSET_CurrentVersion_[Data_AttributeKey]>	Reference to attributes from the asset class definition
<KIX_ASSET_CurrentVersion_Data_[KEY]>	for XML data (class-specific)
<KIX_ASSET_CurrentVersion_Name>::<<KIX_ASSET_CurrentVersion_Data_RoomLabel>::<<KIX_ASSET_CurrentVersion_Data_Floor>	Example entry if the asset belongs to the "Room" class

 **Hint**

If the value to be set is a CMDB-wide unique identifier (asset number), the KIX asset number is replaced by the value.

If an asset number is not unique or if no value is set for the relevant attribute, the existing asset number is not replaced and an error message is entered in the log ("ERROR" level).

3.4 Anonymisation

You can have selected data on tickets anonymized. For example, to anonymize personal data when archiving tickets in accordance with GDPR. The anonymization takes place via the time-controlled or event-controlled execution of a job (menu *Automation > Jobs*). With the help of the initially delivered "Anonymization" job, you can anonymize the following ticket data:

- Editor
- Responsible person
- Contact
- Organisation
- History entries
- Creation and modification date for tickets
- Creation and modification date for articles
- E-mail addresses stored on articles.

In addition, a dynamic "Anonymize Ticket" field is provided on the ticket. The anonymization job is initially configured in such a way that all tickets are processed for which this field has the value "ToDo". In the delivery state, you cannot set the value of the field manually, as this is not configured in any ticket dialog. This is a safety measure to prevent accidental data loss. You can configure a new job that fills the "Anonymize Ticket" field with the criteria you have specified.

3.4.1 Adjust the "Anonymization" job

The "anonymization" job is delivered inactive and is filled with sample values. Edit the configuration of the job to adapt it to your needs.

In step 1 (Job Information):

Define the validity of the job. Only jobs set to "valid" are executed.

- Initial configuration: "invalid" (set the job to "valid" if necessary).

In step 2 (Execution plan):

Determine the time or event at which the anonymization should take place. You can change or add to the configuration by specifying a time for the execution of the job or by selecting another event that triggers the job.

- Initial configuration:
 - Event-controlled execution
 - when changing the dynamic field "Anonymized Ticket" ("TicketDynamicFieldUpdate_AnonymiseTicket")



In step 3 (Filter):

Determine which tickets should be anonymized. You can add or change filter criteria to narrow the selection of tickets.

- Initial configuration: The job is triggered when the dynamic field "Anonymize Ticket" has the value "ToDo".

In step 4 (Actions):

Specify which ticket data should be overwritten with which content. You can customize the configuration and add further macro actions, for example by moving all anonymized tickets to the "Archive" team.

- Initial configuration:
 - Set agent: 1 (admin)
 - Set responsible: 1 (admin)
 - Set contact: 1 (admin @ localhost)
 - Set customer: 1 (MY_ORGA)
 - Delete history: Replace "OwnerUpdate", "ResponsibleUpdate", "CustomerUpdate", "TicketLinkAdd", "TicketLinkDelete", "SendAnswer", "SendAgentNotification", "SendCustomerNotification", "EmailAgent", "EmailCustomer", "FollowUp", "Forward ", "LoopProtection", "Subscribe" & "Unsubscribe" by: "Comment replaced by" Anonymisation "job."
 - Created by and modified by set: 1 (admin)
 - Set created by and modified by for article: 1 (admin)
 - Set article email attributes:
 - Bcc: admin @ localhost
 - Cc: admin @ localhost
 - By: admin @ localhost
 - To: admin @ localhost
 - Set dynamic field: AnonymizeTicket: 2
 - The dynamic field "Anonymized Ticket" is set to the value "Done" and the ticket is marked as "is anonymous". This is displayed in the agent portal interface so that you can also search for all anonymized tickets.

3.5 LDAP/AD-Synchronisation

You can fill the contact and user data in KIX Pro with data from the LDAP/Active

Content on this page:

- [Use synchronisation script or daemon \(see page 22\)](#)
 - [Automatic deactivation of contacts that are no longer available \(see page 23\)](#)
- [Set up synchronization job \(see page 23\)](#)
 - [Parameters \(see page 25\)](#)
 - [Configuration notes \(see page 35\)](#)

Directory. This data is used as

- Contacts in the ticket and asset area
- Customer user ("User" with "IsCustomer")
- Agent user ("User" with "IsAgent").

Depending on the available LDAP source, data synchronisation can be carried out in different ways:

- Synchronisation script or daemon via console command and SysConfig key
- Automated via event- or time-based synchronisation jobs

Prerequisite: At least one LDAP-based directory service that is accessible from the backend of KIX.

Warning

Changes to the contact data must then be made exclusively in LDAP/AD. Changes made in KIX are overwritten with the synchronisation.

Information on connecting LDAP/AD hosts can also be found in the KIX Start manual.

3.5.1 Use synchronisation script or daemon

Console commando	Console::Command::Maint::Auth::Sync::Synchronize
SysConfig key	Daemon::SchedulerCronTaskManager::Task###AuthSynchronize

You can use a script or daemon to simplify the AD connection if all contacts from LDAP should also be available as users in KIX. Setting up an LDAP2Contact job can thus be omitted.

The console command "`Console::Command::Maint::Auth::Sync::Synchronize`" starts a synchronisation script that obtains the contacts available in KIX exclusively from LDAP sources and simultaneously makes them available as users in the system. When the command is executed, all configured auth. sync backends are queried. The users and contacts determined from this are created in KIX and assigned to their authorisation contexts and roles (see Authentication/Authorisation and Connection Active Directory). The console can be found in the menu *System > Console*.

If the synchronisation is to be automated periodically, you can activate the SysConfig key: "`Daemon::SchedulerCronTaskManager::Task###AuthSynchronize`" (menu *System > SysConfig*). Please note the timing with which the execution takes place.

 **Info**

If the SysConfig key "`Daemon::SchedulerCronTaskManager::Task###AuthSynchronize`" is set to valid, a reload of the frontend configuration is not sufficient. The daemon must be restarted. Use the console command: "`Console::command::admin::daemon::Reload`".

3.5.1.1 Automatic deactivation of contacts that are no longer available

You can also use the script or daemon to automatically deactivate users and contacts that are no longer available in the LDAP sources. This can happen, for example, in the case of incomplete offboarding processes in which adjacent tools are not informed. Since these entries are no longer updated in KIX, they are not deactivated. Deactivation can be done manually. Alternatively, the command "`Console::Command::Maint::Auth::Sync::Synchronize`" can be executed with the option "`--invalidate-unsynced`". The script deactivates all users and contacts that can no longer be obtained from one of the Auth. sync sources. To prevent system accesses or manually created users from being subject to this behaviour, selected login identifiers can be excluded. These login identifiers are defined in the SysConfig key "`Maint::Auth::Sync::Synchronize::Skip`" using regular expressions. **Do not remove the initial entries!**

3.5.2 Set up synchronization job

You can directly connect specific LDAP sources to periodically synchronise the contact data they contain without resorting to authentication synchronisation. To do this, set up a synchronisation job as described below.

Several LDAP/AD hosts in different formats can be specified per job. In addition, directory services that use a size limit for the result set ("max size exceeded") can also be queried. By specifying the request page size, blockwise processing can be performed.

Set up the synchronisation job:

1. Navigate in the explorer to *Automation > Jobs*. A table opens in the content area listing all the jobs created in the system.
2. Click on "New Job" in the table. A form dialogue opens in which you can create the synchronisation job step by step. Use the small blue arrow or dot buttons to go to the next step or to switch between steps.
 - **Step 1** (Job Information):
 - Select the option "Synchronisation" under "JobType".
 - Give the job a meaningful name.
 - Describe the job in the "Comment" field (optional).
 - Set the job to "valid".
 - **Step 2** (Execution Plan):
 - Select when the synchronisation should take place (weekdays and time).
 - **Step 3** (Actions):
 - Select the macro action "LDAP2Contact". Further input fields are displayed.
 - Enter the connection data to the LDAP/AD server as well as the attributes which are to be synchronised with the LDAP/AD server (see table below).
 - You can optionally enter additional hosts.
 - Click on . A 2nd action opens.
 - Select "LDAP2Contact" again and enter the connection data and attributes here as well.
 - For a better overview, you can expand and collapse the input fields of the actions. To do this, click on the small arrow button next to the plus symbol  .
 - Save the job by clicking on "Save". The synchronisation is carried out at the specified time.

 Note

If an LDAP2Contact-Sync job is executed and the user already exists, the user found is updated and the job continues with the next user entry.

3.5.2.1 Parameters

Parameter	Meaning	Example
Skip	If the check mark is set here, the corresponding action is not carried out. This allows you to (temporarily) exclude individual actions from synchronisation without having to reconfigure the job. If several actions are configured in the job (e.g. two different LDAP servers), the other actions continue to run normally.	
Host	FQDN which the LDAP server uses for synchronisation.	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;"> <code>ldap.exampleCompany.com</code> </div>
BaseDN	Entry point into the directory structure	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;"> <code>dc=exapleCompany,dc=com</code> </div>

Parameter	Meaning	Example
<p>Contact / User Sync Map</p>	<p>JSON string containing the mapping from KIX attributes to AD/LDAP attributes. The mapping must have the following form:</p> <pre data-bbox="363 533 1035 719"> { "KIXAttributName" : "LDAPAttributName", "KIXAttributName" : "LDAPAttributName" } </pre> <p>All attributes specified in the mapping must exist. Otherwise, no correct assignment can take place. Also make sure that the spelling is correct.</p> <p>The contact entry is created or updated based on the values determined or fixed from the LDAP/AD. The specification of the following attributes is required as a minimum:</p> <ul data-bbox="379 1037 533 1187" style="list-style-type: none"> • Email • Firstname • Lastname • UserLogin <p>Attention: If this information is not set in the LDAP/AD entry (empty), no contact entry can be created or updated.</p> <p>Concrete values are indicated with a preceding "SET :." (e.g.: "Email1": "SET:mail@example.com").</p> <p>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.</p>	<pre data-bbox="1070 398 1423 1792"> { "Email": "mail", "Email1": "SET:mail@example.co m", "Email3": "mail", "Email5": "automatedmail", "Title": "title", "Firstname": "givenname", "Lastname": "sn", "Street": "streetAddress", "City": "l", "Zip": "postalCode", "Phone": "telephoneNumber", "Mobile": "mobile", "Fax": "facsimileTelephoNeN umber", "UserLogin": "sAMAccountName", "IsAgent": "SET:1", "IsCustomer": "SET:0", "PrimaryOrganisation ID": "department", "OrganisationIDs": ["department", "SET:123"], "DynamicField_XYZ": "SET:ActiveDirectory 1", "CONCAT: {postalCode}-{l}-DE" } </pre>

Parameter	Meaning	Example
	<p>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. Prerequisite: The target attribute must support the supplied data type and the definition of the dynamic field.</p> <p>The array values are concatenated as a single value with a separator symbol (analogue "CONCAT" → see below). Note: the "mail" attribute supports a maximum of 6 values in the array.</p> <ul style="list-style-type: none"> • Directive: "ARRAYJOIN[<SeparatorString>]:{<ADAttributeName>}" <ul style="list-style-type: none"> • ARRAYJOIN : Command for creating a concatenated character string (array) • SeparatorString : Specification of the separator (comma, semicolon, hyphen, etc. including any spaces). Specification in square brackets. • ADAttributename : Name of the LDAP/AD attribute to be concatenated. The attribute is specified in curly brackets. • Example: "objectClass" is an attribute that occurs several times in LDAP/AD. <ul style="list-style-type: none"> • 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". 	

Parameter	Meaning	Example
	<p>The assignment to a primary organisation is defined by means of " <code>PrimaryOrganisationID</code> ". In addition to an LDAP/AD attribute, a fixed value can also be configured ("<code>PrimaryOrganisationID</code>": "<code>SET:123</code>"). "123" is interpreted as the ID of an organisation known in KIX. However, if an LDAP/AD attribute is specified ("<code>PrimaryOrganisationID</code>": "<code>department</code>"), the value of the attribute is interpreted as a customer number and evaluated by lookup to a KIX-internal organisation ID. If nothing is 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.</p> <p>The assignment to <i>n</i> further organisations is defined by means of " <code>OrganisationIDs</code> ". For this purpose, an array of LDAP/AD attribute names or fixed organisation ID assignments is specified. This allows <i>n</i> LDAP attributes or SET values to be configured. The lookup behaviour is identical to " <code>PrimaryOrganisationID</code> ". For each match, the information stored at the contact is completely replaced. If the " <code>OrganisationIDs</code> " specification is not maintained in the mapping, the value specified in <code>PrimaryOrganisationID</code> is automatically appended to any existing entries in the list of organisations. In this case, no complete replacement takes place.</p> <p>If another user of the system is to be activated via the Agent or Self Services Portal, the parameters " <code>IsAgent</code> " or " <code>IsCustomer</code> " must be set to "1"; a fixed value assignment can be entered for this ("<code>IsAgent</code>": "<code>SET:1</code>" or "<code>IsCustomer</code>": "<code>SET:0</code>"). The other required authorisation roles are to be stored in the authentication and authorisation configuration.</p>	

Parameter	Meaning	Example
	<p>Dynamic fields can be filled with attribute values from the LDAP/AD. Thus, for example, dynamic fields of the object types contact or organisation can be used to</p> <ul style="list-style-type: none"> • store the AD from which the contact originates in a contact entry • to provide the identifiers for organisation and contact data of other systems also in KIX. <p>KIX provides the dynamic fields "Type" and "Source" (KIX Pro)" for this purpose.</p> <p>Concatenation (chaining) of LDAP attributes: Each KIX attribute can be composed of n LDAP attributes and separator strings by means of " <code>CONCAT</code> ", e.g. to combine several LDAP attributes in one KIX contact attribute. Instead of specifying an LDAP attribute directly, these are named in curly brackets (e.g. " <code>PrimaryOrganisationID</code>": " <code>CONCAT: {givenName} from the {company}-{department}</code> ").</p> <p>All characters not contained within curly brackets are transferred directly into the contact attribute (e.g. " <code>CONCAT: {postalCode}-{l}-DE</code> " becomes "09113-Chemnitz-DE").</p>	

Parameter	Meaning	Example
UID	<p>Name of the LDAP attribute used to uniquely identify an LDAP entry (user login). If no <i>UserLogin</i> is specified in the Sync Map, a <i>UserLogin</i> is created in KIXDB based on the <i>UID</i> and set to invalid. <i>UID</i> and <i>UserLogin</i> can point to the same attribute without any problems.</p> <p>Tip: By setting the attribute " <code>AuthAtt</code> " in the SysConfig key "<code>Authentication###000-Default</code>" you can set an alternative identifier for the <i>UserLogin</i>. If this is set to "mail", for example, a user can log in with the mail address stored in the system. If the user data is also synchronised with the LDAP by job, the user can always log in with his current mail address, even if it has changed, e.g. due to a name change. If logging in via the mail address fails, the UID/user login stored in the system is used as a fallback for authentication.</p> <p>Please note: Updating a contact attribute "Email" is only possible if a user entry exists in KIX. Otherwise, when an email address is changed in AD/LDAP, a new contact is created.</p>	<div style="border: 1px solid #ccc; padding: 5px; width: fit-content; margin: 10px auto;"> UID sAMAccountName </div>
Params	<p>JSON string for LDAP connection parameters</p> <p>Default:</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px auto; width: fit-content;"> <pre> { "port": "", "version": "3", "timeout": "120", "async": "0" } </pre> </div>	<div style="border: 1px solid #ccc; padding: 10px; margin: 10px auto; width: fit-content;"> <pre> { "port": "389", "version": "3", "timeout": "60", "async": "0" } </pre> </div>



Parameter	Meaning	Example
Search User DN	User DN of the user with which KIX connects to the LDAP server to perform the search.	<code>cn=kix,cn=user,dc=exampleCompany,dc=com</code>
Search User Password	Password of the user with which KIX connects to the LDAP server to perform the search.	*****
Limit GroupDN	Allows you to restrict the entries to be queried to objects that are contained in the specified group. This may be necessary if the group membership cannot be restricted via a direct filter on the contact entry.	
Access Attribute	Defines LDAP-attribute which is used to check group membership. Uses <code>memberUid</code> if empty.	<code>member memberUid ...</code>
User Attribute	Determines whether the user distinguished name ("DN") or the LDAP/AD attribute defined in the "UID" parameter is used to verify group membership.	<code>DN UID</code>



Parameter	Meaning	Example
Always Filter	<p>Permanent LDAP filter applied to all requests.</p> <p>Example: all non-deactivated user accounts:</p> <pre>(&(objectClass=user)(!(userAccountControl:1.2.840.113556.1.4.803:=2)))</pre> <p>Example: all user entries with "departmentNumber" and "mail" set: <code>codeb</code></p> <pre>(&(objectClass=user)(departmentNumber=*)(mail=*))</pre> <p>You can also enter group filters. You can find more examples here: http://www.selfadsi.de/ldap-filter.htm.</p>	<pre>(&(mail=*)(sAMAccountName=*))</pre>
Destination Charset	<p>Character set into which the LDAP search results are converted.</p> <p>Default: utf-8</p>	<code>utf-8</code>
Page Size	<p>If necessary, you can split the total number of data to be synchronised into several data packages of "n" data records each. For example, if you receive a message from the server that the response data set is too large ("max size exceeded"). Enter the number of data records that are transmitted per request. Processing then takes place in blocks of packets with, for example, 20 data records.</p>	<code>20</code>

Parameter	Meaning	Example
Debug	<p>Deactivate/Activate debug outputs.</p> <p>If the parameter is active, essential steps of the LDAP/AD data synchronization are noted in the <u>job log</u> (job history).</p> <p>Notes:</p> <ul style="list-style-type: none"> • This option should only be activated temporarily, since otherwise the history entries become very extensive and generate corresponding memory requirements. • The SysConfig key "<i>Automation::MinimumLogLevel</i>" must be set to the value " <code>Debug</code> ". Otherwise, only the error messages delivered by the macro action will be captured, but all other debug information will not. 	yes no
Dry Run	<p>Deactivate/activate dry run (test run).</p> <p>If the parameter is activated, no actual data synchronisation takes place. However, the data determined from the LDAP/AD source is noted in the job log as if an update were taking place. The data synchronisation can thus be simulated in advance for test purposes.</p> <p>Note: Like Debug, this option should only be activated temporarily, as otherwise the history entries become very extensive and generate corresponding memory requirements. It is still recommended to use this function in conjunction with AlwaysFilter to check specific or a small number of LDAP entries.</p>	

 **Info**

- If the mapping contains the user attribute "Login", a user is also synchronised.
- An existing contact is updated when the job is executed based on the mapping.
- A contact that does not exist will be created when the job is executed.

 **Notes**

- The user login is not directly affected by the synchronisation, but the "AuthSync" function is.
- During the automatic import, data restrictions of the AD must be taken into account (blockwise import).
- The single point of trust is the external system.



3.5.2.2 Configuration notes

You can specify a preset value instead of an LDAP attribute for certain KIX attributes on a contact/user. This is done via the keyword "SET:", which is included before the value to be set. The keyword is case sensitive.

Example: SyncMap with fixed parameters

```
{  
  "Email": "mail",  
  "Title": "title",  
  "Firstname": "givenname",  
  "Lastname": "sn",  
  "Street": "streetAddress",  
  "City": "l",  
  "Zip": "postalCode",  
  "Phone": "telephoneNumber",  
  "Mobile": "mobile",  
  "Fax": "facsimileTelephoneNumber",  
  "UserLogin": "sAMAccountName",  
  "IsAgent": "SET:1",  
  "IsCustomer": "SET:0",  
  "PrimaryOrganisationID": "department"  
}
```



3.6 Automating Reports

KIX Pro allows to create reports in further output formats (Atom Feed, Excel, HTML, JSON, PDF, XML). By extracting the report contents, they can be directly inserted into the message text of a ticket and/or sent as an article attachment.

In the following example, a job of the type "Reporting" is created, which creates an HTML report every Monday morning and sends the report content in the message text of the ticket. The job has the following tasks (in required order):

1. Create a report in HTML format every Monday at 6:00 am.
2. Extract the HTML table from the report
3. Send a new ticket containing the HTML table of the report in the message body.

The possibility to send a report as an article attachment is already given in KIX Start. An example of sending a report as an article attachment can therefore be found in the Admin Manual of KIX Start.

3.6.1 Preconditions

- The job requires an existing report definition as a basis for creating the report. If necessary, create a report definition for this purpose.
- Since the HTML table of the report is to be extracted, it must be possible to create the report as an HTML file. For this purpose, "HTML" must be specified as the output format in the report definition.

3.6.2 Example configuration of the job

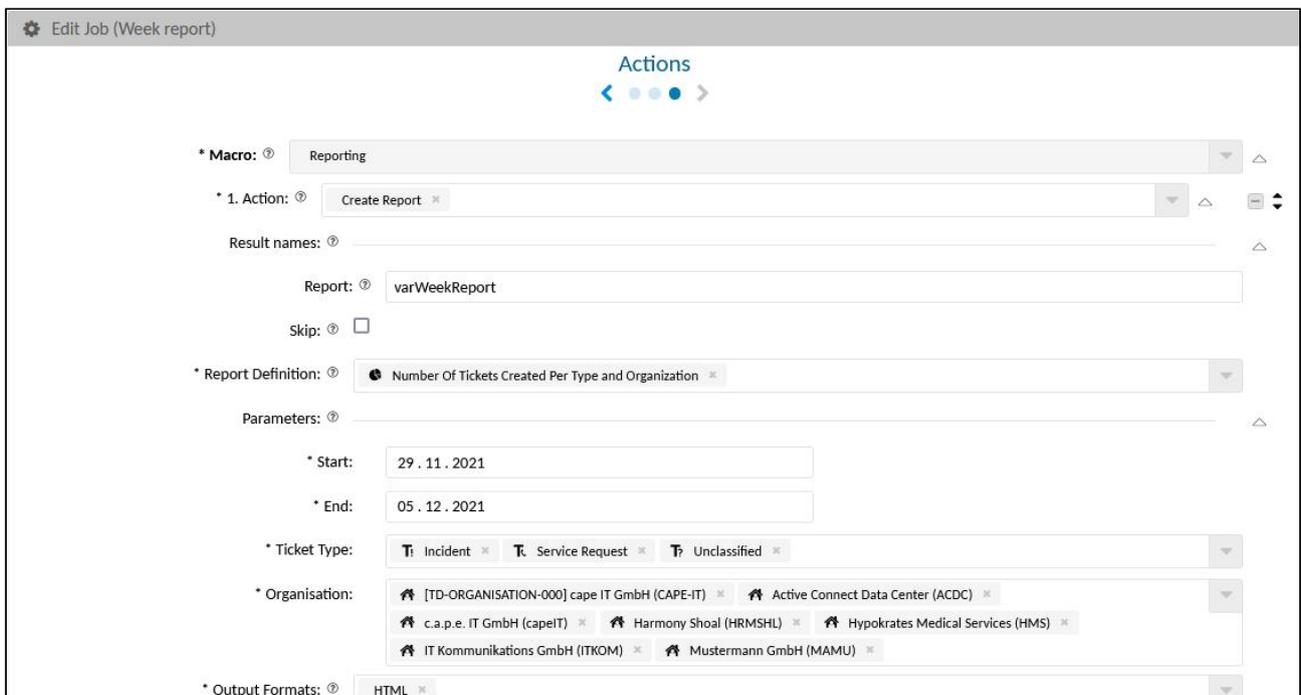
- Job Information:
 - Job Type: Reporting
 - Name: NameofJob (e.g. weekly report)
 - Validity: valid
- Execution Plan:
 - Weekday(s): Monday
 - Time: 6:00 a.m.
 - Events: none
- Filter: none or as required
- Actions (in required order)
 1. Action: "Create Report" - Creates the report based on its report definition.
 2. Action: "Extract Text" - Extracts the HTML table from the generated report.
 3. Action: "Execute Macro" - Executes a macro. (here: create new ticket).

3.6.3 Configuration of the actions

3.6.3.1 1. Action "Create report"

The action "Create report" creates the report based on the selected report definition. The parameters set in the report definition are included. Enter the appropriate parameter values so that the job can set these values when creating the report. You can only specify output formats that were defined in the report definition. In the example, the report is output as HTML so that the HTML table can be extracted with the next action.

In the example, the report is stored in the object variable `varWeekReport` ("Report" field). If no variable is declined, the report is stored in the variable `Report`. Then the created report can be referenced with `Report.Results:n.Attribute`.



The screenshot shows the configuration for the 'Create Report' action within a macro named 'Reporting'. The configuration includes the following fields:

- Macro:** Reporting
- 1. Action:** Create Report
- Result names:** (empty)
- Report:** varWeekReport
- Skip:**
- Report Definition:** Number Of Tickets Created Per Type and Organization
- Parameters:**
 - Start:** 29.11.2021
 - End:** 05.12.2021
 - Ticket Type:** Incident, Service Request, Unclassified
 - Organisation:** [TD-ORGANISATION-000] cape IT GmbH (CAPE-IT), Active Connect Data Center (ACDC), c.a.p.e. IT GmbH (capeIT), Harmony Shoal (HRMSHL), Hypokrates Medical Services (HMS), IT Kommunikations GmbH (ITKOM), Mustermann GmbH (MAMU)
- Output Formats:** HTML

3.6.3.2 2. Action "Extract Text"

This action can extract text using RegEx. Depending on the regular expression used, both the whole text and individual parts of the text can be extracted. The text to be extracted must already exist (e.g. in a variable).

In the example, the HTML table of the report is extracted and stored in the variable `varTableWeekReport` declared under "ExtractedText". `varTableWeekReport` is later referenced in the action "Execute Macro" to output the extracted text in the message text.

You can use the expression specified under "RegEx" to restrict the text to be extracted. In the example, no restrictions take place; the entire body of the HTML report is extracted. The expression is thus:

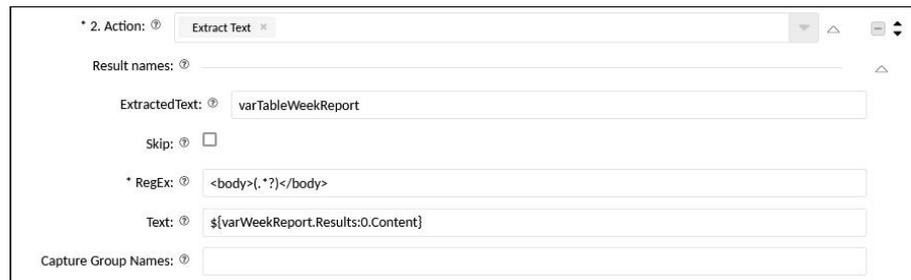
```
<body>(.*?)</body> .
```

The variable specified under "Text" defines where the text is extracted from. In the example, this is the object variable declined in the 1st action.

Variables can represent complex objects. The result properties are accessed using a dot (.). If the property is an array, an element of the array can be accessed using " :<index> ". The variable is thus composed as follows:

```
`${varWeekReport.Results:0.Content}`:
```

- The variable `varWeekReport` is returned by the action "Create Report".
 - The variable `varWeekReport` has the property `Results`. `Results` is an array that contains the reports in the respective output formats (HTML, CSV, JSON, XLSX, etc.).
 - by accessing `:0`, the first output format is used (here: HTML)
 - The `content` property of the output format is used. This refers to the entire file content. In the example: the entire HTML document.



3.6.3.3 3. Action "Execute Macro"

This action executes further macros. In the example, a new ticket is created with the following information:

- the required ticket details such as status, priority, subject, channel, etc.
- by specifying the variable ``${varWeekReport.1}``, the extracted text (see 2. action "Extract text") is output in the message content.

* 3. Action: Execute Macro

Skip:

ObjectID:

* Macro: Ticket

* 1. Action: Create Ticket

Result names:

NewArticleID:

NewTicketID:

Skip:

Channel:

Sender Type:

From:

To:

Cc:

Bcc:

Enable for Self Service Portal:

* Body:
B I U S x₂ x² I_x 

 Styles - Format - Font - Size - A - A - 

 Week report:
 Below is the current weekly report:
 \${varTableWeekReport.1}

Result:

After the job has been executed, a new ticket exists. The ticket contains the HTML table of the report in the message text.

Article Overview [1]

No.	New	Sender Type	From	Char...	Subject	Created at	Attachm...
1		agent	not assigned <admin@localhost>		WeekReport	12/13/2021, 01:59 PM	

From: not assigned <admin@localhost>
 Subject: WeekReport

WeekReport:

Organisation	Unclassified	Incident	Service Request	Total
My Organisation	225	1		226
[TD-ORGANISATION-000] cape IT GmbH	5	14		19
[TD-ORGANISATION-001] Springfield Nuclear Plant	27	28		55
Total	257	43		300



i Info

The validation of the macro actions only takes place when the job is executed. Notes on possible errors can be found in the kix.log (menu *System > Logs*) or in the job.log (tab "History" in the Job Details).



4 System - Enhanced Functions

KIX Pro and the KIX additional modules have enhanced configuration options in the "System" menu.

Dynamic fields

The overview lists and describes all dynamic fields that are initially delivered with KIX Pro. In addition, other field types are available in KIX Pro for the configuration of individual dynamic fields.

GUI configuration

For a convenient configuration of the user interface of the agent portal, KIX Pro provides a JSON editor. The JSON editor allows editing of the SysConfig keys in prepared form incl. syntax highlighting.

Icons

In the *System > Icons* menu you will find a list of all the icons in the system. The overview can help you to find icons and you can change existing icons.

Configuration transfer

KIX Pro enables the transfer of complete object configurations from one KIX environment to another. For example, configurations can be transferred from test to productive systems or from on-premises installations to the KIX Cloud.

Migration

KIX 18 Pro users can automatically transfer data from KIX 17 to KIX 18. This is done conveniently via the user interface in the *System > Migration KIX 17* menu. 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).

Single Sign On via Kerberos

When calling up the agent or self-service portal, the user can be authenticated via Kerberos. This means that the user does not have to log on to KIX with a user name and password if he is already authenticated by his login to the domain (= Single Sign On).



4.1 Analyses

You will find tools for analysing the system in the Analysis menu.

4.1.1 FE - Request Logs

Menu	<ul style="list-style-type: none"> • KIX > System > Logs • KIX > System > Analysen > FE-Request Logs
-------------	---

KIX logs the HTTP requests and saves them in a log file. One request log is written per day. The HTTP request log files can be found in the *System > Logs > Frontend Server* menu. Here you can display and download the log files. The log files are available in CSV format. The separator is the tabulator (\t).

In addition, KIX Pro provides you with a support tool that gives you an overview of the current HTTP requests on the system. This can be found in the menu *System > Analyses > FE request logs*. The metrics of the request logs for the front end are analysed and displayed graphically for you. You can select individual or multiple log files and use them to analyse the performance.

4.1.1.1 Selection of log files

The selection field offers you the HTTP request log files stored in the log for selection. Select which log files you would like to analyse. The selected log files are analysed together and the result is displayed in the metrics, charts and table.

4.1.1.2 Metrics

The table provides you with an overview of the calculated metrics and their determined values:

Metric	Description
Total Request Count	How many requests are there in total?
Average Request Duration	Average runtime of a request
Fastest Request	Which is the fastest request? In brackets: Duration of the request in ms
Slowest Request	Which is the slowest request? In brackets: Duration of the request in ms
Most Frequented Resource	Which resource is the most frequented/penetrated? In brackets: Number of requests

Metric	Description
Requests/Second (last minute)	Average number of requests per second within one minute The time in the selected log files is analysed.
Requests/Second (last 5 minutes)	Average number of requests per second within 5 minutes The time in the selected log files is analysed.
Requests/Second (last 10 minutes)	Average number of requests per second within 10 minutes The time in the selected log files is analysed.

4.1.1.3 Charts

The charts visualize the calculated values:

Chart	Description
Request Count	24-hour chart to determine the peak times, grouped into 30-minute packages. Shows how many requests took place at what time.
Request Duration in ms	Runtime chart Display of the duration of the requests, grouped in 50 ms Shows the number of requests within a specific time spectrum
Request Count in Last x Minutes	Number of requests in the last 1 5 15 minutes (relative to the selected log files)

4.1.1.4 Table

The table lists the log entries and displays additional comments on the requests. Anomalies are highlighted in color:

- Yellow: Request took longer than 500 ms
- Red: Incorrect request



Column	Description
Date	Time of the log entry
Operation	The method used
Status	HTTP status code for recognizing successful or faulty operations
Remarks	Notes on certain specifications of the request <ul style="list-style-type: none">• expand: Request contains "expand"• include: Request contains "include"• limit =0: Request contains no limit• no search: Request contains filters, but no search• no search limit: Request does not contain a search limit
Duration	Duration of the request
Resource	Display of the affected resource
Parameter	Display of the parameters in the request

4.2 Config-Transfer

In the menu *System > Config-Transfer* you can transfer complete object configurations from one environment to another. Both the export from KIX and the import to KIX are possible.

This enables:

- the transfer of configurations from test to productive systems or from on-premises installations to the KIX Cloud
- the roll-out of pre-configured system settings by support or customer advisory services
- the transfer of configurations for setting up, using and displaying dynamic fields
- and much more.

You can import or export the configurations of the following objects:

- Jobs
- Dynamic fields
- Report definitions
- Templates and Template groups
- Actions
- Teams
- Ticket status and types
- System configurations

Note: Currently, configurations can only be imported and exported in JSON format.

4.2.1 Export configurations

The export of configurations is done in the menu *System > Config-Transfer > Export*.

In the dialogue, define the output format in which the export is to take place and which data sets are to be exported. You can define the data sets to be exported by selecting the object (e.g. Report definition) and by setting filters (for details see the following table).

The export starts after a click on the "Export" button. Exporting can take some time depending on the number of selected objects and the data records stored in them. A file is then generated in the selected output format and made available for download via the functions of the browser used.

Inhalte auf dieser Seite:

- [Export configurations \(see page 46\)](#)
- [Import configurations \(see page 48\)](#)
- [Notes on import and export \(see page 50\)](#)

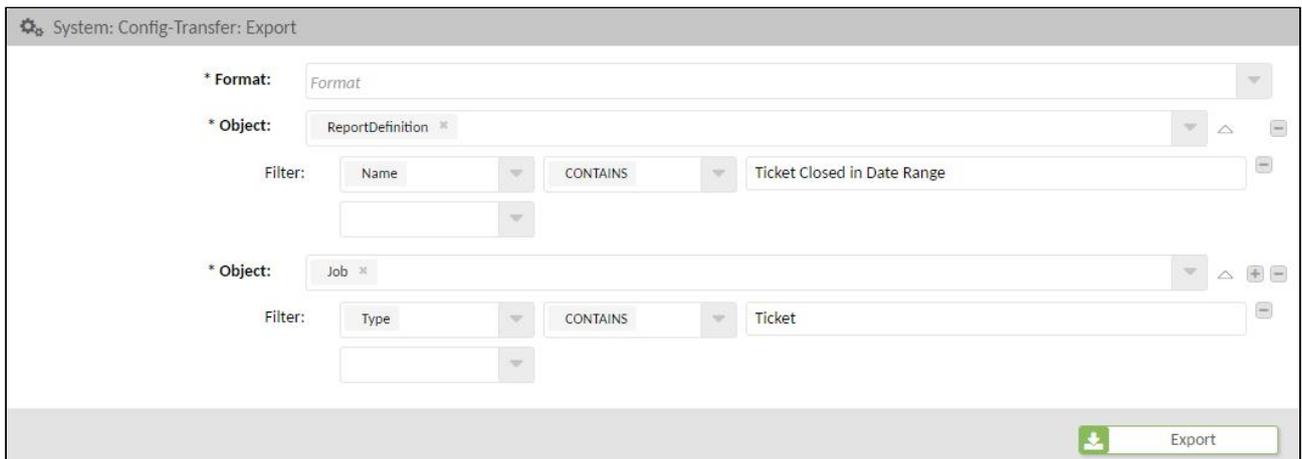


Fig.: Object selection for export

The form dialogue contains the following input fields:

Field	Description
Format	Select the output format to which the configuration is to be exported (currently only JSON).
Object	Select which object type you want to export. <ul style="list-style-type: none"> You can select several objects one after the other. To do so, click on the plus button. If no object is selected, all available configurations are exported. The plus button must not yet be clicked. Both valid and (temporarily) invalid objects are exported.
Filter	You can specify the data records to be exported by specifying filter criteria. <ul style="list-style-type: none"> All data records of the object that match the selected filter criteria are exported. The available filter criteria depend on the respective object type. If no filter is set, all permitted data records of the selected object are exported.
	Add or remove objects and filters.

 **Important**

Authorisation assignments stored in the configurations are currently **not** exported and must be set manually.

4.2.2 Import configurations

The import of configurations is done in the menu *System > Config-Transfer > Import* by file upload. The file must contain the data records for the import and be in a supported format (see above: Export -> Format). Depending on the file size, the import may take a moment. Please also note the notes on import and export listed below!

Important

- The import does not create any new objects. Therefore, make sure that all objects to be imported exist in KIX, e.g. the status type specified in the filter of ObjectActions. If necessary, create the objects in KIX beforehand.
- For a correct and complete display, please delete the cache after the import using the console command "Console::Command::Maint::Cache::Delete" and carry out a logout and logon.

The form dialogue contains the following input fields:

File	Description	
Modus	The selected mode determines how the data records to be imported are handled. For example, whether a new record is created if the identifier is identical or whether an existing record is updated.	
	Default	Update or add records if they exist or not (default).
	Only Update	Only updates existing data sets.
	Only Add	A record is only added if it does not exist.
	Force Add	<ul style="list-style-type: none"> • A record is copied if it already exists. The record name is preceded by "Copy". • Records that do not exist are added.

File	Description
File	Select the file to be uploaded. Only 1 file can be uploaded. Hint: You can also drag and drop the file onto the button.

After the import, a results table is displayed with the following information:

System: Config-Transfer: Import: Result							
Objekt	Einträge	Ignored	Updated	Added	Force Added	Failed	
DynamicField	27	0	25	0	0	2	
Job	9	0	9	0	0	0	
ObjectAction	11	0	11	0	0	0	
ReportDefinition	6	0	6	0	0	0	
Template	13	0	13	0	0	0	



Fig.: Result table of the import

Field	Description
Object	The object whose records were imported.
Entries	Number of imported records
Ignored	Number of ignored data records Data records are ignored if, for example, they already exist in the "OnlyAdd" mode or do not exist in the "OnlyUpdate" mode.
Updated	Number of updated records
Added	Number of records added
Force Added	Number of data sets copied by the "ForceAdd" mode and marked with "Copy" (except for ObjectIcons - see below). This basically only affects the data sets that already exist in the system.
Failed	Number of records that could not be imported. The reasons for this can be found under <i>System > Logs</i> .

4.2.3 Notes on import and export

The export always considers the basic configurations. Individual parameters (e.g. icons of teams) are not taken into account.

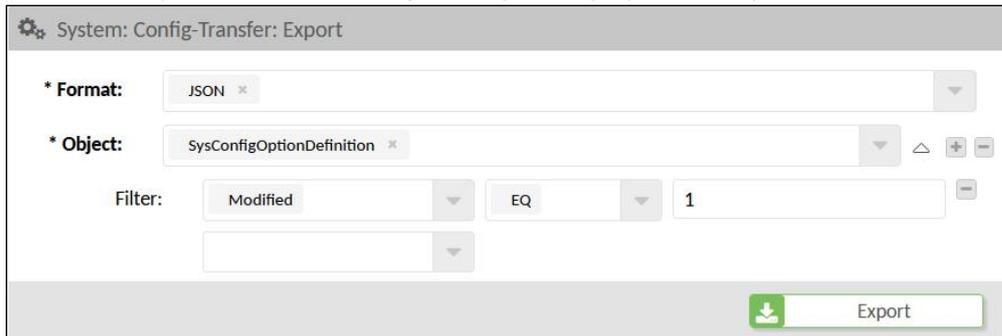
Tree structures (e.g. teams, statuses, template groups) are exported completely, i.e. including all their sub-categories. The starting point is always the root team. When exporting sub-categories, their parent elements are also taken into account so that the categories can be correctly arranged in the tree during (re-)import.

Team configurations:

- The calendar is only recorded as a calendar number (name).
- Icons are not considered.
- The sender e-mail addresses (system addresses) are exported in processed form.

SysConfig configurations:

- Enables the export of SysConfig settings whose key corresponds to certain name patterns (wildcard pattern).
- Allows the export of all modified SysConfig settings ("modified")



- The import processes configuration files (JSON) that were previously created by the export. After the import, the changed configuration is active.
- When using the "ForceAdd" mode, existing configurations are ignored because configurations cannot be copied and prefixed with "Copy_".
- **Important:** When importing, the applicability of plug-in-specific configurations (KIX Connect/ KIX Pro) is NOT checked. The responsibility lies with the executing administrator.
 - In case of serious errors, the SysConfig can be completely reset to default. To do this, execute the console commands: " Console::Command::Maint::Config::CleanUp " and " Console::Command::Maint::Config::Rebuild " in succession.
- After the import, the frontend must be loaded manually. To do this, navigate to the menu System > SysConfig and click on "Reload frontend configurations" in the overview.



ObjectIcons

- This object cannot execute a ForceAdd or only partially.
 - If the data set does not exist, it is treated normally as an add.
 - A copy is not possible because there is no name for the icon.
- The uniqueness is determined via the parameters Object and ObjectID.
 - Object: Object assignment for the icon where it is to be applied.
 - E.g. TicketState, TicketType, GeneralCatalogItem, MIMEType, etc.
 - ObjectID: Is the ID of the respective object.
- This means that the prefix "Copy" cannot be added to either ObjectID or Object, otherwise the icon can never be displayed because the assignment is incorrect.

E-mail filter

- ⚠ Please note, especially when using IDs, that e-mail header values to be imported are not checked for plausibility on the target system!
- The PostMasterHeader "X-KIX-TicketTemplate" is checked.
 - If *Match* or *Set* contain this, the filter will not be imported.
 - PostMasterHeaders which are not known to the system will be added to the "PostmasterX-Header", if it is not "X-KIX-TicketTemplate".



4.3 Dynamic Fields in KIX Pro

KIX Pro and some additional modules use additional dynamic fields, which are initially delivered with KIX Pro. 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.

The dynamic fields initially delivered with KIX Start can be found in the KIX Start admin manual.

Pay attention to 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 organizations.

The integration of dynamic fields in the ticket creation masks of KIX Pro is done quickly and conveniently via the configuration of templates (see page 191). The configuration via the SysConfig key can thus be omitted, but serves as a fallback. To integrate dynamic fields in the "New Ticket" dialog, you can adapt the "Default - New Ticket Template" accordingly. Dynamic fields can be integrated in all other interfaces via the GUI configuration. Use the configuration examples from KIX Start as a guide.

Field name	Field type	Object type	Description	Configuration options
AffectedServices	AssetReference	Ticket	Adopts the ticket service assignments when migrating from KIX 17 to KIX 18. If services are stored on a ticket, they are displayed in the ticket details.	not mandatory However, the usage status and the maximum number of fields can be changed if necessary.
AnonymiseTicket	AssetReference	Ticket	Is used internally to identify the anonymization status of tickets. The status is set by the "anonymization" job.	not mandatory

Field name	Field type	Object type	Description	Configuration options
ChildTickets	TicketReference	Ticket	<p>Is used to link tickets with child tickets (sub-tickets).</p> <p>When linking tickets, agents can choose the type of link (parent child normal).</p> <p>The configuration determines which properties tickets must have in order for them to be available for selection in the list. The initial configuration allows the entry of a maximum of 25 tickets with the status "new", "open", "waiting for reminder and" waiting for automatic closing. "The tickets are linked as child tickets.</p>	<ul style="list-style-type: none"> • Limitation of the number of selectable tickets (Count Max) • Add or remove status • Limitation of the selectable tickets to certain ticket types (TicketTypes)
CloseCode	Selection	Ticket	<p>Is used for the selection list "Completion Code" in the close dialog.</p> <p>The configuration defines the content of the selection list.</p> <p>The field is used by the "Close" action, which opens the close dialog. The action can be reconfigured in the <i>Ticket > Actions</i> menu.</p>	<ul style="list-style-type: none"> • Change the name of the list entries • Adding and removing list entries

Field name	Field type	Object type	Description	Configuration options
CreateFAQSuggestion	Selection	Ticket	<p>Used in connection with the job "Create FAQ Suggestion".</p> <p>The field is available as a selection field in the dialogues "Edit ticket" and "Close ticket" if the channel "Note" or "E-mail" is selected.</p> <p>If the option "yes" is selected, the job creates a new FAQ entry specifying the information used in the ticket (see also Jobs - Enhanced Functions (see page 12)).</p>	not mandatory
MergeToTicket	TicketReference	Ticket	<p>Is used to group tickets, e.g. to attach a misdirected email to a ticket.</p> <p>The field is integrated in the dialog of the initial action (see page 137) "Merge (see page 142)". It references all tickets from which a ticket can be selected as the destination ticket. The action can be reconfigured if necessary (see actions (see page 137)).</p> <p>The grouping of the tickets is triggered by the initial job "TicketMerge" when the action is saved. The job can also be reconfigured (see page 12).</p>	The configuration must not be changed!



Field name	Field type	Object type	Description	Configuration options
ParentTickets	TicketReference	Ticket	<p>Is used to link tickets with parent tickets (superordinate tickets). Refers to the parent ticket.</p> <p>When linking tickets, agents can choose the type of link (parent child normal).</p> <p>The configuration determines which properties tickets must have in order for them to be available for selection in the list. The initial configuration allows the entry of a maximum of 5 tickets with the status "new", "open", "waiting for reminder and" waiting for automatic closing. "The tickets are linked as parent tickets.</p>	<ul style="list-style-type: none">• Limitation of the number of selectable parent tickets (Count Max)• Add or remove status• Limitation of the selectable tickets to certain ticket types (TicketTypes)



Field name	Field type	Object type	Description	Configuration options
PlannedEffort	Text	Ticket	<p>Contains the value for the planned time expenditure (target time). The target time forms the basis for time recording. It defines the time in which a ticket should be processed.</p> <p>Used by the Auto Set Planned Effort (Incident) and Auto Set Planned Effort (Service Request) jobs and by the Planned Effort action:</p> <ul style="list-style-type: none">• The jobs set the value of the target time in the dynamic field. The default value can be changed in the jobs.• The "Planned effort" action opens a dialog that contains the dynamic field including the target time set by the job. Agents can change this specified target time manually for each ticket.	The configuration must not be changed!



Field name	Field type	Object type	Description	Configuration options
RelatedTickets	TicketReference	Ticket	<p>Is required in the ticket edit mask for the selection of the related tickets.</p> <p>The configuration determines which properties tickets must have in order for them to be available for selection in the list. The initial configuration allows the entry of a maximum of 25 tickets with the status "new", "open", "wait as a reminder and" wait for automatic closing. "The selected tickets are linked as" normal ".</p>	<ul style="list-style-type: none">• Limitation of the number of selectable tickets (Count Max)• Add or remove status• Limitation of the selectable tickets to certain ticket types (TicketTypes)
RelatedNewsTickets	TicketReference	News	Used by the News module to refer to related tickets in News.	Initially, a maximum of 5 tickets can be selected. You can change this number as required.
Satisfaction Points	Selection	Ticket	<p>Selection field for evaluation points.</p> <p>Is required in the Self Service Portal for the "Customer Feedback" action. Allows you to specify a ranking.</p>	not mandatory
Satisfaction Remark	Textarea	Ticket	<p>Comment on the evaluation points.</p> <p>Is required in the Self Service Portal for the "Customer Feedback" action. Allows a comment on the ranking.</p>	not mandatory



Field name	Field type	Object type	Description	Configuration options
Type	Selection	Organisation	<p>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.</p> <p>The field can be specified in the UserSyncMap (see page 22), for example, in order to record organisational data from the LDAP/AD. With that it is possible</p> <ul style="list-style-type: none">• to store at an organisation from which AD the organisation originates• to provide the identifier for organisational data of other systems in KIX as well.	<ul style="list-style-type: none">• Define the number of selectable entries• Define the selectable entries
WorkOrder	Textarea	Ticket	<p>Is required for the ticket action "AppAssignWorkOrder (see page 140)" and forms the "Work instruction" input field in this dialog.</p>	not mandatory

4.4 GUI Configuration

KIX Pro administrators can easily customize the agent portal's user interface in the *System > GUI configuration > Agent portal* menu. Here all the keys relevant for the GUI configuration are clearly listed in an Explorer (1). An integrated JSON editor (2) makes editing easier and the default value of the configuration key can be displayed in parallel for comparison (3).

The basics and principles of the GUI configuration correspond to those described in the KIX 18 Start admin manual.

Note: The Explorer only contains the configuration keys that are relevant for adapting the **user interface in the agent portal**. All other configuration keys must be adjusted in the *System > SysConfig* menu.

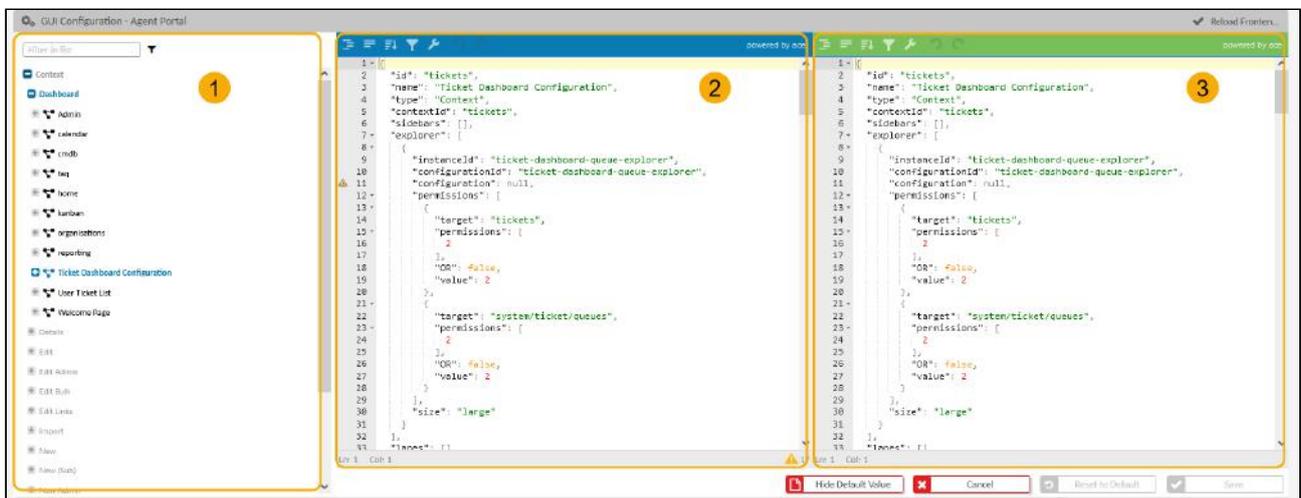


Fig. : Structure of the GUI configuration

⚠ Caution!

Changes to the GUI configuration can cause **serious system errors**. You should therefore have a thorough knowledge of the system and JSON when making changes to the GUI configuration. Please contact our support if you need help or if you would like us to adapt the user interface.

4.4.1 The Explorer

The explorer contains the configurations relevant for the user interface of the agent portal. The tree structure corresponds to the configuration hierarchy on which the GUI configuration is based.

The roots in the configurations tree are:

- Context
 - Contains all configurations that are assigned to a context: e. g. Dashboards, detail pages, dialogs etc.
- Shape
 - Contains all configurations that are assigned to a form: e. g. Dialogs "New Ticket", "Edit Ticket", etc.
- Not referenced configuration
 - Contains all configurations that are not referenced in any context or form: e. g. Status of the setup assistant.

The tree can be filtered using the search field above the explorer. The search is based on a full text search without wildcards (*). Entering "tic" finds both **articles** and **tickets**. The metadata of the configuration (e.g. ID, name, type, etc.) is stored for each node in the tree, so that filtering can be carried out according to this (e.g. "FormField" filters all FormField configurations). A search for "modified" finds all keys whose configuration has been changed. Modified configurations are marked with "modified" in brackets.

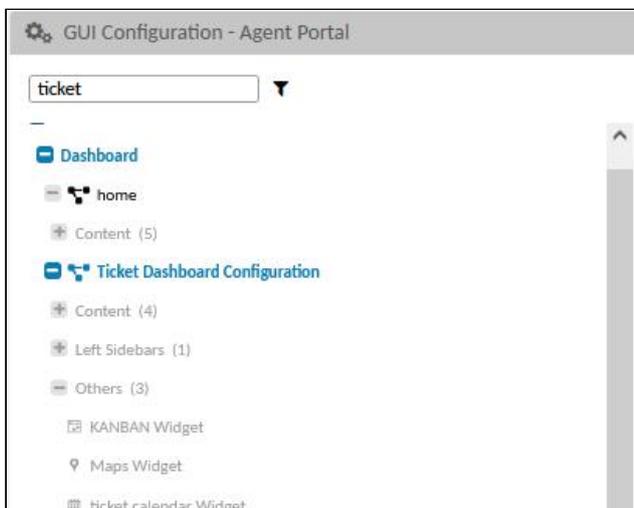


Fig.: The configuration tree

4.4.2 The JSON Editor

After clicking on a configuration branch in Explorer, its value is displayed in the integrated JSON editor and can be edited there. This is done in the same way as described in the configuration examples in the KIX 18 Start admin manual.

The left side of the editor (blue) contains the current configuration of the selected key for viewing and editing. The right side of the editor (green) contains the initial configuration of the key for comparison. If required, it can be displayed by clicking on "Show default value".

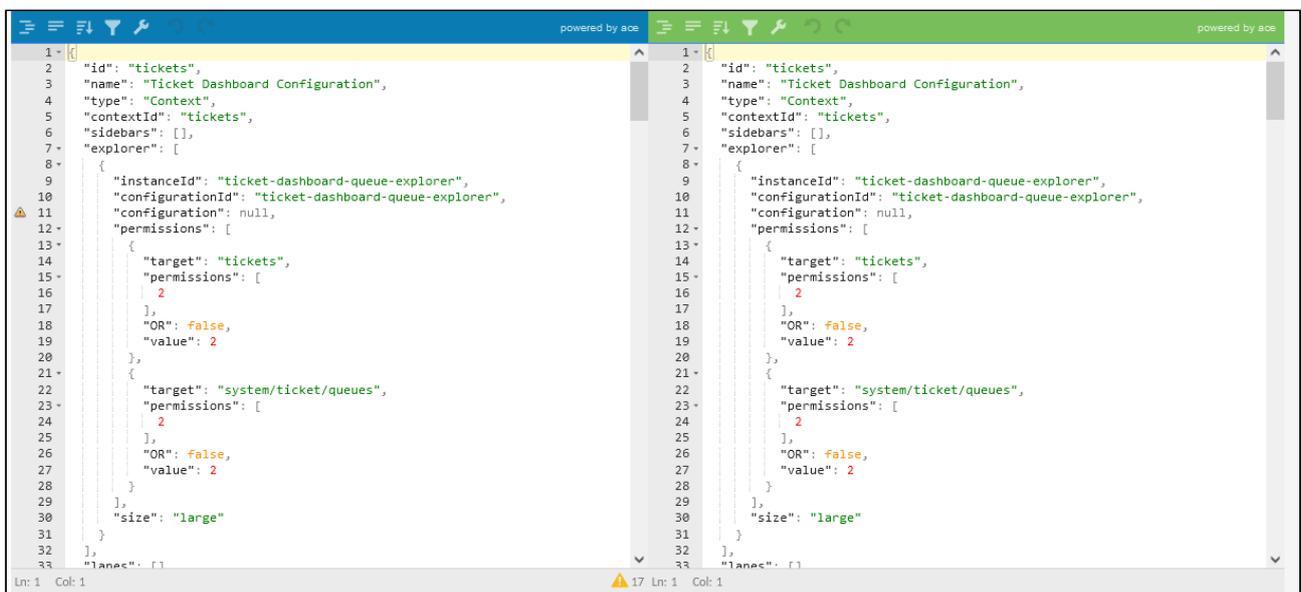


Fig.: The JSON editor with a parallel default value

For better readability, the JSON string is shown with indentations, line feeds and syntax highlighting. The view of the editor can be changed using the buttons in the JSON editor.

	Formatted representation of the JSON string with indentations and line feeds
	Condensed representation of the JSON string without spaces and line breaks
	Sorting the content (ascending/descending)
	Filtering, sorting and transforming the content

	Correction of the JSON string (equalize quotation marks and line breaks, remove comments)
	Undo last action
	Restore/repeat last action

The configurations are validated using a JSON scheme and corresponding warnings and errors are displayed in the editor. Hover over the yellow execution mark for information about the error.

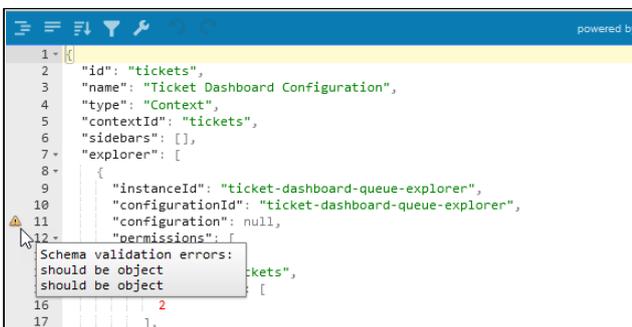


Fig. : Error message in the JSON editor

Note

After changing the configuration key, click on "Reload frontend configurations". Only then is the front-end cache re-created and the configuration tree re-created.

Using the buttons

Show Default Value	Opens another JSON editor (green) with the default value of the key to compare the changes with the initial value. Changes to the value are not possible here.
Hide Default Value	Hides the green JSON editor.
Cancel	Discards the current changes without saving.
Reset to Default	Restores the default value. Then you have to save again.  However, this function does not release you from your duty of care when handling the GUI configuration.



Save	Saves your changes to the configuration. It is also necessary to reload the front-end configuration.
------	---

4.5 Icons

In the *System > Icons* menu you will find the icons used by the system. You can

- if you want to use the system icons when configuring the user interface or sidebar, refer to the overview for the names of the icons
- replace the system icons with your own icons if you want to use company-specific icons
- replace the KIX logo and the KIX icon with your company logo or icon.

How to change an icon:

1. Navigate to *System > Icons*.
2. Click on the icon to be changed in the table. A form dialog opens in which you can change the icon.
3. Click the "Select Image File" button. A dialog for navigating your infrastructure is opened.
4. Navigate to the image file that you want to upload as a new icon.
5. Select the relevant image file for the upload.
6. After a successful upload, the image file is displayed as an icon next to the "Select Image File" button.

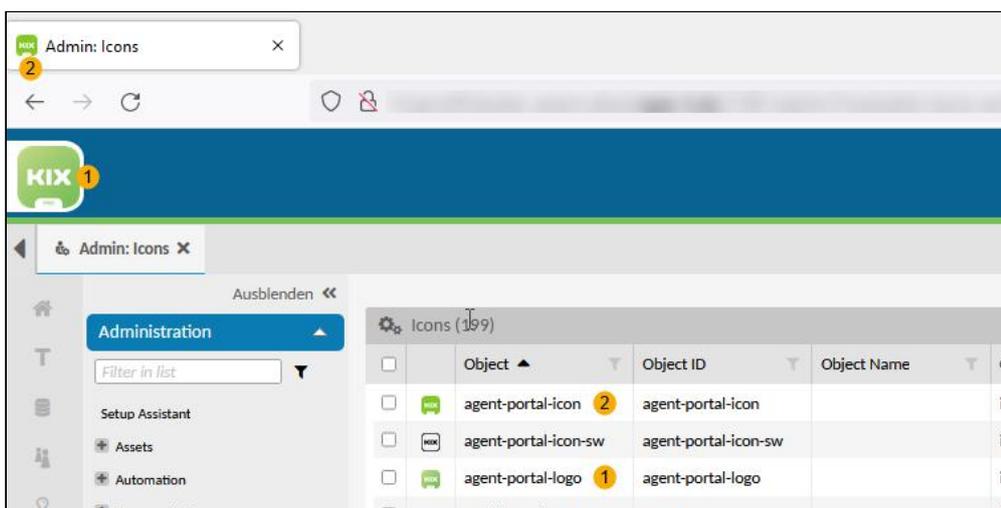
The icon can now be used at the desired location in KIX.

Change KIX logo and KIX icon

The KIX logo is located on the left edge of the screen (**1**) in the header area. The KIX Icon (**2**) is used as a tab icon (favicon).

You can replace both the KIX logo and the KIX icon of the agent portal with your company logo or icon. To do this, filter the table of icons for "agent" and replace the graphic(s) as described above.

- 1 Logo:** Object "agent-portal-logo"
- 2 Icon (favicon):** Object "agent-portal-icon"



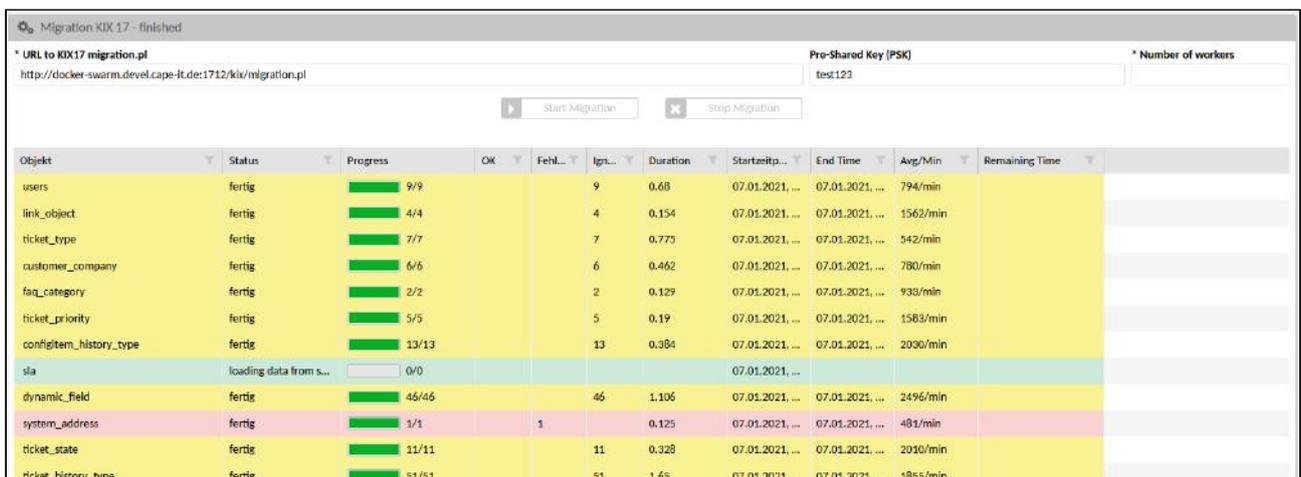
4.6 Migration KIX 17

KIX 18 Pro users can automatically transfer data from KIX 17 to KIX 18. This is done in the *System > Migration KIX 17* menu. 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).

The prerequisites for the migration are:

- a current version of KIX 17 (KIX 17.14.0 and higher)
- the activation of the [pre-shared key mode](#) (see page 66) in the configuration file of KIX17 with specification of the [pre-shared key](#) (see page 66) .

Alternatively, the migration can also be carried out manually via the console in the *System > Console* menu. The console also enables the migration of individual objects such as tickets, organizations, etc. Information on this can be found in the Migration KIX 17> KIX 18 chapter of the start manual. There you will also find further information on migration.



The screenshot shows the 'Migration KIX 17 - finished' interface. It includes a URL to the migration script, a 'Pre-Shared Key (PSK)' field with the value 'test123', and a 'Number of workers' field. Below these are 'Start Migration' and 'Stop Migration' buttons. The main part of the interface is a table with the following columns: Objekt, Status, Progress, OK, Fehl..., Ign..., Duration, Startzeitp..., End Time, Avg/Min, and Remaining Time. The table lists various objects and their migration status.

Objekt	Status	Progress	OK	Fehl...	Ign...	Duration	Startzeitp...	End Time	Avg/Min	Remaining Time
users	fertig	9/9			9	0.68	07.01.2021, ...	07.01.2021, ...	794/min	
link_object	fertig	4/4			4	0.154	07.01.2021, ...	07.01.2021, ...	1562/min	
ticket_type	fertig	7/7			7	0.773	07.01.2021, ...	07.01.2021, ...	342/min	
customer_company	fertig	6/6			6	0.462	07.01.2021, ...	07.01.2021, ...	780/min	
faq_category	fertig	2/2			2	0.129	07.01.2021, ...	07.01.2021, ...	933/min	
ticket_priority	fertig	5/5			5	0.19	07.01.2021, ...	07.01.2021, ...	1583/min	
configitem_history_type	fertig	13/13			13	0.384	07.01.2021, ...	07.01.2021, ...	2030/min	
sla	loading data from s...	0/0					07.01.2021, ...			
dynamic_field	fertig	46/46			46	1.106	07.01.2021, ...	07.01.2021, ...	2496/min	
system_address	fertig	1/1		1		0.125	07.01.2021, ...	07.01.2021, ...	481/min	
ticket_state	fertig	11/11			11	0.328	07.01.2021, ...	07.01.2021, ...	2010/min	
ticket_history_type	fertig	51/51			51	1.65	07.01.2021, ...	07.01.2021, ...	1855/min	

Fig .: Automated migration from KIX 17 to KIX 18

4.6.1 Perform the migration

4.6.1.1 Step 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 unprotected or unauthorized access to the database. The migration only takes place if this previously agreed key is known to both systems. KIX17 rejects requests if no or an incorrect PSK is transmitted.

The PSK is stored in the configuration file (Config.pm) of KIX 17 and given to the migration as a parameter in KIX 18.

Warning

If the PSK mode is activated, the system is open to 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!

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.

```
$Self->{'Migration::Active'} = 1;           //Activate PSK mode
$Self->{'Migration::PSK'} = 'my_PSK_Password'; //Name PSK
```

```
# 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->{EQDN} = 'kalliope';
$Self->{'Migration::Active'} = 1;
$Self->{'Migration::PSK'} = 'my_PSK_Password';

# ----- #
# data inserted by installer #
# ----- #
# SDBIS
```

4.6.1.2 Step 2: Start the migration

1. In KIX 18 Pro, navigate to the *System > Migration KIX 17* menu. A form opens in the content area in which you can enter the required parameters.
2. Enter the following parameters:

Field	Description
URL to KIX17 migration.pl	<p>Enter the URL of the source (KIX 17), e.g. "http://myKIX17domain.de/kix/migration.pl". Entering the domain is sufficient. KIX adds "http://" and "/kix/migration.pl" to the URL.</p> <p>You can find the URL in the address line in the browser of your open KIX17 system: e. g. <code>http://192.168.160.200:4711/kix/index.pl</code> or <code>https://myKIX17domain.de/kix/index.pl</code></p> <p>The specification of the URL corresponds to the specification of the "--source-id" in the console. If you would like to clean up the migration afterwards via the console, please follow the instructions in the start manual (chapter "Installation > Migration KIX 17 > KIX 18").</p>
Pre-Shared Key (PSK)	Enter the value stored in KIX 17, e.g. "start123".
Number of workers	<p>Number of parallel processes to be used for the bulk objects such as tickets etc. Standard: 1</p> <p>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!</p> <p>i Already 4 workers can utilise a host to 100% capacity. Therefore, KIX.Cloud environments are limited to a maximum of 2 workers.</p>

Migration KIX 17

* URL zu KIX17 migration.pl: Pre-Shared Key (PSK): * Anzahl Worker:

3. Start the migration by clicking on "Start Migration". The table shows the status, progress and possible errors of the transfer per object.

4. After successful transfer, the data from KIX 17 are contained in KIX 18.

Übersicht Assets (501) Filtern in Liste							
<input type="checkbox"/>	A#	Name			Klasse	Geändert am	Geändert...
<input type="checkbox"/>	1722000001	Computer 1001	[?]	🟢	Migration-Computer	23.11.2020, 08:17	Admin KIX
<input type="checkbox"/>	1722000002	Computer 1002	🟢	🟢	Migration-Computer	23.11.2020, 08:17	Admin KIX
<input type="checkbox"/>	1722000003	Computer 1003	🟢	🟢	Migration-Computer	23.11.2020, 08:17	Admin KIX

4.6.1.3 Step 3: complete the migration

1. Deactivate pre-shared key 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.

2. Review and edit the migrated data

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

The migrated data is 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".



4.6.2 Notes on migration

Cancel migration:

You can interrupt the migration. To do this, click on "Stop Migration". The background processes are terminated. This can take some time, depending on the progress of the migration. When the migration is restarted, only the tickets that have not been migrated or that have been partially migrated are considered.

Migration failed

If an error occurs during the migration, this is documented in a log file (menu System> Logs). The display only documents that an object was faulty.

In the event of an HTTP error (e.g. wrong URL), the HTTP status is displayed as progress status.

Legend

The lines are color-coded according to their status:

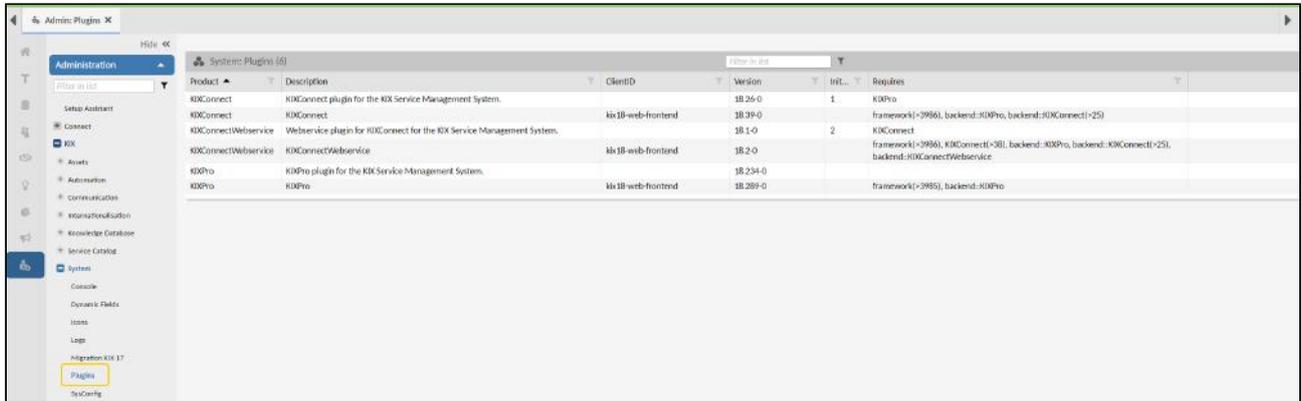
red	Error
yellow	ignored
green	successful
white	pending

Migrated data

For further information on the migrated data, please refer to the KIX 18 Start Admin manual under Installation > Migration KIX 17 > KIX 18.

4.7 Plugins

The menu *KIX > System > Plugins* lists the installed plugins.



Product	Description	ClientID	Version	Inst.	Requires
KIXConnect	KIXConnect plugin for the KIX Service Management System.		18.26-0	1	KIXPro
KIXConnect	KIXConnect	kix18-web-frontend	18.29-0		framework(-3986), backend:KIXPro, backend:KIXConnect(-25)
KIXConnectWebservice	Webservice plugin for KIXConnect for the KIX Service Management System.		18.1-0	2	KIXConnect
KIXConnectWebservice	KIXConnectWebservice	kix18-web-frontend	18.2-0		framework(-3986), KIXConnect(-38), backend:KIXPro, backend:KIXConnect(-25), backend:KIXConnectWebservice
KIXPro	KIXPro plugin for the KIX Service Management System.		18.234-0		
KIXPro	KIXPro	kix18-web-frontend	18.289-0		framework(-3985), backend:KIXPro

Fig.: Overview of the installed plug-ins

4.8 Single Sign On (SSO) with Kerberos

Configuration key	Authentication###00-Kerberos
--------------------------	------------------------------

When calling up the agent or self-service portal, the user can be authenticated by Kerberos. This means that the user does not have to log on to KIX with a user name and password if he is already authenticated by his login to the domain (= Single Sign On).

The prerequisite for this is that Kerberos is actively used in your company and a user is authenticated against a domain (e.g. Active Directory/LDAP).

For the connection to KIX, you need a **keytab file** which you receive from your administrator. In order for KIX to use the keytab file directly in SysConfig, its content must be stored in Base64 format (recommended procedure). To convert the keytab file into Base64 format, you can either use free tools such as base64encode.org² or the Linux console (see below). Alternatively, you can include the keytab file in the backend container and then store its path in SysConfig.

Content on this page:

- [Connection](#) (see page 71)
- [Activate SSO in the frontend](#) (see page 73)
- [The keytab file](#) (see page 73)
- [Notes on Base64 coding](#) (see page 73)
- [Use / client-side requirements](#) (see page 74)
- [Frequently occurring error causes](#) (see page 74)
- [References](#) (see page 74)

4.8.1 Connection

1. Navigate to *System > SysConfig* and open the configuration key *Authentication###00-Kerberos*.
2. Mark the authentication as "active" ("`Enabled`": 1).
3. Store the `keytab` file converted to Base64 format under `Keytab`.
4. Remove or set optional parameters (see section "Configuration" below).
5. Save your changes and click on "Reload frontend configuration".
6. Activate SSO in the frontend
7. Afterwards, Kerberos authentication can be used.

² <http://base64encode.org>



Configuration key: "Authentication###00-Kerberos"

```
[
  {
    "Name": "Kerberos Example",
    "Enabled": 0,
    "Module": "KIXPro::Kernel::System::Auth::Kerberos",
    "Config": {
      "Keytab": "<insert base64 content of keytab file here>",
      "KeytabFile": "(optional)<path to keytab file in filesystem>",
      "Realm": "(optional)<insert your realm here>",
      "krb5.conf": "(optional)<insert base64 content of krb5.conf file here>"
    }
  }
]
```

KIX Configuration

Parameter	Description
Keytab	Content of the keytab file as a Base64 string
Keytabfile	Specification of the file path to the keytab file as it is included in the backend container. (Alternative specification to "Keytab")
Realm	Optionally, enter the Kerberos realm (name of the DNS domain in capital letters), e.g. @EXAMPLE.ORG Specify the realm especially if the user ID contains the domain, e.g. user name: "john.doe@example.org ³ ". Without specifying the realm, the login would fail because Kerberos returns the user name "john.doe" (without domain).
krb.conf	Kerberos configuration file as Base64 string (optional)

³ <http://example.org>

4.8.2 Activate SSO in the frontend

SSO can be activated for the agent or self-service portal. To do this, the respective configurations must be activated in the environment file of your on-premise configuration. A stack restart is then required.

environment

```
SSO_ENABLED=true  
SSO_ENABLED_SSP=true
```

After that, the stack must be restarted.

4.8.3 The keytab file

A keytab file is a cryptographic, binary text file that contains the user accounts including the passwords encrypted with a hash. It enables the automatic login KIX.

You can create the keytab file with the Kerberos utility *ktpass*, for example. Information on this can be found at: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc753771\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc753771(v=ws.11))

In the following example, replace `FQDN`, `DOMAIN` and `PASSWORD` with your corresponding parameters. Note that these are case-sensitive.

Example: Specifications for a keytab file

```
C:\temp> ktpass -princ HTTP/fqdn@DOMAIN -mapuser USER@DOMAIN -crypto RC4-HMAC-NT  
-ptype KRB5_NT_PRINCIPAL -pass PASSWORD -out c:\temp\kix.keytab
```

Important

When creating the keytab file, make sure to specify a correct Service Principal Name (SPN), e.g.:
HTTP/my.webserver.hostname@DOMAIN.IN.CAPITALLETTERS.ORG

4.8.4 Notes on Base64 coding

Under **Linux** you can use the following command line to encode the keytab Base64 (result in file `kix.keytab.b64`). The content can be used directly in parameter "Keytab".



Base64-Coding Keytab File (Linux)

```
SomeLinuxSystem:~# base64 -w 0 ./kix.keytab > ./kix.keytab.b64
```

Under **Windows** you can use the following command lines to encode the keytab Base64 (result in file kix.keytab.b64). For use in parameter "Keytab", the line breaks must be removed.

Base64-Coding Keytab File (Windows)

```
C:\temp> certutil -encode -f kix.keytab kix.keytab.encoded  
C:\temp> type kix.keytab.encoded | find /v "CERTIFICATE" > kix.keytab.b64
```

4.8.5 Use / client-side requirements

For Single Sign On to work via Firefox, the setting "network.negotiate-auth.trusted-uris" must contain the FQDN of the KIX environment or even the entire target domain (e.g. "yourkix.example.com" or "*.example.com").

In the Edge browser, the security policy must be configured accordingly.

4.8.6 Frequently occurring error causes

If SSO via Kerberos does not work, first check whether a Kerberos ticket for the KIX FQDN is available on the authenticating Windows client. To do this, use the command line tool "klist".

In order to prevent problems, continue to observe the following essential requirements when using Kerberos:

- Name resolution of the FQDN must also work backwards.
- FQDN must be an A-host entry in the DNS.
- The system time of the KIX (backend) must be correct.

4.8.7 References

- Kerberos: <https://www.ibm.com/docs/en/was/9.0.5?topic=mechanism-kerberos-krb5-authentication-support-security>
- Service principal name (SPN): <https://www.ibm.com/docs/en/was/9.0.5?topic=server-creating-kerberos-service-principal-name-keytab-file>
- Kerberos Realm: https://web.mit.edu/kerberos/krb5-1.12/doc/admin/realms_config.html
- ktpass: <https://www.ibm.com/docs/en/finenet-p8-platform/5.5.x?topic=system-creating-kerberos-keytab-using-ktpass>



5 Services and SLA

Services are understood in KIX as operating equipment and resources for the provision of other services. They are therefore maintained in the "Service" asset class. The asset class "Service" is initially delivered with KIX Pro. On the ticket, services can be selected in the "Affected Services" field (analogous to "Affected Assets").

Services can be assigned to explicit SLAs so that these assignments can be used automatically on the ticket. The assignment of services to SLAs and the definition of the conditions under which they are available on the ticket is done in the module "Service Contracts" . Please also refer to the chapter "Service Catalog Management" in the KIX 18 User Manual.

You can administratively control the selection of services and SLAs on the ticket. This information can be found under [Service Contracts \(see page 76\)](#) in the KIX 18 Pro Admin Manual.

5.1 Service Contracts

A service contract defines which service and which SLA (Service Level Agreement) are available on the ticket under which conditions. Service contracts are managed in the "Service Contracts" module . Below you will find information on administrative configuration settings of the service contracts.

A detailed description of the module and how to use service contracts can be found in the KIX Pro user manual.

Content on this page:

- [Updating the service tree \(see page 76\)](#)
- [Steering of the "Affected Services" selection \(see page 76\)](#)
- [Affected Assets vs. Affected Services. \(see page 78\)](#)
- [Criticality \(see page 79\)](#)
- [Import and export service contracts \(see page 79\)](#)
- [Use of post-productive services \(see page 80\)](#)

5.1.1 Updating the service tree

The service tree in the explorer of the *Service Contracts* module contains all assets of the asset class "Service" and lists them hierarchically. If a new asset of the class "Service" is created, the service tree is reprocessed in the backend. Alternatively, the service tree can be rebuilt using a console command. To do this, start the command `Console::Command::Maint::Service::UpdateServiceCatalog` under *System > Console*. After that, the service tree is updated.



Fig.: Updating service tree via console

5.1.2 Steering of the "Affected Services" selection

Configuration key:	Agent Portal	service-contract-configuration
---------------------------	--------------	--------------------------------

	Self Service Portal	service-contract-configuration-ssp
--	---------------------	------------------------------------

With the above mentioned SysConfig keys (menu "System > SysConfig") you can control the selection in the dynamic field "Affected Services". The behaviour can be set separately for the Agent Portal and for the Self Service Portal.

The configuration fundamentally affects the Dynamic Field, regardless of whether it is included in the ticket or in the configurations of templates or actions.

Configuration:

Parameters	Description
enabled	<p>Activation/deactivation of the service contracts in the ticket.</p> <ul style="list-style-type: none"> <code>true</code> (Default): <ul style="list-style-type: none"> The service tree is available for selection. The services and SLA available for selection in the ticket are filtered based on the parameters set in the ticket (contact, organisation, type, priority). If one of these parameters is changed in the ticket, the tree is redetermined for the dropdown (alternative: see "Updating the service tree" above). The filter also takes into account the values set in a template as "fixed value" or "in the background". <code>false</code> : <ul style="list-style-type: none"> The field "Affected Services" corresponds to the normal search for an asset (full text or with *). The assets of the class Service are pulled; the dependencies defined in them are taken into account. The display in the selection field is a list and not a tree structure. The configuration of the dynamic field "Affected Services" applies.
singleSelection	<p>Single or multiple selection of services on the ticket. The services available for selection are based on the configuration of the service contracts.</p> <ul style="list-style-type: none"> <code>true</code>: A maximum of 1 "Affected Service" can be selected. <code>false</code> (default): Allows multiple selection of services.

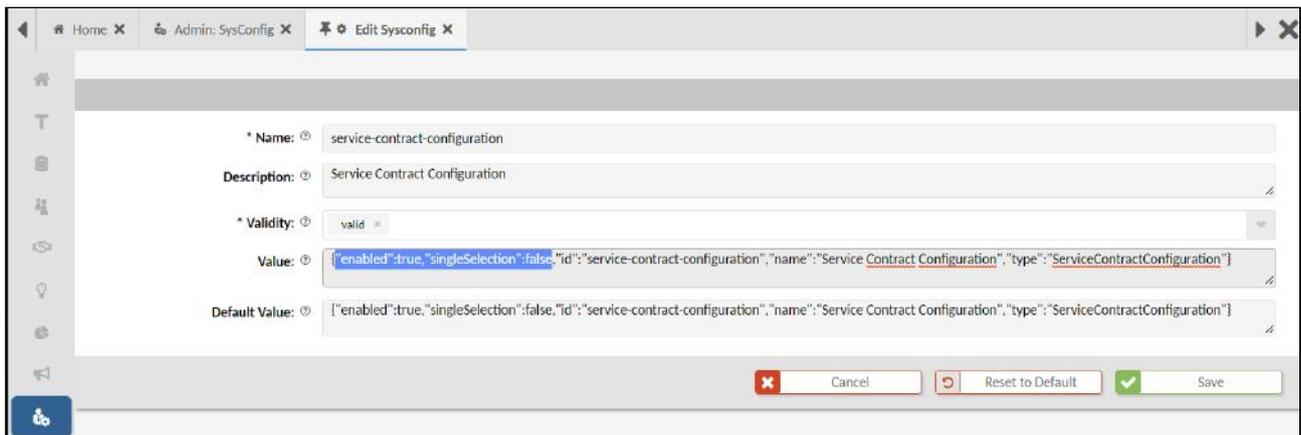


Fig.: The SysConfig-Key "service-contract-configuration"

i Information

KIX Pro allows you to comfortably customise templates in the menu *Ticket > Templates*. If you use service contracts, do not change the field order in the default template, as this could lead to implausible ticket creation. The initial field order results from the logical sequence of dependencies and the workflow:

- The selection of contact, organisation and ticket type influence the services. Therefore, these fields are above the "Affected Service" field.
- The selected priority influences the possible SLA. Therefore, the priority is placed directly above the field "SLA/Service Agreement".

If the field order still has to be changed, this does not affect the selection options of the services and SLAs on the ticket.

5.1.3 Affected Assets vs. Affected Services.

- **AffectedServices:** The assignments defined in the "Service Contracts" module apply.
 - Default services are services for which neither organisation, contact, priority nor type are stored. They are therefore also available for every SSP user.
If services have been restricted via "Service Contracts", they can only be selected according to their configuration (contact, organisation, priority and type) in SSP as well as in the AP.
- **AffectedAssets:** The assignments and visibilities according to the SysConfig key "AssignedConfigItemsMapping" apply (see also: Configuration settings and [Control visibilities in the SSP \(see page 98\)](#)).

5.1.4 Criticality

An asset of the class "Service" must be assigned a criticality. The criticality defines the system relevance of the service. Initially, 4 levels of criticality can be selected:

1. existential
2. business critical
3. support process
4. none/minor.

You can change these criticality levels or add more. This is done in the Admin module in the menu *Assets > General Catalog*. How to add or change entries to the General Catalog can be found in the chapter General Catalog of the Admin Manual of KIX 18 Start.

5.1.5 Import and export service contracts

You can import and export services including their assignments to SLA criteria via CSV file. This enables you to maintain mass data externally and to transfer service contracts from a KIX 18 test environment to a KIX 18 productive environment or from an on-premises installation to the KIX 18 Cloud.

To do this, first export at least 1 existing service contract as a CSV file. Modify it as required and then import it into the target system.

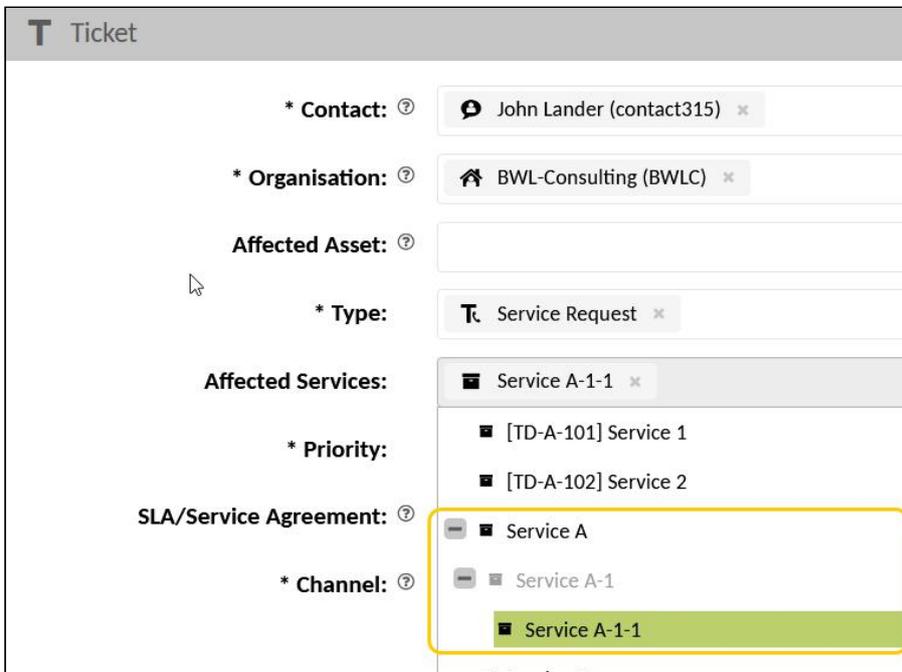
The detailed procedure for this can be found in the KIX 18 Start user manual:

- Execute import/export: General Function Descriptions
- Notes for the import: Service Contracts - Create and Edit

5.1.6 Use of post-productive services

A service is post-productive if it has the usage status "Out of Service" or "Inactive" (see also Configuration Settings). Nevertheless, post-productive services can be used to some extent:

- **In the Asset module:** Post-productive services - like all post-productive assets - are not visible and can only be found via the complex search.
- **In the Service module:** The Services tree contains all services. Post-productive services are marked greyed out, but can be used.
- **In the Ticket:** Post-productive services are greyed out if they have productive/selectable sub-services. However, they cannot be selected. Productive and pre-productive services and sub-services can be selected depending on the conditions defined in the service contract.



The screenshot shows a 'Ticket' form with the following fields and values:

- * Contact:** John Lander (contact315)
- * Organisation:** BWL-Consulting (BWLC)
- Affected Asset:** (empty)
- * Type:** Service Request
- Affected Services:**
 - Service A-1-1
 - [TD-A-101] Service 1
 - [TD-A-102] Service 2
- * Priority:** (empty)
- SLA/Service Agreement:**
 - Service A
- * Channel:**
 - Service A-1
 - Service A-1-1 (highlighted in green)

Fig.: Post-productive and productive Services in the Ticket

5.2 Service Level Agreements (SLAs)

In KIX Pro you can define Service Level Agreements (SLA) and assign them to tickets or assets. SLAs are created and configured in the *Service Catalog > Service Level Agreements (SLA)* menu.

Service level agreements are - to put it simply - agreements about the time in which a service is to be provided.

An SLA is made up of two SLA criteria: response and resolution. Both criteria are based on a default duration in business minutes and a reminder threshold in percent. If the default duration is 120 minutes, the criterion 120 business minutes after the ticket was created must be met in order not to violate the SLA. The calendar selected in KIX defines the business hours.

If an SLA is assigned to a ticket, the objectives defined in the SLA are stored on the ticket. Concrete values for start time, reminder time and target time are calculated. This information is then available on the ticket.

Content of this page:

- [SLA criteria and attributes \(see page 81\)](#)
- [Scheme of an SLA \(see page 84\)](#)
- [Create or Edit an SLA \(see page 85\)](#)
- [SLA to Ticket \(see page 86\)](#)
- [SLA to Assets / Device SLA \(see page 86\)](#)
- [Notifications for SLA \(see page 87\)](#)
- [Use of SLA in KIX placeholders \(see page 87\)](#)
- [Configuration of the fulfillment time \(see page 88\)](#)
- [Calendar configuration \(see page 89\)](#)

5.2.1 SLA criteria and attributes

The following SLA criteria and attributes are saved on tickets. These can be found in the lane "Service Level Agreement (SLA)" in the detailed view of a ticket.

SLA Criterion (Attribute)	Explanation
SLA/Service Agreement	Name of SLA
<i>Reaction (First Response)</i>	<i>Refers to the first reaction of an agent after a ticket has been created and the SLA has been assigned.</i>
Start time	Starting time of the First Response Time. Corresponds to the point in time when the ticket was created.



SLA Criterion (Attribute)	Explanation
Target time	Target time of the First Response Time. There must be a response by this point in time. i.e. an agent must create an article on the ticket that is visible for the customer.
Fulfillment time	Point at which the First Response was actually performed, meaning the agent created an article that is visible for the customer. Initial: Time at which an item visible to the customer was created by the agent (see below).
Reminder	Reminder time for the First Response Time: At this point in time KIX sends a reminder message to all agents who have permission to edit the ticket, informing them that the First Response Time is about to be escalated.
Deviation (Service time)	Absolute time difference between target time and fulfillment time (also includes "overnight" or "weekend"). For "Service time" the term "Business time" is also common.
Deviation	Time difference between the target time and fulfillment time in business minutes (without "night").
Violation	"Yes" or "No"; shows at a glance whether the obligation defined in the SLA has been fulfilled.
<i>Solution</i>	<i>Refers to completing the ticket to which the SLA has been assigned.</i>
Start time	Starting time of the First Response Time. Corresponds to the point in time when the ticket was created.
Target time	Target time of Solution Time: The ticket must be resolved by this point in time, i.e. it must have achieved the state "closed".

SLA Criterion (Attribute)	Explanation
Fullfillment time	Time at which the ticket was actually resolved, i.e. the state "closed" was saved for the ticket.
Reminder	Reminder time for the Solution Time: At this point in time, KIX sends a reminder message to all agents who have permission to edit the ticket, informing them that the Solution Time is about to be escalated for this ticket.
Deviation (Service time)	Absolute time difference between target time and fulfillment time (also includes "overnight" or "weekend"). For "Service time" the term "Business time" is also common.
Deviation	Time difference between the target time and fulfillment time in business minutes (without "night").
Violation	"Yes" or "No"; shows at a glance whether the obligation defined in the SLA has been fulfilled.

ⓘ Information

The SLA criteria and attributes are available in the ticket complex search, in filters for notifications and jobs, in job actions and in placeholders (see below).

5.2.2 Scheme of an SLA

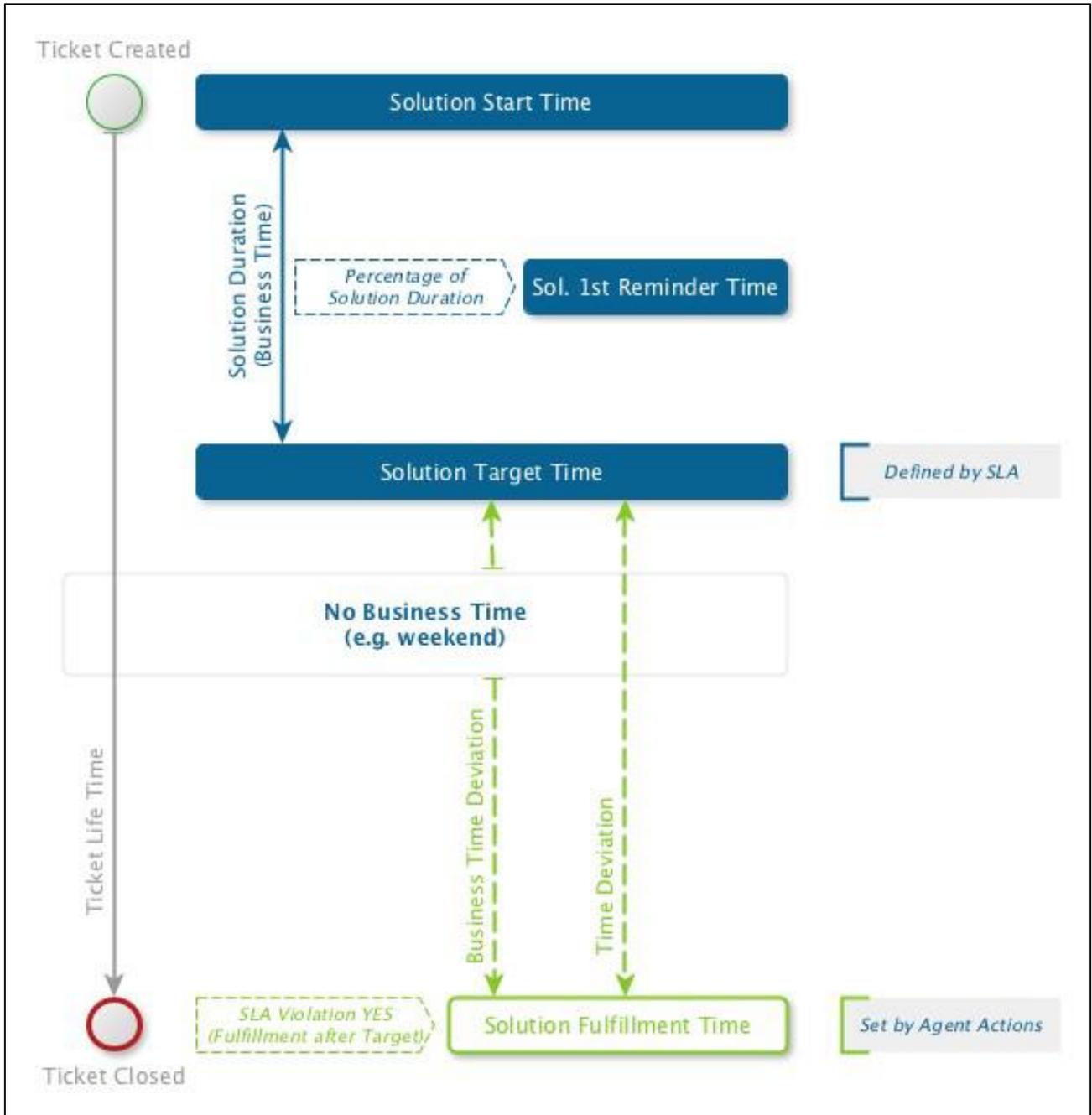


Fig.: Visualization of SLA Attributes for Solution Time Criterion



5.2.3 Create or Edit an SLA

In the delivery state, KIX already contains two SLAs. Their function is explained below in the areas of *SLA on tickets* and *SLA on assets*. To create a new SLA, click on the "New SLA" action in the overview table. Fill out the form (see table below) and set the validity to "valid". Save the SLA with "Save". Your new SLA is now displayed in the overview table.

To edit or to see details, click on the desired SLA in the overview table. The "Edit SLA" dialog opens. Change the information and apply the changes with "Save". Your changes take effect immediately. The agents may need to update their browser page so that the changes are also visible to them.

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

Field	Description
Name	Give the SLA a meaningful and unambiguous name. (e.g. : "Company Mayer - Gold - Major Incident", "Company Mayer - Gold - Service Request")
Calendar	Select the calendar that serves as the basis for calculating the SLA times. You can configure the calendar via SysConfig (see below)
First Response	Enter the response time in business minutes on which the SLA is based. Entering "0" does <u>not define</u> a response time, e.g. if you only work with solution times (Solution Duration) and not with reaction times.
Notify at	Specify when you want to be notified in tickets with this SLA. Example "50%": If half of the time entered above has passed on a ticket with this SLA, you will receive a reminder notification.
Suspending Ticket States	Specify the statuses that cause the response time to pause on the ticket (optional).
Solution Duration	Enter the solution duration on which the SLA is based (in business minutes). Entering "0" does <u>not define</u> a solution time, e.g. if you only work with reaction times (First Response) and not with solution times.

Field	Description
Notify at	Specify when you want to be notified in tickets with this SLA. Example "50%": If half of the time entered above has passed on a ticket with this SLA, you will receive a reminder notification.
Suspending Ticket States	Specify the statuses that cause the response time to pause on the ticket (optional).
Validity	<ul style="list-style-type: none"> • valid: The SLA can be used. • invalid/temporarily invalid: The SLA cannot be assigned to a ticket. The SLA is retained for existing tickets and is still valid. If the ticket information is opened for processing, a red frame indicates the lack of the SLA. A new, valid SLA can be selected.

5.2.4 SLA to Ticket

When creating or editing a ticket, all valid SLAs can be selected via a dropdown. The SLA criteria are set on the ticket as described above. If a ticket is to be created without an SLA, the standard SLA "No escalation" can be set.

If the SLA "SLA by affected service and priority" is selected on the ticket, the target criteria of all permissible SLAs based on services, organisation, contact, type and priority are determined and the shortest target time in each case is stored on the ticket.

The reaction or solution time on the ticket can be paused, e.g. when waiting for feedback from the customer. The pausing is linked to the status of a ticket. You can determine which states trigger the pause when you create the SLA ("Suspending Ticket States" field). If necessary, create additional ticket states, e.g. "Waiting for reply" (see chapter States). If the ticket leaves this status, the time for the corresponding criterion continues to run. The job "SLA Time Suspension" delivered initially with KIX checks the status and controls the display in the lane "Service Level Agreement (SLA)".

5.2.5 SLA to Assets / Device SLA

To set SLA on assets, the attribute type "SLAReference" is available for the asset class definition. If this attribute is included in the class definition of an asset, exactly one SLA can be set for each asset when creating and / or editing. In order for the SLA of one or more assets to be stored on a ticket, the initially delivered SLA "SLA according to affected asset/s" must be selected in the ticket.

When calculating the specific SLA targets, all assets assigned to the ticket are taken into account ("Affected assets" field). The "strictest" criteria are selected from all the SLA criteria specified by the assets and used on the ticket to calculate the target values. It is therefore possible that the response time of one asset and

the resolution time of another asset take effect on the ticket. This ensures that all SLA requirements can be met.

5.2.6 Notifications for SLA

Corresponding events are available for configuring notifications related to SLA. For example, to send notifications when an SLA criterion is changed or a ticket threatens to escalate. The following four notifications are initially delivered:

- *Notify First Response Warning To Team*
 - Is sent when the reminder for the response time of the SLA has been reached on a ticket.
 - Recipient: all agents with update authorization for the ticket.
- *Notify First Response Escalation to Team*
 - Will be sent when the response time of the SLA has been reached on a ticket.
 - Recipient: all agents with update authorization for the ticket.
- *Notify Solution Warning To Team*
 - Is sent when the reminder for the resolution time of the SLA has been reached on a ticket.
 - Recipient: all agents with update authorization for the ticket.
- *Notify Solution Escalation To Team*
 - Will be sent when the solution time of the SLA has been reached on a ticket.
 - Recipient: all agents with update authorization for the ticket.

5.2.7 Use of SLA in KIX placeholders

The SLA criteria can be used for placeholders according to the pattern "`<KIX_TICKET_SLACriteria_Criteria_Attribute>`".

Hint

When creating a new ticket, only the placeholders "KIX_TICKET_SLALID" and "KIX_TICKET_SLA" can be used.

Background: The SLA attributes are set on tickets via an event. The SLA attributes are only available once the ticket has been saved. No placeholder replacement can be made beforehand. This also applies if the SLA is changed when processing a ticket.



5.2.8 Configuration of the fulfillment time

KIX PRO initially has two jobs with SLA criteria. With these you can configure what counts as "fulfillment". You can adapt the configuration of the jobs according to your business processes (see chapter "Creating or editing a job").

Job	Configuration
SLA First Response Time Fulfillment	<p>This job sets a time stamp in a ticket when an article is created that is visible in the Self Service Portal. It defines the time of the first response for SLA fulfillment. You can reconfigure the job as required.</p> <p>The action "Set fulfillment time" should not be removed, otherwise the time stamp will not be set.</p>
SLA Solution Time Fulfillment	<p>This job sets a timestamp in a ticket when a ticket has the status "removed", "closed" or "summarized". It thus defines the point in time of the solution for SLA fulfillment. You can reconfigure the job as required.</p> <p>The action "Set fulfillment time" should not be removed, otherwise the time stamp will not be set.</p>
SLA Time Suspension	<p>This job initiates the suspend/resuming of the response and solution time on the ticket based on the status set in the ticket.</p> <p>The statuses that lead to the suspend of response or resolution time can be specified when creating the SLA. The macro action "SLA Criterion Suspend/Resume" used in the job compares these statuses with the status set on the ticket and initiates the suspend/resume of the reaction or solution time.</p> <p>If the status of the paused ticket is changed, the ticket leaves the pause; the time for the corresponding criterion continues to run. The time is recalculated. The duration of the pause is added to the original time.</p> <p>The configuration of the job should not be changed.</p>



5.2.9 Calendar configuration

KIX is initially delivered with 9 calendars (designation: Calendar1-9). The calendars can be configured via SysConfig keys (menu *System* > *SysConfig*). Information on configuration can be found in the Admin Manual of KIX 18 Start (see Calendar configuration).

Note: If no calendar is stored at the SLA, the default values of the system are used.



5.3 Set response time when creating the ticket

5.3.1 Scenario:

An agent opens a new ticket during an incoming phone call. In the course of the conversation, some things could be clarified, so that the response time has already been satisfied. However, this is not yet set because the current mechanism for setting the response time requires that the item is created by Sender Type "Agent". If a ticket is created in the "Note" channel, however, "external" will be recorded as the sender type.

So that the agent can now already set the response time in the ticket he is creating and does not have to create a new "phone ticket", a Dynamic field of type "Selection" can be integrated into the ticket template. If the option "yes" is selected, a job is triggered that sets the response time on the ticket.

Below, you will find a description of how to create the selection field and integrate it into the ticket template as well as how to configure the triggering job.

5.3.2 Procedure:

1. Create dynamic field
The dynamic field serves as a "switch" to trigger the job (see step 4).
2. Create translation (optional)
Used to automatically translate the dynamic field in the template.
3. Extend standard ticket template
Integration of the dynamic field in the ticket template.
4. Create job
Used to automatically set the response time on the new ticket.

5.3.3 The configurations in detail

5.3.3.1 1. Create dynamic field

Create a new dynamic field of the type "Selection" in the Admin module in the menu "System > Dynamic fields". This serves as a "switch" for setting the response time on the ticket. If the option "yes" is selected in the selection field, the job configured under 4. will be triggered and the response time will be set automatically on the ticket. If the option "no" is selected, the job will not be triggered and no reaction time will be set.

The configuration of the dynamic field in detail. You can also name the name and label used in the example differently.

- Name: FirstResponseByTicketCreation
- Label: Response by Ticket Creation

- Field Type: Selection
- Object Type: Ticket
- Configuration:
 - Count Min: 1
 - Count Max: 1
 - Count Default: 1 (optional)
 - Default value: 0
 - Translatable values: 
 - Possible values (key/value):
 - 0 / no
 - 1 / yes

Further information about dynamic fields:

- Creating, Editing and Incorporating Dynamic Fields
- Object and field types of dynamic fields

5.3.3.2 2. Create translation

In order to translate the label of the dynamic field in the template, a new pattern must be created for it. This step can be omitted if a translation is not required or not desired.

To create the translation, navigate to the menu "*Internationalisation > Translations*" in the Admin module.

Create a new pattern with the following parameters:

- Pattern: Response by Ticket Creation
- English: Response by Ticket Creation
- German: Reaktion durch Ticketerstellung

Further information about translations: Translations

5.3.3.3 3. Extend template

In order for the agents to be able to set the response time when creating the ticket, the dynamic field created in step 1 must be integrated into the template. In the example, the standard ticket template is extended for this purpose. However, you can also integrate the dynamic field into another template.

1. To do this, navigate to the "*Workflow > Templates*" menu in the Admin module and open the "Default - New Ticket Template" template for editing.
2. Go to step 2 "Input fields".
3. At the bottom of the last selection field, select the Dynamic Field created in step 1 and select the option "Set in form".
4. Optionally, you can drag and drop the Dynamic Field to the desired location in the form.
5. Finally, save the template. Afterwards, the Dynamic Field is integrated into the template and can be used for every ticket creation.



Further information on ticket templates: [Templates](#) (see page 191)

5.3.3.4 4. Create job

Setting the response time on the ticket is done automatically by a job. This will be triggered if the agent has selected the option "yes" in the dynamic field "Response by Ticket Creation" during ticket creation. The time of ticket creation is set as the fulfillment time.

To do this, create a new job in the Admin module in the "*Automation > Jobs*" menu with the configuration listed below. You can use the initial job "SLA First Response Time Fulfillment" as a template:

- Job Information
 - Job type: Ticket
 - Name: SLA First Response Time Fulfillment - by DF FirstResponseByTicketCreation
 - Comment: (optional)
 - Validity: valid
 - Execution plan
 - Events: TicketCreate
 - Filter
 - FirstResponseByTicketCreation - contained in - yes
 - Actions
 - Set fulfillment time
 - SLA criterion: FirstResponse
 - Force if not null: 0

Finally, save the job. After that, the response time will be set on the ticket, if requested by the agent.



6 Self-service Portal

The Self Service Portal (SSP) is the customer portal of KIX and is delivered with KIX Pro - both for on-premises installations and for KIX Cloud.

Your customers can use the Self Service Portal to create and edit tickets and also view FAQ articles and assets (operating resources).

The Self Service Portal is linked to KIX, so that direct collaboration between helpdesk and customer is possible.

The Self Service Portal is administered and configured in the Admin module of the agent portal (menu: *KIX > System > GUI configuration > Self Service Portal*).

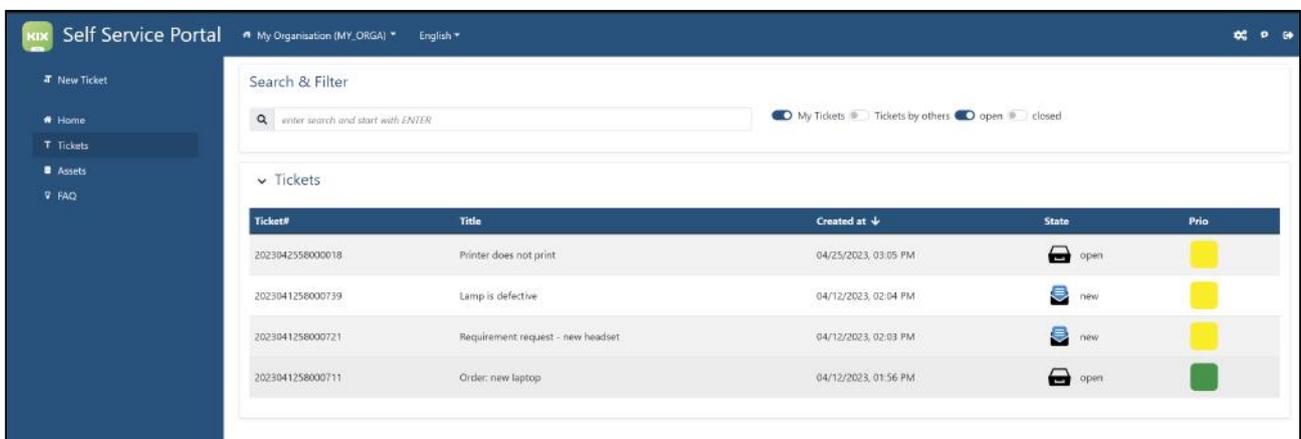


Fig.: The ticket overview in the Self Service Portal

i Self Service Portal 1 was completely replaced by Self Service Portal 2 on 3rd November 2023. The following description deals exclusively with Self Service Portal 2. On request, we can provide you with the description of Self Service Portal 1 as a PDF (version v30) until further notice. To do so, please contact: info@kixdesk.com⁴.

⁴ <mailto:info@kixdesk.com>



6.1 Accessibility of the Self Service Portal

KIX Cloud:

KIX.Cloud users can access the SSP by adding the parameter "-ssp" to the URL of the agent portal, e.g:

Portal	Reachability	Example
Agentportal	URL of the KIX.Cloud environment	https://your.agentportal.kix.cloud
SSP	URL of the KIX.Cloud environment with additional specification of "-ssp".	https://your.agentportal-ssp.kix.cloud

On Premises Installations:

For on-premises installations, the SSP can be accessed by calling the KIX URL with the port number appended:

Portal	Reachability	Example
Agentportal	URL of the KIX installation with additional specification of the port number "20001"	https://your.kix.docker-host.org:20001
SSP	URL of the KIX installation with additional specification of the port number "20002"	https://your.kix.docker-host.org:20002



If necessary, you can change the default ports in the `environment` file. Specify the port numbers under which you connect to KIX.

```
Specify port numbers for the Self Service Portal
1  ...
2  # -----
3  # frontend service configuration
4  # -----
5
6  # the port on the docker host system where the frontend service is
7  # listening
8  FRONTEND_PORT=20001
9  FRONTEND_PORT_SSL=20444
10 ...
```

⚠ Please note
The script alias with added "/ssp" (e.g.: <https://your.kix-docker-host.org:20001/ssp>) is no longer used since 03.11.2023!

Further Information in the [GitHub](#)⁵:

- Linux: <https://github.com/kix-service-software/kix-on-premise/blob/master/deploy/linux/README.md>
- Windows: <https://github.com/kix-service-software/kix-on-premise/blob/master/deploy/windows/README.md>

⁵ <https://github.com/cape-it>



6.2 User settings for the SSP

6.2.1 Set up user login

A prerequisite for access to the Self Service Portal is that the customer contact has been created in the agent portal as a user with the appropriate access authorisation. He needs at least the role "Customer".

To do this, open an existing contact in the *Organisations* module or create a new contact. For customer contacts who already have access to the agent portal (users), the setup can also be done in the module *Admin > KIX > User Management > Users*.

Under "User Information" (additional), select the "Self Service Portal" and assign at least the role "Customer". The following form fields are relevant for the Self Service Portal:

Form field	Explanation
access	Select "Self Service Portal" in the dropdown and click "Apply". With your selection you determine which portals the contact has access to.
user name	Assign a unique user name for the contact with which they can log in to the Self Service Portal.
password	Assign an initial password for the contact with which they can log in to the Self Service Portal. The contact can change this himself in the Self Service Portal.
roles	Select at least the role "Customer" so that the customer has access to the Self Service Portal.
language	Set the language in which the Self Service Portal is displayed for the client.

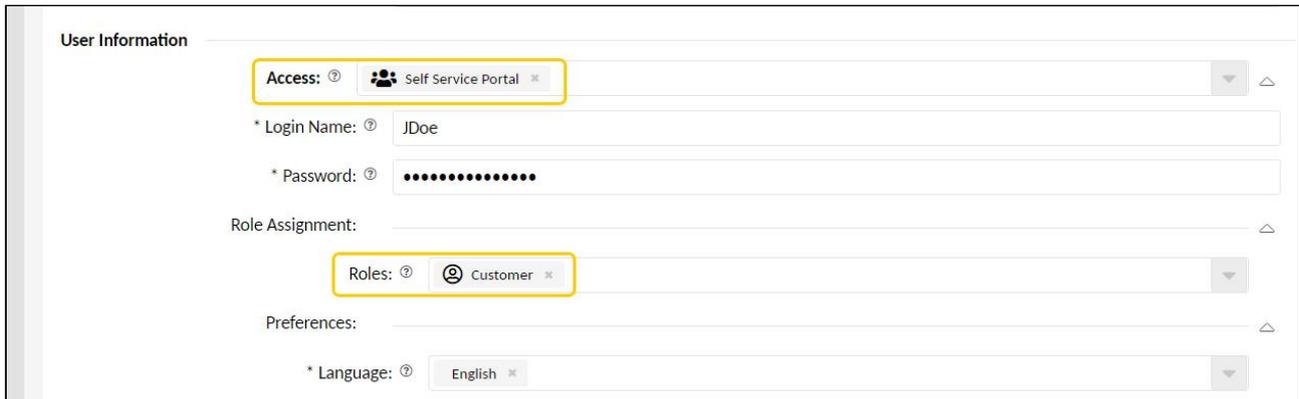


Fig.: Setting up access for the SSP

i Information

If a customer contact has access to the Self Service Portal, he is displayed both as a contact (*Organisations* module) and as a user (*Admin > User Management > Users* module). It can be accessed and edited via both views.

6.2.2 Roles and permissions

For a customer user to have access to the Self Service Portal, he or she needs at least the role "Customer". Other roles can be added to this role in order to extend the permissions of the customer user, e.g. "Customer Manager", "Customer Reader" or others (see also Overview of preconfigured roles in KIX).

Each customer user only sees the objects (tickets, assets, FAQ) to which he has at least read permission.

The display of the information and dynamic fields provided in the Self Service Portal as well as the selection options in select fields depends on the respective user rights.



6.3 Control visibilities in the SSP

You can define which information a customer contact can see in the Self Service Portal:

- all relevant tickets completely
- only selected articles of a ticket
- FAQ items
- complete assets
- selected asset attributes
- selected dynamic fields
- news

Content on this page:

- [Visibility of tickets and articles \(see page 101\)](#)
- [Visibility of follow-ups \(see page 102\)](#)
- [Asset visibility \(see page 104\)](#)
- [Visibility of FAQ articles \(see page 105\)](#)
- [Visibility of Dynamic Fields \(see page 106\)](#)
- [Visibility of news \(see page 107\)](#)

6.3.1 Basic settings for visibilities

Configuration key	Description	Example
<p>AssignedObjectsMapping</p>	<p>Controls the assignment of objects to contacts and/or organisations as well as the visibility of objects in the SSP.</p> <p>Defines which dependent object (e.g. ticket) is assigned to a relevant object (e.g. contact) by which attributes (e.g. tickets from contact).</p> <p>In the example opposite, contacts only see tickets from their own organisation and only articles and FAQ articles marked with "Show in customer portal".</p> <p>Structure:</p> <pre data-bbox="467 1055 1010 1592"> { "RelevantObject": { "DependentObject": { "DependentObjectAttribut": { "SearchAttribut OR SearchStatic": ["RelevantObjectAttribut OR StaticValue"] } } } } </pre> <p>All attributes are combined with OR.</p>	<pre data-bbox="1048 517 1425 1621"> { "Contact": { "Ticket": { "OrganisationID": { "SearchAttributes": ["RelevantOrganisationID"] }, "TicketArticle": { "CustomerVisible": { "SearchStatic": [1] }, "FAQArticle": { "CustomerVisible": { "SearchStatic": [1] } } } } } } </pre>



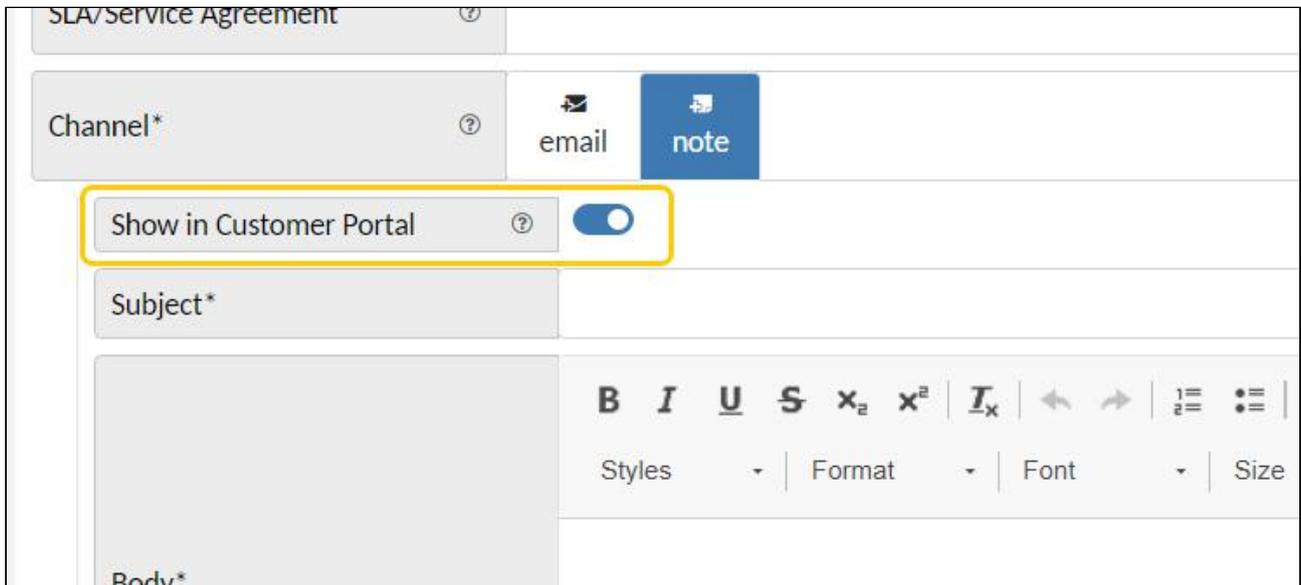
Configuration key	Description	Example
	<p>Note: If emails are retrieved from previously unknown contacts, KIX can create a new contact and assign the organisation based on the mail domain (see also Automatic assignment of the organisation to a contact). The assignment method can be defined in the SysConfig key <i>Contact::EventModulePost###800-AutoAssignOrganisation</i>. If "DefaultOrganisation" is activated as the assignment method, viewing the "Tickets by others" should be prevented for data protection reasons:</p> <pre data-bbox="467 918 1010 1261">{ "Contact": { "Ticket": { "ContactID": { "SearchAttributes": ["ID"] } } }, [...] }</pre>	

Configuration key	Description	Example
<p>AssignedConfigItems Mapping</p>	<p>Controls the assignment of assets to contacts and/or organisations and the visibility of assets in the SSP. Assignments can be made based on usage statuses.</p> <p>In combination with specific usage statuses, e.g. "Baseline Configuration", the display of baseline assets or "shopping item" versions of assets in the Self Service Portal becomes possible.</p> <p>By specifying search criteria, you can limit visibility to post- or pre-production statuses. For example, you can define a static search criterion based on deployment status so that SSP users do not have to distinguish between current data and legacy data themselves (for shopping baskets or baseline configurations).</p> <p>The following attributes are possible:</p> <ul style="list-style-type: none"> • DeploymentState - Expects a list of the status names. • IncidentState - Expects a list of status names. • Name - Expects a string; in the case of a list, only the first value is considered. • Number - Expects a string; in the case of a list, only the first value is considered. <p>For all 4 attributes, the following is possible as a search criterion:</p> <ul style="list-style-type: none"> • SearchStatic - Static search criterion • SearchAttributes - Searches for the specified attributes (e.g. contact attributes) 	<pre data-bbox="1046 398 1423 1603"> { "Contact": { "Computer": { "SectionOwner::OwnerContact": { "SearchAttributes": ["ID"] }, "DeploymentState": { "SearchStatic": ["Production", "Planned"] }, "SectionOwner::OwnerOrganisation": { "SearchAttributes": ["RelevantOrganisationID"] } }, ... </pre>

6.3.2 Visibility of tickets and articles

Basically, all tickets of the logged-in customer contact are displayed in the Self Service Portal. All articles that the contact creates himself are automatically always displayed in the Self Service Portal. For tickets and

articles created in the KIX Agent Portal, the agent can decide whether they are visible in the Customer Portal. This is controlled by activating/deactivating the option "Show in customer portal".



The screenshot shows a form for creating or editing a ticket. The 'Channel*' field is set to 'email'. Below it, the 'Show in Customer Portal' toggle switch is turned on (blue). The 'Subject*' field is empty. Below the subject field is a rich text editor with a toolbar containing bold (B), italic (I), underline (U), strikethrough (ABC), subscript (x₂), and superscript (x²) buttons, as well as text color, background color, bulleted list, and numbered list options. The 'Body*' field is also empty.

Fig.: The ticket is displayed in the Self Service Portal

Hints

- If an item is created in the "Email" channel in which the customer contact is the recipient, the tick "Show in customer portal" cannot be removed.
- If a ticket contains a dynamic field of the type "Checklist", it can be displayed in the Self Service Portal, but not edited.

6.3.3 Visibility of follow-ups

Follow-ups are incoming replies to emails or incoming follow-up messages to existing tickets.

Initially, incoming emails are only visible in the SSP if:

- the ticket contact is included in the recipient list (To, Cc, Bcc, etc.).
- the ticket contact is the sender (not configurable) .
- an incoming message is a referenced response to an article visible to the customer user
- the configuration key *PostMaster::FollowUp::CheckFromOrganisation* is activated and the sender belongs to the ticket organisation
- an X-KIX-FollowUp header is set to "CustomerVisible".

You can deactivate/activate this behaviour as required by setting the following configuration keys to "valid" or "(temporarily) invalid". They are initially active.



Configuration options

Configuration key	Description
Ticket::EventModulePost###900- ArticleCustomerVisibleByRecipient	<p>Controls the visibility of the article in the Self Service Portal based on the email recipient.</p> <p>Checks whether at least one of the email recipients (To, Cc or Bcc) is assigned to the ticket contact when creating an article in the "Email" channel (e.g. after receiving an email or sending an email from KIX).</p> <ul style="list-style-type: none">• If yes: The system automatically sets the visibility of the article in the SSP. This is regardless of whether the "Show in customer portal" option has been set by the agent or not.• If no: The option selected by the agent for "Show in customer portal" is not changed. <p>When replying to an article ("Reply" article action), the visibility in the SSP is initialised with the value of the source article.</p>
Ticket::EventModulePost###910- ArticleCustomerVisibleByReferences	<p>Article event that controls the visibility in the SSP based on the email reference.</p> <p>Is executed when an article is created in the "Email" channel. Checks whether at least one email reference (References and InReplyTo) is the message ID of an article visible to the customer in the ticket.</p> <ul style="list-style-type: none">• If Yes: The visibility of the article in the SSP is automatically activated (regardless of the agent's setting).• If No: The visibility selected by the agent is not changed.

Configuration key	Description
PostMaster::FollowUp:: CheckFromOrganisation	<p>Formerly: <i>TicketStateWorkflow::PostmasterFollowUpCheckCustomerIDFrom</i></p> <p>Checks whether the sender of a follow-up message/reply (follow-up) is contained in the contact database and is assigned to the same organisation as the ticket. If this is the case, the email is displayed in the Self Service Portal.</p> <p>The system only checks for a matching message ID on the follow-up ticket. The visibility check based on the reference is performed by the <i>Ticket::EventModulePost###910-ArticleCustomerVisibleByReferences</i> event (see above).</p>

6.3.4 Asset visibility

Basically, the assets to which a customer contact is assigned as a contact are displayed.

To control the visibility of asset attributes in the Self Service Portal, the property "CustomerVisible" is available for class-specific asset attributes in the class definition.

By default, the cross-class attributes are displayed in the Self Service Portal. For the class-specific attributes, "Customer Visible = 0", i.e. they are not displayed. Set the attribute "CustomerVisible = 1" for the attributes (form fields) you want to display in the Self Service Portal.

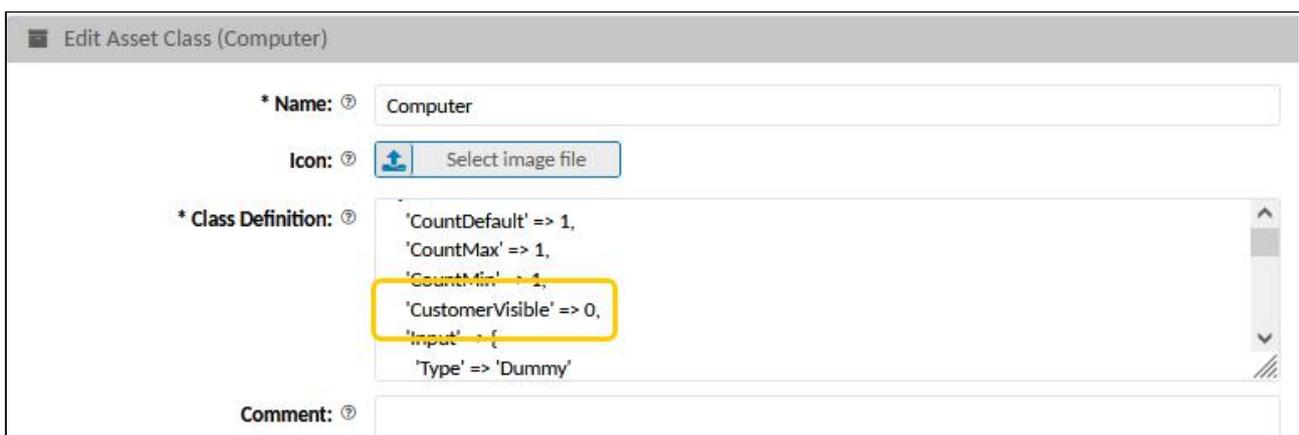
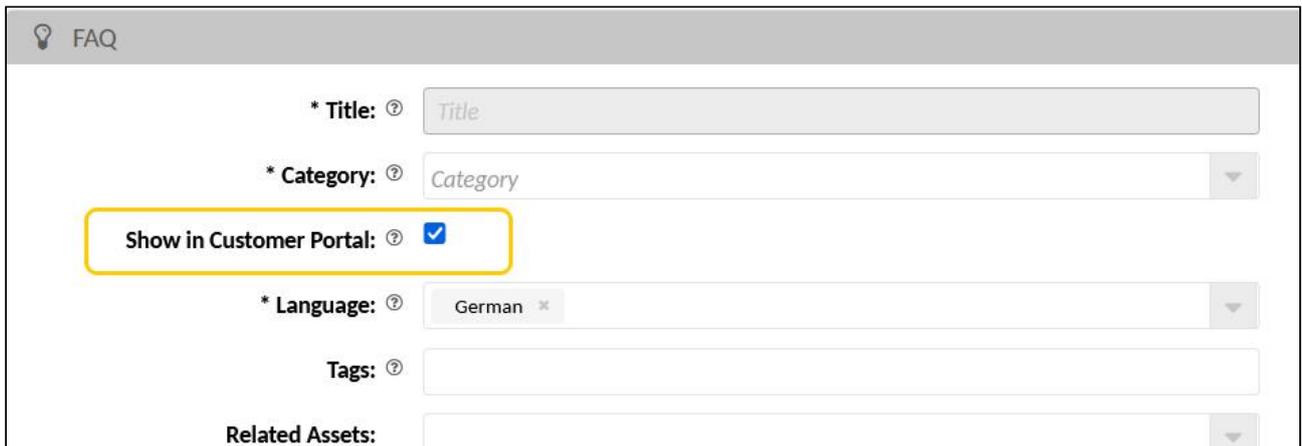


Fig.: The attribute "Customer Visible" controls the visibility of assets in the SSP

6.3.5 Visibility of FAQ articles

By activating/deactivating the option "Show in customer portal", an agent can decide when creating or editing a FAQ article whether the FAQ article is displayed in the Self Service Portal. If the check mark is set, the FAQ article is visible to all users of the Self Service Portal.



The screenshot shows the configuration form for a FAQ article. The form is titled "FAQ" and includes the following fields:

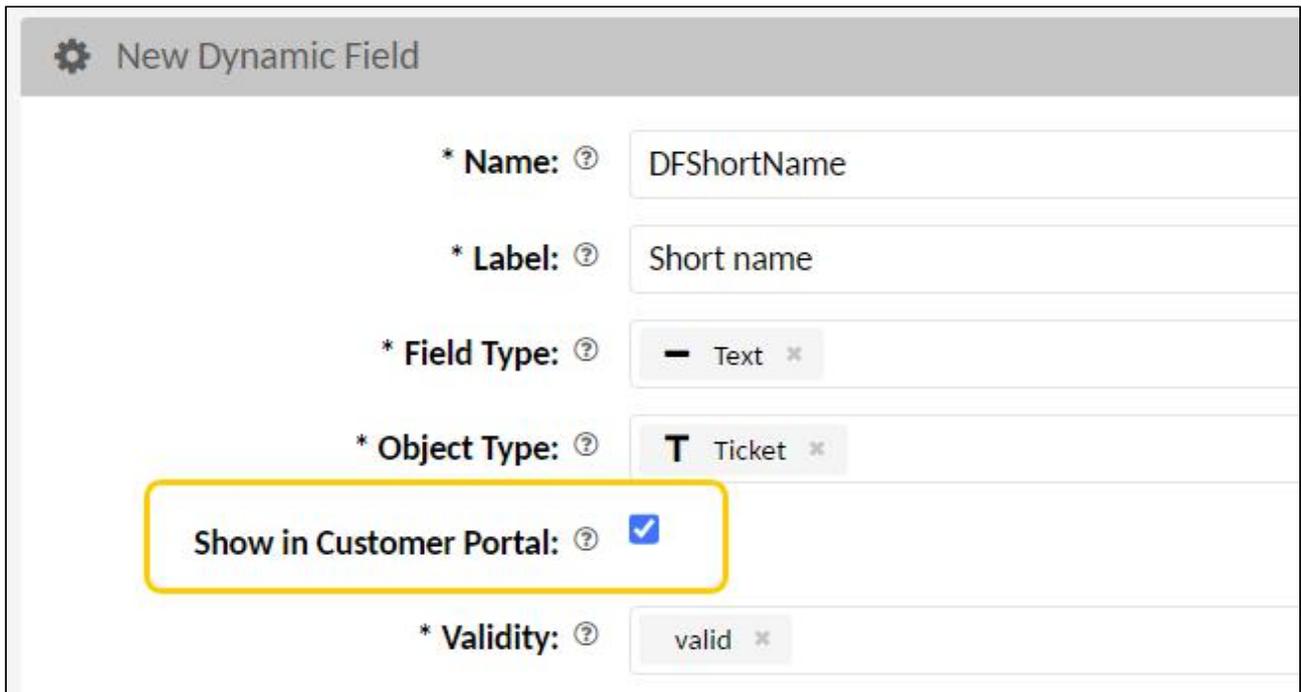
- * Title:** A text input field with the placeholder "Title".
- * Category:** A dropdown menu with the placeholder "Category".
- Show in Customer Portal:** A checkbox that is checked, highlighted with a yellow box.
- * Language:** A dropdown menu with "German" selected and a close button (x).
- Tags:** A text input field.
- Related Assets:** A dropdown menu.

Fig.: Setting the visibility of FAQ articles in the Self Service Portal

6.3.6 Visibility of Dynamic Fields

The Self Service Portal supports the provision of Dynamic Fields. All field types that are also supported by the Agent Portal are supported.

In order for a dynamic field to be displayed in the Self Service Portal, the option "Show in Customer Portal" must be activated in its configuration.



The screenshot shows the 'New Dynamic Field' configuration interface. It contains the following fields and values:

- Name:** DFShortName
- Label:** Short name
- Field Type:** Text
- Object Type:** Ticket
- Show in Customer Portal:** (This field is highlighted with a yellow border in the original image)
- Validity:** valid

Fig.: Making a dynamic field available in the SSP

By activating or deactivating this option, you determine whether the dynamic fields provided in the agent portal are also visible in the Self Service Portal (e.g. in ticket templates or ticket actions).

Referencing dynamic fields (e.g. ticket reference type) in ticket templates or actions only contain the objects for selection that the logged-in SSP user has rights to.

Dynamic fields provided in the Self Service Portal are displayed above the item overview (collapsible area "More information"). Dynamic fields without a value are not displayed.

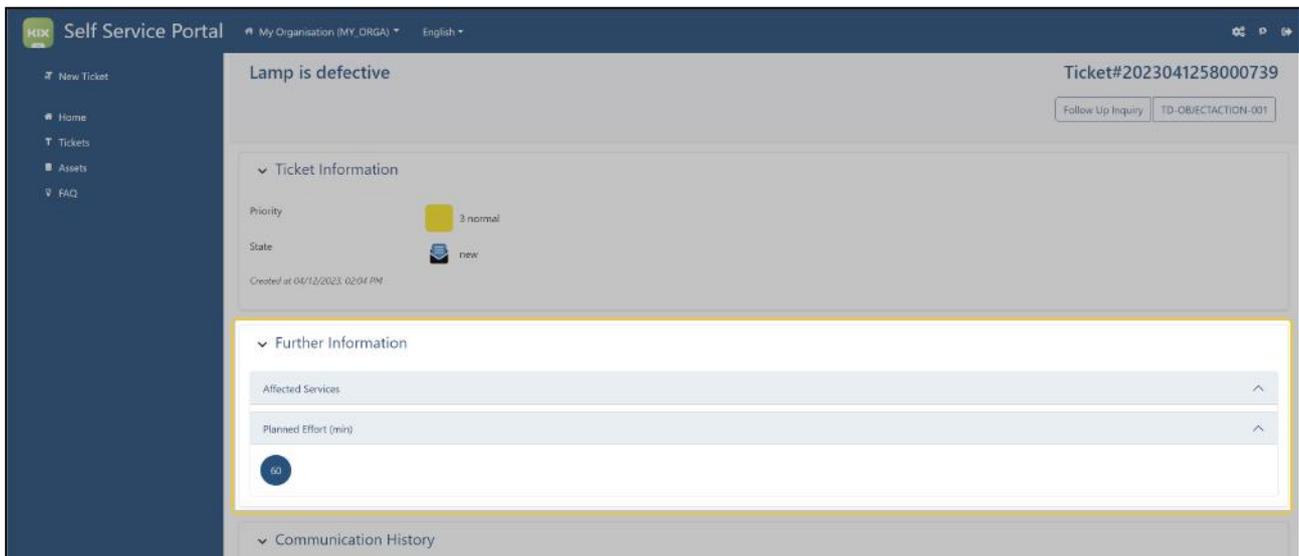


Fig.: Dynamic fields used in the ticket are displayed in a separate area.

6.3.7 Visibility of news

You can provide users of the Self Service Portal with news and information, for example, to inform them about planned system maintenance.

News is created and managed in the "News" module . In order for news to be displayed in the Self Service Portal, the usage context "Customer" must be assigned to them. In addition, you can specify whether the display should take place before and/or after the login. Further information on news can be found in the Admin-Pro manual in the section [News](#) (see page 134) .

New News

* **Type:** Important

* **Title:** System maintenance

* **Display from:** 07.05.2023 01:00

* **Display to:** 08.05.2023 04:00

Keywords:

Teaser Text: System maintenance

* **Full Text:**
The next server maintenance will take place on 05/07/23 at 1 am and will last until 05/08/23 around 4 am

* **Usage Context:** Customer

* **Login Context:** Post Login, Pre Login

Related Tickets:

Fig.: Displaying news in the SSP

6.4 Ticket settings

6.4.1 Fallback for ticket title

Configuration key	ssp-ticket-title-fallback-pattern
--------------------------	-----------------------------------

In order for KIX to store tickets in the system, each ticket needs a title or a subject. In KIX, however, ticket templates can be created in which the user does not have to specify a ticket title or subject. In these cases, KIX initially uses the current date and time as fallback.

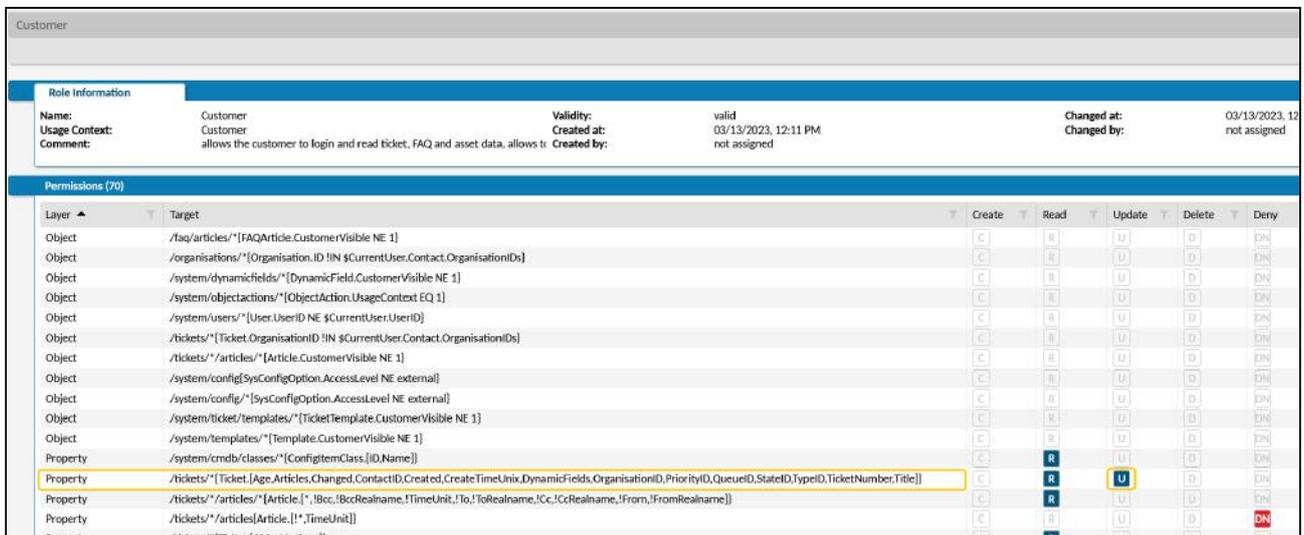
If another value is to serve as fallback instead of this date-time specification, you can store this value in the SysConfig key "ssp-ticket-title-fallback-pattern".

6.4.2 Enable editing of ticket header attributes (permission)

Holders of the role "Customer" initially have read-only rights to the ticket header attributes (priority, status, title, etc). To enable editing of the header attributes, you can grant the role "Customer" update rights to the header attributes. To do so, click on "Edit role" and set a check mark at "U - Update" at the property:

`/tickets/*{Ticket.`

`[Articles,Changed,ContactID,Created,CreateTimeUnix,DynamicFields,OrganisationID,PriorityID,QueueID,StateID,TypeID,TicketNumber,Title]}".`



Layer	Target	Create	Read	Update	Delete	Deny
Object	/faq/articles/*[FAQArticle.CustomerVisible NE 1]	C	R	U	D	DN
Object	/organisations/*[Organisation.ID !IN \$CurrentUser.Contact.OrganisationIDs]	C	R	U	D	DN
Object	/system/dynamicfields/*[DynamicField.CustomerVisible NE 1]	C	R	U	D	DN
Object	/system/objectactions/*[ObjectAction.UsageContext EQ 1]	C	R	U	D	DN
Object	/system/users/*[User.UserID NE \$CurrentUser.UserID]	C	R	U	D	DN
Object	/tickets/*[Ticket.OrganisationID !IN \$CurrentUser.Contact.OrganisationIDs]	C	R	U	D	DN
Object	/tickets/*articles/*[Article.CustomerVisible NE 1]	C	R	U	D	DN
Object	/system/config[SysConfigOption.AccessLevel NE external]	C	R	U	D	DN
Object	/system/config/*[SysConfigOption.AccessLevel NE external]	C	R	U	D	DN
Object	/system/ticket/templates/*[TicketTemplate.CustomerVisible NE 1]	C	R	U	D	DN
Object	/system/templates/*[Template.CustomerVisible NE 1]	C	R	U	D	DN
Property	/system/cmdb/classes/*[ConfigItemClass.{ID,Name}]	C	R	U	D	DN
Property	/tickets/*{Ticket.[Age,Articles,Changed,ContactID,Created,CreateTimeUnix,DynamicFields,OrganisationID,PriorityID,QueueID,StateID,TypeID,TicketNumber,Title]}	C	R	U	D	DN
Property	/tickets/*articles/*[Article.{,!Bcc,!BccRealname,!TimeUnit,!To,!ToRealname,!Cc,!CcRealname,!From,!FromRealname}]	C	R	U	D	DN
Property	/tickets/*articles[Article.{!,!TimeUnit}]	C	R	U	D	DN
Property	/tickets/*{Ticket.[ObjectAction]}	C	R	U	D	DN

Fig.: Extended role authorisation on Update for Customer

You can edit the specified properties to exclude or include selected header attributes from editing. The specified properties define on which objects the "Customer" has the set permissions.

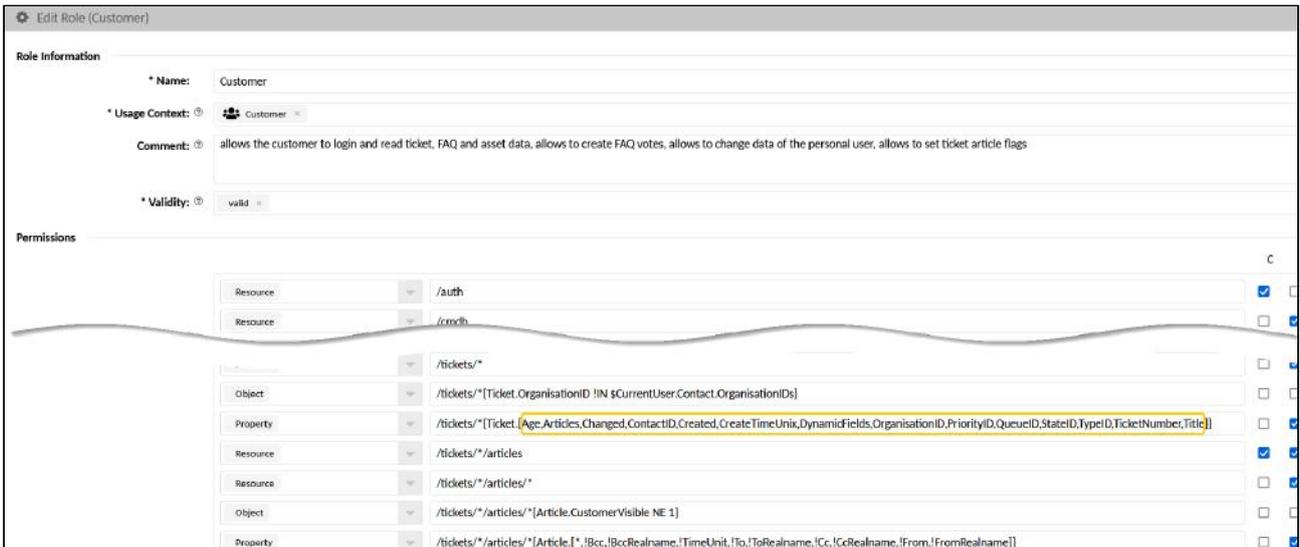


Fig.: Ticket attributes to which the "Customer" has permissions.

If required, you can add further properties to the role in order to granulate the permissions more strongly and also create further roles (see also Configuring and Assigning a Role).

Example: Split permissions on ticket header attributes. All specified ticket header attributes may be read (marked line below). But only the organisation and dynamic fields may be edited (marked line above).

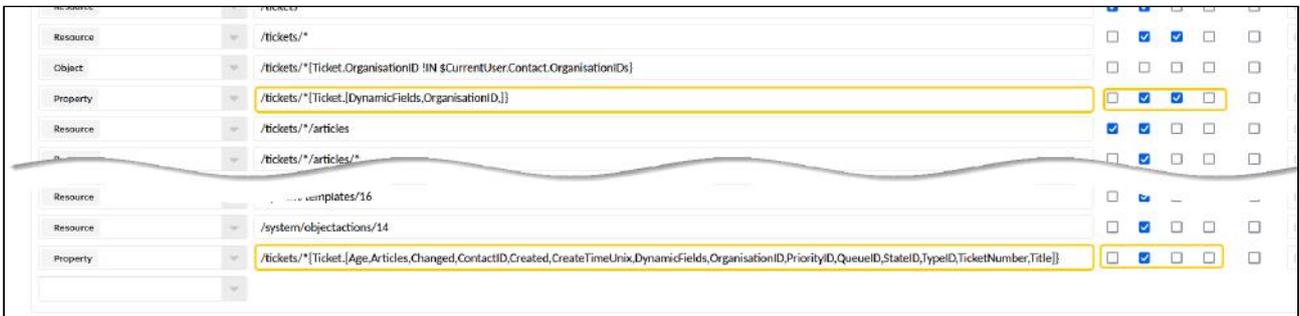
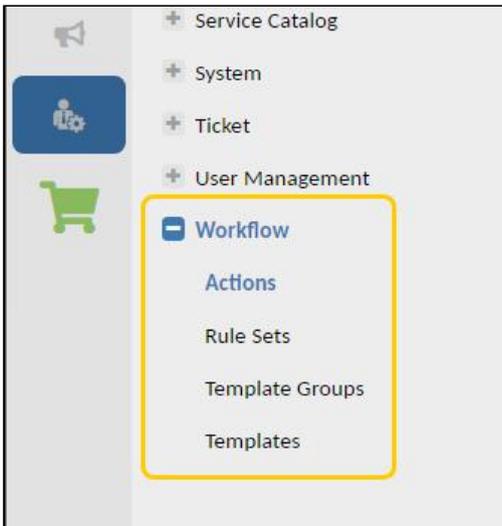


Fig.: Split permission for ticket header attributes

6.5 Templates, Actions, Rule Sets

Ticket templates, actions and workflow rulesets can also be configured for the Self Service Portal. The on-board tools of the agent portal are available for this purpose.



Content on this page:

- [Provision of ticket actions \(see page 111\)](#)
- [Provision of article actions \(see page 112\)](#)
- [Provision of templates \(see page 112\)](#)
- [Workflow Rulesets \(see page 114\)](#)

Fig.: The configuration of templates, actions and rulesets is done via the admin module of the agent portal

6.5.1 Provision of ticket actions

You can release ticket actions for the Self Service Portal (SSP). The configuration is done analogously to the ticket actions for the agent portal. The ticket actions released for the Self Service Portal are provided as buttons in the ticket header.

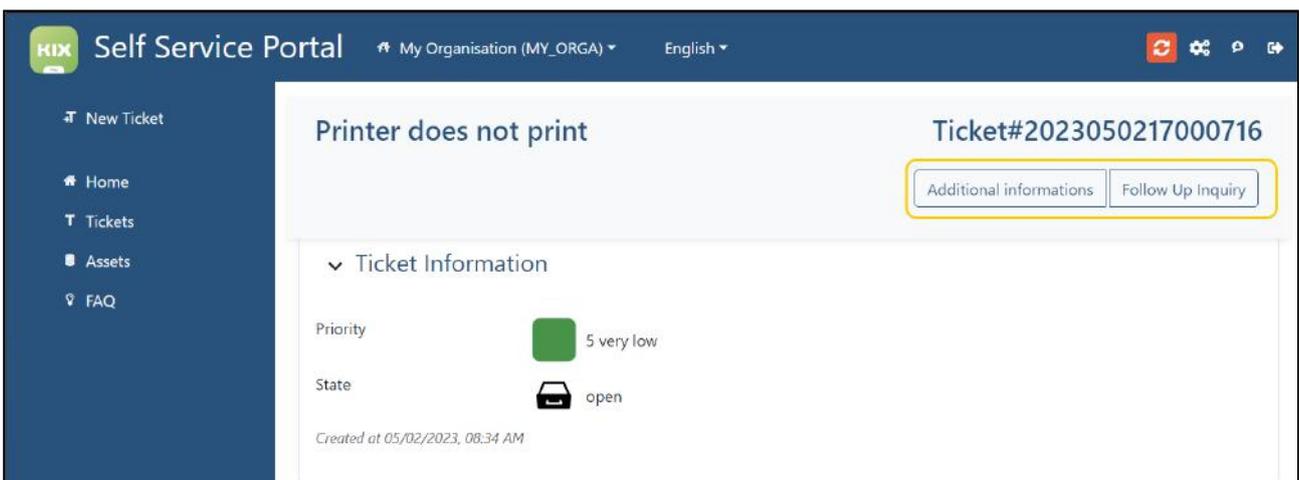


Fig.: Ticket actions made available in the SSP

Prerequisites for using a ticket action in the SSP:

- In order for a ticket action to be available in the Self Service Portal, it must at least be assigned to the usage context "Customer".
If the action is assigned to the usage context "Customer" and "Agent", the ticket action is available in both portals.
- The action must be assigned at least the role "Customer" in the filter.
- The action must be "valid".
- For the dynamic fields used, the option "Show in Customer Portal" must be activated in their configuration. (see also [Control visibilities in the SSP \(see page 98\)](#)).
If this option is not activated, the dynamic field is not displayed in the SSP, even if it is included in the configuration of the action. This allows for a differentiated provision of dynamic fields in ticket actions that are shared between agent portal and SSP.

Notes for ticket actions in the SSP:

- Ticket actions for the SSP are initially limited to setting and editing dynamic fields and to creating articles. The header attributes of tickets (status, type, etc.) can only be edited if the role "Customer" is extended by update rights (consider the effects!), see also [User settings for the SSP \(see page 96\)](#)).
- If fields are configured in a ticket action for which the SSP user has no authorisation, their use will fail.
- No emails can be sent from the SSP, only notes. Therefore, the "Channel | Chanel" is always set to Note, even if "Email" was selected in the configuration.
- Pre- and post-actions are not supported by the SSP.
- A ticket action can be called up again, e.g. to correct entries (e.g. change the rating from 3 to 4).
- The channel is fixed to "Note" (no emails can be sent from the Self Service Portal).
- The item sender type is fixed to "external" and is not displayed.
- The option "Show in Customer Portal" cannot be changed by the SSP user and is not displayed.
- If the configuration does not contain a channel/article specification or if these specifications are set in the background, their input fields are not available.
Thus, actions can be created without articles, e.g. initial action "Customer Feedback". This is available in the self-service portal on closed tickets and allows the customer to give a rating including a comment on a ticket.

6.5.2 Provision of article actions

The provision of article promotions is currently not yet possible.

6.5.3 Provision of templates

You can configure ticket templates for the Self Service Portal. The configuration is done analogously to the [templates \(see page 191\)](#) for the agent portal.

Grouping templates into template groups is also possible for the Self Service Portal. If the templates released for the SSP are assigned to a template group, this template group is displayed in the SSP. Template groups and templates are offered for selection above the editor.

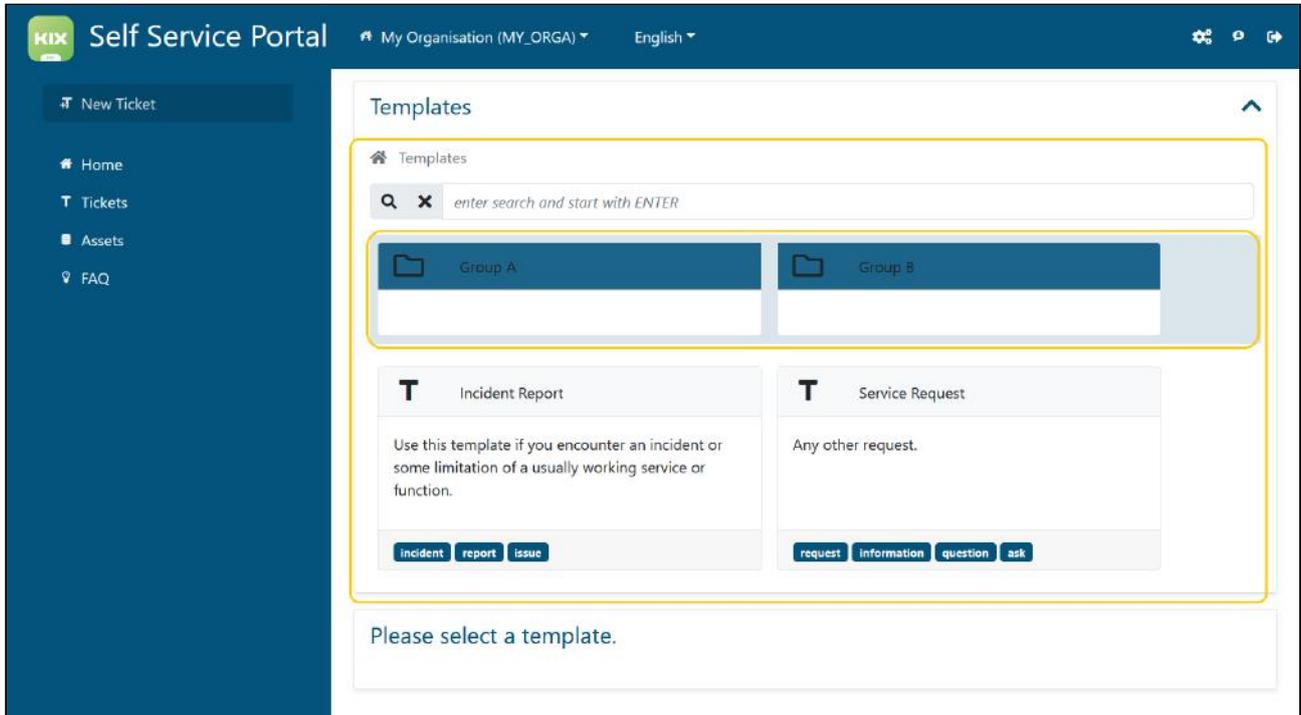


Fig.: Templates and template groups in the SSP

Prerequisites for displaying the template in the SSP:

- The template must be assigned to the usage context "Customer".
If the template is assigned to the usage context "Customer" and "Agent", the ticket action is available in both portals.
- The template must be assigned at least the role "Customer".
- The template and, if applicable, the template group must be "valid".
- The option "Show in Customer Portal" must be activated for the dynamic fields used. (see also [Control visibilities in the SSP \(see page 98\)](#)).

If this option is not activated, the dynamic field is not displayed in the SSP, even if it is included in the template configuration. This allows for a differentiated provision of dynamic fields in ticket templates that are shared between the agent portal and the SSP.

Notes on configuring templates for the SSP:

- Consider permissions: If fields are configured in a ticket template to which the customer user does not have permission, their use will fail for him.
- The header attributes of tickets (status, type, etc.) can only be edited if the role "Customer" is extended by update rights (consider effects!) → see also [user settings for the SSP \(see page 96\)](#)).
- The channel is fixed to "Note" (no emails can be sent from the Self Service Portal).
- The item sender type is fixed to "external" and is not displayed.

- The option "Show in Customer Portal" cannot be changed by the SSP user and is not displayed.
- If the template definition does not contain a channel/article specification or if these specifications are set in the background, their input fields are not available.

Thus, a ticket without an article can be created from a template (e.g. holiday request, which only contains the agent and the period).

6.5.4 Workflow Rulesets

You can control the behaviour of input and selection fields in the Self Service Portal using individually designed rulesets. You define the rules for this in the Admin module of the agent portal (menu Workflow > Rulesets). By specifying the condition "User.isCustomer", the rule is applied to the Self Service Portal. Information on this can be found under [Workflows > Rulesets > Section "Restricting the context of use \(see page 161\) "](#).

The workflow evaluation is only executed if valid rulesets are defined AND the workflow evaluation is activated. This behaviour is activated by default. If you want to deactivate this behaviour, you must insert the following code section in the configuration of the Self Service Portal (menu: "System > GUI configuration > Self Service Portal" > SysConfig key "self-service-portal-configuration")

```
"workflows": {  
  "enabled": false  
}
```

Links

- [Ticket actions \(see page 137\)](#)
- [Templates \(see page 191\)](#)
- [Template Groups \(see page 187\)](#)
- [Rule Sets \(see page 161\)](#)



6.6 GUI configuration of the Self Service Portal

You can configure the user interface of the Self Service Portal and thus determine the information provided there. For example, you can integrate dynamic fields and individual widgets into the user interface (GUI) of the Self Service Portal or remove superfluous or unwanted information from it.

As in the agent portal, the adjustments are made by modifying the corresponding configuration keys. The configuration keys for the Self Service Portal can be found in the menu System > GUI Configuration > Self Service Portal. Alternatively, you can call up the configuration keys in the menu System > SysConfig and edit them in an external editor (e.g. www.jsonformatter.io⁶).

Content on this page:

- [Dashboard Configurations](#) (see page 117)
 - [Home Dashboard](#) (see page 117)
 - [Ticket, Asset, FAQ Dashboard](#) (see page 118)
- [Configuration of the zoom views](#) (see page 118)
 - [Ticket Details](#) (see page 118)
 - [Asset Details](#) (see page 119)
 - [FAQ details](#) (see page 120)
- [New ticket](#) (see page 121)
- [Personal Preferences](#) (see page 121)
 - [Enable/disable password change](#) (see page 122)
 - [Enable/disable language selection](#) (see page 122)
 - [Configuration of the Personal Data](#) (see page 123)
- [Configuration examples](#) (see page 123)

⁶ <http://www.jsonformatter.io>

All configurations are located within the respective contexts. Therefore, only the top-level contexts can be edited (e.g. Home Dashboard, Ticket Details, Ticket Overview, etc.). The elements subordinate to a context cannot be configured.

- [Object-information-card-widget](#) (see page 123)
- [Table widget](#) (see page 124)
- [Implementing your own widget](#) (see page 125)

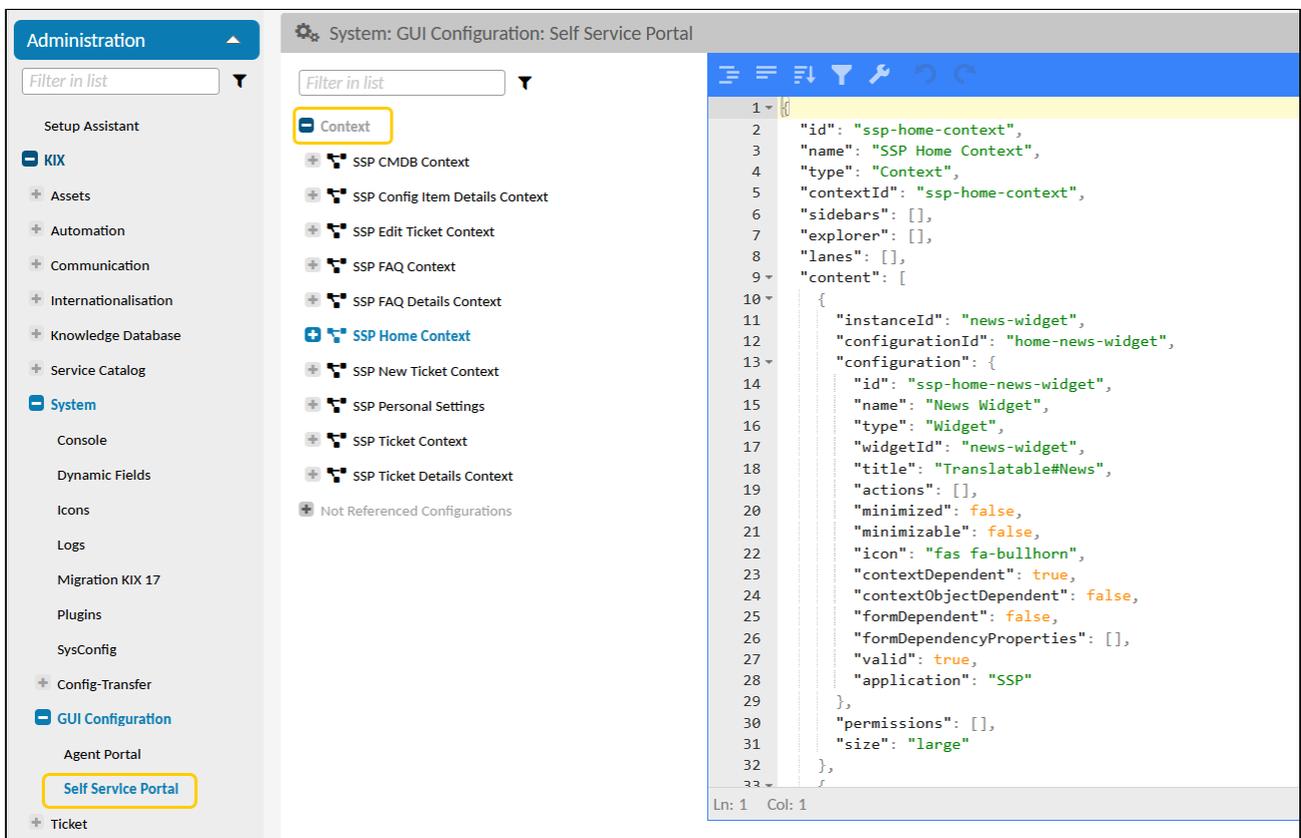


Fig.: Configuration of the GUI for the Self Service Portal

Please note!

After making changes to the configurations, always click on "Reload frontend configuration" so that the changes are transferred to the Self Service Portal. Users of the Self Service Portal may also have to reload the browser window.

Hints:

- The provision and usability of attributes and dynamic fields depends on the respective rights of the logged-in user. He must have at least read permissions on the object to be displayed (e.g. "Affected Asset", dynamic fields "Contact reference" and "Organisation reference").

Example:

If a column for displaying the Affected Assets is implemented in the ticket overview, only those Affected Assets will be displayed there to which the logged-in user has at least read permissions.

Tickets						
Ticket#	Affected Asset	Title	Created at ↓	State	Prio	
2023031517000026	A#175000001 - chprn13-bw	Headset defect	03/15/2023, 09:50 AM	open	High	
2023031517000017		Headset defect	03/15/2023, 09:49 AM	new	High	
2023031317000717	A#175000003 - chprn05-color	Printer doesn't print	03/13/2023, 12:23 PM	open	Low	

Fig.: Display of dynamic fields depend on user rights

- Depending on the user rights, various ticket information will not be displayed in the Self Service Portal, even if it is included in the ticket information (e.g. the agent and the person responsible for the ticket).
- Please also note the information on [Control visibilities in the SSP](#) (see page 98) .

6.6.1 Dashboard Configurations

6.6.1.1 Home Dashboard

SysConfig key	ssp-home-context
----------------------	------------------

The configuration includes the widgets located in the Home Dashboard: News, Tickets, Assets, FAQ. Customisation of these widgets is not required. However, you can implement your own widgets if required (see below).

Widget	InstanceID	Configuration
news	news-widget	not required
tickets	home-ticket-widget	not required
assets	home-asset-widget	not required
FAQ	home-faq-widget	not required

6.6.1.2 Ticket, Asset, FAQ Dashboard

These dashboards each contain an overview of their objects. These overviews are table widgets whose columns can be extended, for example, with dynamic fields (see below).

The configuration is done in the SysConfig key belonging to the context:

Context	SysConfig key	InstanceID	Configuration
Ticket Dashboard	ssp-ticket-context	ticket-list-widget	possible
Asset Dashboard	ssp-cmdb-context	asset-list-widget	possible
FAQ Dashboard	ssp-faq-context	faq-list-widget	possible

The display of dynamic fields and ticket attributes depends on the user rights.

Only attributes that are not version-dependent can be displayed in the asset dashboard, e.g. header data of the asset.

6.6.2 Configuration of the zoom views

6.6.2.1 Ticket Details

SysConfig key	ssp-ticket-details-context
----------------------	----------------------------

The configuration includes the widgets contained in the ticket details view:

Widget	InstanceID	Konfiguration
Ticket informationen	ticket-information-widget	possible
Additional information	ssp-dynamic-fields-widget	not required
Communication history	ticket-article-view	not required

The Ticket Information widget displays the header attributes of the ticket such as priority, status and creation



date. You can add more information (e.g. incident status) to this object-information-card widget if required (see below).

Please note that due to user rights, various ticket information is **not displayed** in the Self Service Portal, even if it is included in the ticket information. This includes, for example, the agent and the person responsible for the ticket.

It is not necessary to implement dynamic field values in the "Ticket information". These will be displayed in the widget "Additional information", provided that

- the option "Show in customer portal" is activated in the configuration of the dynamic field,
- the field has a value,
- the user has at least read permission.

If required, you can implement further widgets with individual contents in the ticket details (see configuration options at the end of the page).

Info: Dynamic fields of the type Checklist: Initially, checklists in the widget "Ticket information" are only displayed as a numerical value (2/5). If you want the progress bar to be displayed here, set the value `"ComponentID": "dynamic-field-checklist-cell"`. In the widget "Further information", the checklist is initially displayed as a progress bar.

Ticket actions

Ticket actions are available in the ticket detail view. If required, these can be used by the customer user, e.g. to send queries to the helpdesk or to add additional information to the ticket via dynamic fields.

Initially, the ticket action "query" is available in the Self Service Portal, which creates a new ticket item and is related to the ticket action "edit ticket". You can configure further ticket actions and make them available in the Self Service Portal (see also [Workflow > Actions \(see page 137\)](#)). Please note that the header attributes of the ticket cannot initially be changed by the customer user. If this is nevertheless desired, you must give the role "Customer" update rights or create new roles if necessary (see also [User Settings for the SSP \(see page 96\)](#)).

6.6.2.2 Asset Details

SysConfig key	ssp-config-item-details-context
----------------------	---------------------------------

The configuration includes the widgets contained in the asset details view:

Widget	InstanceID	Configuration
Asset informationen	config-item-information-widget	possible

Widget	InstanceID	Configuration
Versions informationen	config-item-version-information-widget	not required

The Asset Information widget initially displays the header information of the selected asset. The widget is an object-information-card widget whose contents you can configure as needed to provide additional asset information to the customer user (see below). Note that only attributes that are not version-dependent can be displayed, e.g. header data of the asset.

The Version Information widget provides an overview of the current version of the selected asset. It is permanently implemented and does not need to be changed.

6.6.2.3 FAQ details

SysConfig key	ssp-faq-details-context
----------------------	-------------------------

The configuration includes the widgets contained in the FAQ details view:

Widget	InstanceID	Configuration
FAQ informationen	faq-information-widget	possible
symptom	faq-symptom-widget	not required
cause	faq-cause-widget	not required
solution	faq-solution-widget	not required

The widget "FAQ Information" initially displays the basic information about the FAQ entry. The widget is an object-information-card-widget whose content you can configure as needed to provide the customer user with further basic information about the FAQ entry (see below).

The widgets Symptom, Cause and Solution do not need to be changed.

6.6.3 New ticket

SysConfig key	ssp-new-ticket-context
----------------------	------------------------

The dialogue for creating a new ticket is permanently implemented. Modification is not necessary.

Ticket templates

You can provide different ticket templates for creating new tickets, so that the customer user has separate input masks for each use case. The configuration of templates is done in the Admin module of the agent portal (see also [Workflow > Templates](#)) (see page 191).

6.6.4 Personal Preferences

SysConfig key	ssp-personal-settings
----------------------	-----------------------

Customer users in SSP have the possibility to make personal system settings. For example, to regularly change their password or to set their preferred system language.



The configuration of the above-mentioned SysConfig key includes the widgets contained in the Personal Settings. You can define which options and information are provided.

Widget	InstanceID	Configuration
Change Password	ssp-personal-settings-password-widget	<ul style="list-style-type: none"> • Disable password change: Set "value" to "false" • Enable password change: Set disable/enable possible
Select language	ssp-personal-settings-language-widget	de-/activate possible
Personal data	ssp-personal-settings-user-info-widget	Configuration of information possible

6.6.4.1 Enable/disable password change

InstanceID	ssp-personal-settings-password-widget
-------------------	---------------------------------------

You can specify whether the option to change the password is available to the client user. Deactivating this option may be necessary, for example, if the user authentication of the customer users is done via LDAP/AD and the user is therefore not allowed to change his password in KIX itself.

- Activate: Set the value "value" in the above InstanceID to "true".
- Deactivate: Set the value "value" in the above InstanceID to "false".

```

{
  "instanceId": "ssp-personal-settings-password-widget",
  "configurationId": "ssp-personal-settings-password-widget",
  "configuration": {
    "id": "ssp-personal-settings-password-widget",
    "name": "Change Password",
    "type": "Widget",
    "widgetId": "ssp-personal-settings-password-widget",
    "title": "Translatable#Change Password",
    "actions": [],
    "minimized": false,
    "minimizable": true,
    "icon": "",
    "contextDependent": false,
    "contextObjectDependent": false,
    "formDependent": false,
    "formDependencyProperties": [],
    "valid": true,
    "application": "agent-portal"
  }
}

```

Fig.: Activated password change

6.6.4.2 Enable/disable language selection

InstanceID	ssp-personal-settings-language-widget
-------------------	---------------------------------------

You can determine whether the client user can select his or her preferred system language. Provision is analogous to the password change widget by disabling/enabling the widget.

- Activate: Set the value "value" in the above InstanceID to "true".
- Deactivate: Set the value "value" in the above InstanceID to "false".

6.6.4.3 Configuration of the Personal Data

InstanceID	ssp-personal-settings-user-info-widget
-------------------	--

You can define which user attributes are displayed in the widget and optionally provide additional content, such as dynamic fields provided at the contact. The configuration is done as in any other object-information-card-widget (see below) by adding or removing `{"values": [...]}` -code blocks.

The provision of dynamic fields depends on the user rights and whether the dynamic field is enabled for the SSP in its configuration.

In addition, you can enable or disable the display of the widget:

- Enable: Set the value "value" in the above InstanceID to "true".
- Deactivate: Set the value "value" in the above InstanceID to "false".

Tip: In order for a Dynamic Field to be displayed in the Personal Data, it may be necessary to reopen and save the Dynamic Field in the Admin Module for editing and/or to execute the console command `Console::Command::Maint::Cache::Delete`.

6.6.5 Configuration examples

Below you will find brief information on providing dynamic fields in widgets. Detailed information on GUI configuration can be found in the Admin Manual of KIX Start and KIX Pro.

6.6.5.1 Object-information-card-widget

To display the value of a Dynamic Field in an Info Widget, you can proceed as follows:

Open the SysConfig key of the context you want to edit (e.g. ssp-ticket-details-context).

1. Insert the following code block into the configuration of the key.
2. Place it within the section "rows[...]". The placement determines the order of the dynamic field in the widget.

Replace "DFName" with the name of the dynamic field you want to include.

```

{
  "title": "Translatable#DFName",
  "separator": true,
  "values": [
    [
      {
        "componentId": "ssp-object-detail",
        "componentData": {

```

```

        "name": "DFName"
    },
    "conditions": [
        {
            "property": "DynamicFields.DFName",
            "operator": "NE",
            "value": null
        }
    ]
}
]
},
],
},

```

3. If necessary, insert the code block several times if you want to provide further dynamic fields and proceed as described in step 2.
4. Click on Save and then on "Reload frontend configuration".
Afterwards, the Dynamic Field is integrated in the interface so that its value can be displayed.

References:

- Agent Portal > GUI Configuration > Object-information-card-widget.
- Configure an object-information-card-widget

6.6.5.2 Table widget

To add another column to a table in the Self Service Portal for displaying a Dynamic Field, you can proceed as follows:

1. Open the SysConfig key of the context you want to edit (e.g. ssp-ticket-context).
2. Insert the following code block into the configuration of the key.
Place it within the section " `tableColumns[...]` ". The placement determines the order of the column within the table.
Replace "DFName" with the name of the dynamic field you want to include.
Optionally, change other column parameters such as column width (attribute " `size` ")

```

{
  "id": null,
  "name": null,
  "type": null,
  "property": "DynamicFields.DFName",
  "showText": true,
  "showIcon": false,
  "showColumnTitle": true,
  "showColumnIcon": false,
  "size": 135,
  "sortable": true,
  "filterable": true,

```

```
"hasListFilter": false,  
"dataType": "STRING",  
"resizable": true,  
"componentId": null,  
"defaultText": null,  
"translatable": false,  
"titleTranslatable": true,  
"useObjectServiceForFilter": false,  
"valid": true,  
"application": "agent-portal"  
},
```

3. If necessary, insert the code block several times if you want to provide further dynamic fields and proceed as described in step 2.
4. Click on Save and then on "Reload frontend configuration".
The column is then added to the table so that the value of the dynamic field can be displayed in the table.

Info: If the dynamic field is a checklist whose progress bar you want to display, note the value "dynamic-field-checklist-cell" under componentID. To display an SLA criterion, use the componentID "sla-criteria-cell".

References:

- Agent Portal > GUI Configuration > Tables Widget.
- Configuration of dashboard tables
- Configuration of table widget "Suggested FAQ"

6.6.5.3 Implementing your own widget

To additionally include an individual object-information-card-widget in the Self Service Portal, you can proceed as follows:

Open the SysConfig key of the context to be edited (e.g. ssp-ticket-details-context).

1. Insert the following code block into the configuration of the key.
2. Place it within the section "content[...]". The placement determines the arrangement of the widget in the context.

```
{  
  "instanceId": "my-ticket-widget",  
  "configurationId": "my-ticket-widget",  
  "configuration": {  
    "id": "my-ticket-widget",  
    "name": "My Ticket Widget",  
    "type": "Widget",
```

```

"widgetId": "my-ticket-widget",
"title": "Translatable#My Ticket Widget",
"actions": [],
"subConfigurationDefinition": null,
"configuration": {
  "id": "4711",
  "name": "4711",
  "type": null,
  "valid": true,
  "application": "agent-portal",
  "avatar": [],
  "rows": [
    {
      "title": "Translatable#DF1",
      "separator": true,
      "values": [
        [
          {
            "componentId": "dynamic-field-value",
            "componentData": {
              "name": "DF1"
            },
            "conditions": [
              {
                "property": "DynamicFields.DF1",
                "operator": "NE",
                "value": null
              }
            ]
          }
        ]
      ],
      "values": [
        [
          {
            "componentId": "dynamic-field-value",
            "componentData": {
              "property": "DF2"
            }
          }
        ]
      ],
      "title": "",
      "style": "",
      "separator": false
    }
  ],
  "minimized": false,
  "minimizable": true,

```

```
"icon": "",  
"contextDependent": false,  
"contextObjectDependent": false,  
"formDependent": false,  
"formDependencyProperties": [],  
"valid": true,  
"application": "SSP"  
},  
"permissions": [],  
"size": "large"  
},
```

3. Replace "DF1" or "DF2" with the names of the dynamic fields to be displayed in the widget.
Under " `conditions` " you can specify under which conditions the field is to be displayed. In the example above, DF1 must not be empty in order to be displayed.
4. You can add more dynamic fields to the " `rows[...]` " section. To do this, use the code block from section "Configure object-information-card-widget".
5. Click on Save and then on "Reload frontend configuration".
After that, the context contains another widget with individual configuration.

References:

- Adding a widget to the dashboard
- Adding a Widget to the Sidebar

6.7 Layout configuration

Configuration key	ssp-layout-configuration
--------------------------	--------------------------

You can change the colour scheme, the title and the logos used for the Self Service Portal and thus adapt the presentation of the Self Service Portal to your corporate identity.

To do this, change the values in the key "ssp-layout-configuration" in the menu *System > GUI Configuration > Self Service Portal*:

- **Change colour presentation:** Replace the colour values (primary, secondary, accent) with the colour values you want. Only hexadecimal values are supported.
- **Change favicon and logo:** Convert your favicon or logo to base64 format and use it to replace the initial specifications. Only the base64 format is supported.

Various online tools help to encode your own graphics according to base64, e.g.:

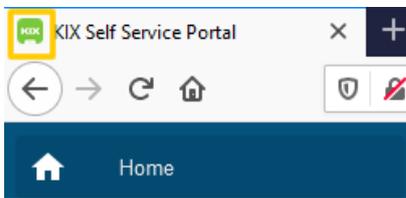
- <https://www.base64-image.de/>
- <https://www.base64encode.org/>
- <https://onlinejpgtools.com/convert-jpg-to-base64>
- **Change title:** Replace the title given under "title" with the title you want.

Content on this page:

- [Attributes in the layout configuration \(see page 128\)](#)
- [Organisation-specific layouts \(see page 130\)](#)
- [Example layout configuration \(see page 131\)](#)

6.7.1 Attributes in the layout configuration

Attribute	Description
ID	The ID (number) of the configuration block. It may only exist 1x in the entire key. Assign each configuration block its own ID if you are configuring layouts for several organisations.
name	Freely definable designation of the configuration block You can describe here, for example, for which organisation the configuration applies.

Attribute	Description
vHostPattern	<p>Regular expression of the VHost (e.g.: "<code>^service4companyA\\.kix\\.de*</code>")</p> <p>KIX checks the expression against the current host of the domain. If the VHost is not found, the next accessible VHost is used (usually the host of the default configuration).</p> <p>Note: The order is crucial. KIX works through the configuration from top to bottom; the first hit leads to the match. I.e.: If a RegEx ".*" is specified in the first configuration, this configuration is always used.</p>
primaryOrgID	<p>ID of the organisation (optional)</p> <ul style="list-style-type: none"> • If an ID is specified, the configuration is applied to the organisation with this ID. • If no ID is given, the configuration is applied to all organisations for which no other configuration applies (default and fallback).
design	<p>Configuration block for layout definition</p>
favicon	<p>Defines the favicon The favicon is displayed as a tab icon in the browser.</p>  <ul style="list-style-type: none"> • <code>content</code>: base64-String of the favicon (see above) • <code>contentType</code>: Data format of the favicon (e.g. <code>img/jpg</code>, <code>img/png</code>, <code>xml/svg</code>)
logo	<p>Defines the logo The logo is displayed in the login window and in the title bar.</p>  <ul style="list-style-type: none"> • <code>content</code>: base64-String of the logo (see above) • <code>contentType</code>: Data format of the logo (e.g. <code>img/jpg</code> oder <code>xml/svg</code>)

Attribute	Description																				
colorScheme	<p>Defines the colour scheme</p> <p>You can change the colours. Only hexadecimal values are supported. The colour changes only affect the "light mode", not the "dark mode".</p> <p>Default values:</p> <table border="1"> <thead> <tr> <th>Attribute</th> <th>Description</th> <th>Colour</th> <th>Colour code</th> <th>Use</th> </tr> </thead> <tbody> <tr> <td>primary</td> <td>primary colour</td> <td>blue</td> <td>#04537D</td> <td>Title bar, menu bar, headings</td> </tr> <tr> <td>secondary</td> <td>secondary colour</td> <td>grey</td> <td>#5B5B5B</td> <td>Cancel button</td> </tr> <tr> <td>accent</td> <td>accent colour</td> <td>green</td> <td>#82C826</td> <td>Label of the templates</td> </tr> </tbody> </table>	Attribute	Description	Colour	Colour code	Use	primary	primary colour	blue	#04537D	Title bar, menu bar, headings	secondary	secondary colour	grey	#5B5B5B	Cancel button	accent	accent colour	green	#82C826	Label of the templates
Attribute	Description	Colour	Colour code	Use																	
primary	primary colour	blue	#04537D	Title bar, menu bar, headings																	
secondary	secondary colour	grey	#5B5B5B	Cancel button																	
accent	accent colour	green	#82C826	Label of the templates																	
imprint	<p>Optional link to the imprint as HTML string, e.g.:</p> <ul style="list-style-type: none"> <code>"body" : "Imprint"</code> <code>"body" : "Imprint"</code> <p>The link is displayed in the footer of the Self Service Portal.</p>																				
title	<p>Defines the title in the Self Service Portal, e.g.: "title": "KIX Self Service Portal".</p> 																				

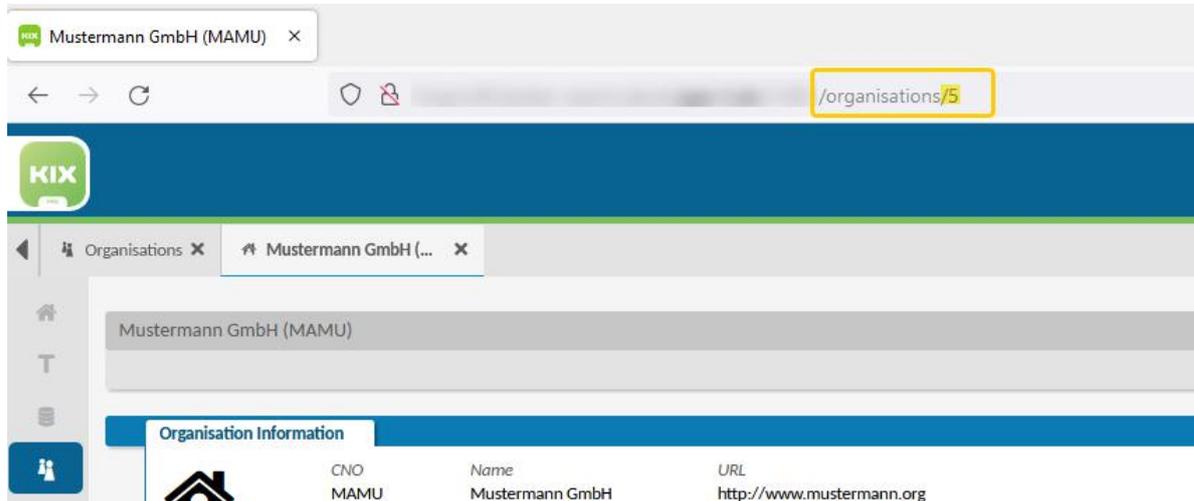
6.7.2 Organisation-specific layouts

You can define an individual layout for each organisation created in KIX and thus adapt the layout of the Self Service Portal to the Corporate Identity of the respective company:

1. Duplicate the initial JSON code (multiple times if necessary) and give each configuration block its own ID.

- In each case, enter the ID of the organisation to which this configuration applies under "primaryOrgID". You can specify multiple organisation IDs to use one layout for multiple organisations.

You can find the ID of an organisation in the URL when you have opened the detailed view of the organisation.



- In each configuration block, adjust the colour values, titles, logos and, if applicable, the link to the imprint.

Note

KIX works through the configuration from top to bottom and falls back to the default behaviour (default values) on its own if necessary. Therefore, place the default configuration with vHostPattern ".*" at the very bottom. It thus serves as a fallback and as the default configuration if the endpoint is not accessible or does not provide a response.

6.7.3 Example layout configuration

Example configuration for 2 organisations and default

```
{
  "id": 0,
  "name": "Layout for customer A",
  "vHostPattern": "service4companyA.kix...*",
  "primaryOrgId": [23, 24, 25],
  "design": {
    "favIcon": {
      "content": "<base64-string>",
      "contentType": "img/png"
    }
  },
}
```

```

    "logo": {
      "content": "<base64-string>",
      "contentType": "img/jpg"
    },
    "colorScheme": {
      "primary": "#ffcc00",
      "secondary": "#333333",
      "accent": "#f0f0f0"
    },
    "imprint": {
      "body": "<a href=\"https://url.to.your.imprint.com\">Imprint</a>"
    },
    "title": "Customer A – Self Service"
  }
},
{
  "id": 1,
  "name": "Layout for customer B",
  "vHostPattern": "service4companyB.kix....*",
  "primaryOrgId": "456",
  "design": {
    "favIcon": {
      "content": "<base64-string>",
      "contentType": "xml/svg"
    },
    "logo": {
      "content": "<base64-string>",
      "contentType": "img/png"
    },
    "colorScheme": {
      "primary": "#333300",
      "secondary": "#336600",
      "accent": "#339900"
    },
    "imprint": {
      "body": "<a href=\"https://url.to.your.imprint.com\">Imprint</a>"
    },
    "title": "Customer B – Self Service"
  }
},
{
  "id": 9999999,
  "name": "Default Layout",
  "vHostPattern": ".*",
  "primaryOrgId": [],
  "design": {
    "favIcon": {
      "content": "<base64-string>",
      "contentType": "image/png"
    },
    "logo": {
      "content": "<base64-string>",

```



```
        "contentType": "image/png"  
    },  
    "colorScheme": {  
        "primary": "#04537D",  
        "secondary": "#5b5b5b",  
        "accent": "#82C826"  
    },  
    "imprint": {  
        "body": "<a href=\"https://kixdesk.com\">Impressum</a>"  
    },  
    "title": "KIX Self Service Portal"  
  }  
}
```

7 News

News are notices and information displayed to the user before and/or after login. They can contain formatted text, tables, links and images and can be displayed time-controlled. As an administrator, you can use news, for example, to inform users about current malfunctions, upcoming changes or maintenance work.

The administration of news messages can be found in the module "News" . Agents with the role "News Manager" can also manage news messages there. A detailed description of how to create and manage news can therefore be found in the News Management chapter of the KIX Pro user manual.

News can optionally be

- in the Agent Portal and/or Self Service Portal (user context)
- before and/or after the user login (login context)

be displayed. The display is controlled by the selection of the context:

Usage Context: Agent + Login Context: Pre Login:

- The news messages are displayed above the input fields.
- A click on the plus symbol opens the message for viewing.

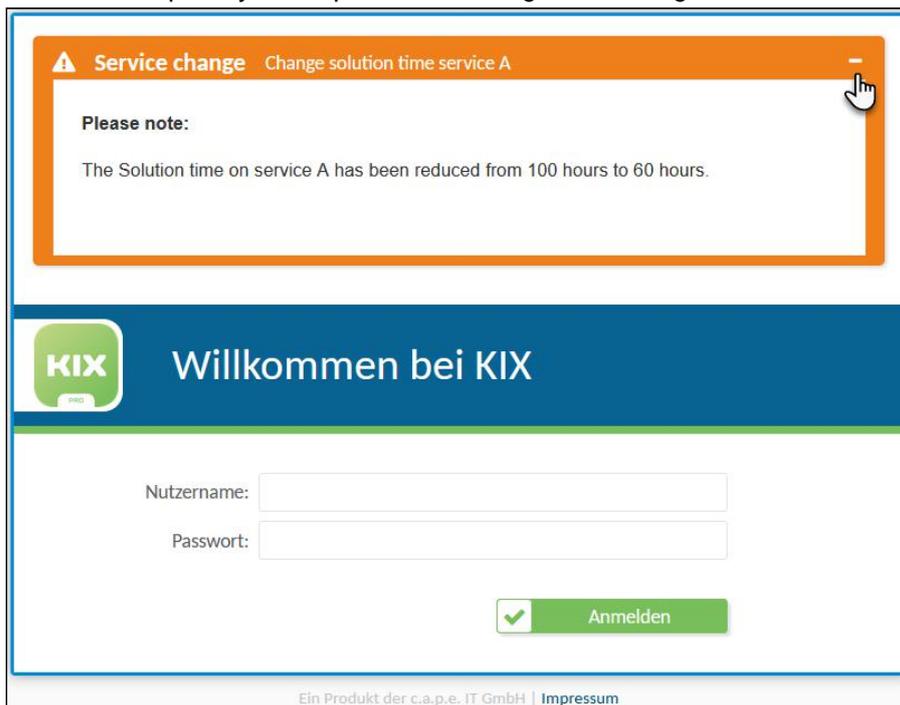


Fig.: News message before login to the agent portal

Usage Context: Agent + Login Context: Post login:

- The news messages are displayed in the user's personal toolbar.
- If errors occur when loading/creating/updating/deleting objects, these are also displayed.
- The messages are grouped according to their type.

- Clicking on a message opens it for full display in an overlay.
- Individual messages and message groups can be removed from the list by clicking on the cross.

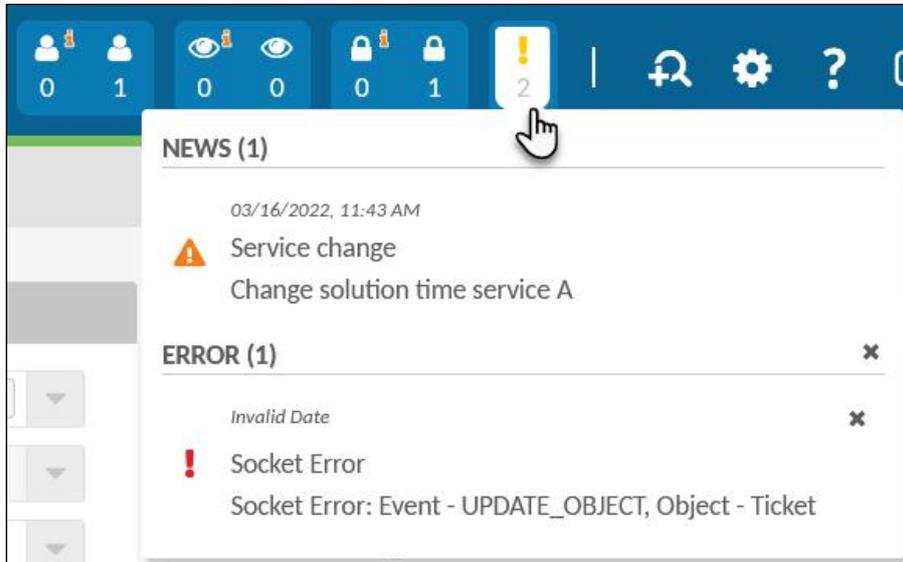


Fig.: News messages after login to the agent portal

Usage Context: Customer + Login Context: Pre Login:



Fig.: News message before login to the Self Service Portal

Usage Context: Customer + Login Context: Post Login:

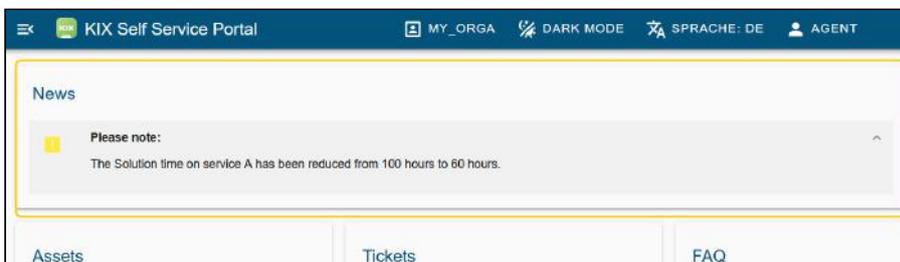


Fig.: News messages after login to the Self Service Portal

8 Workflow

Under the *Workflow* menu you will find advanced functions with which you can modify and extend the behaviour and properties of ticket-relevant forms. In this way, you can adapt tickets to the most diverse requirements and to your workflow.

Actions

In the *Actions* menu, you can configure ticket actions that are executed on existing tickets and their articles. Actions enable the processing of tickets depending on the current ticket state and thus provide the functions for the process-oriented use of KIX.

KIX distinguishes between ticket actions and article 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. Item actions are available as a lateral label on the right side margin of an item. You can configure under which conditions which action is available to whom.

Rule Sets

In the *Rule Sets* menu, you can create a set of rules for the dynamic control of form contents. With this rule set, you define the conditions and the behaviour of the form fields in the ticket dialogue as well as in templates and actions.

Templates

With *templates*, you can create templates with which the ticket creation screens in the Agent Portal and Self Service Portal are adapted to the respective context at the time of need. In a template, you can define which input fields are displayed and are mandatory or which default values are set with the template. It is thus possible to make entries adapted to the message type and to record frequently occurring requests more quickly and process them more efficiently.

Template groups

In the menu *Template groups* you can create categories to group templates thematically. The assignment of a template to a template group is done when creating a template. By assigning a template group to a superordinate template group, you can create a tree structure of template groups.

8.1 Actions

In the *Actions* menu, you can configure ticket and article-based actions. Actions are always applied to existing tickets and their articles. In contrast, templates are only applied to new tickets.

Actions are individually configured functions (macro actions) that are available under certain conditions. Initial actions are, for example, the standard functions such as closing and processing tickets or answering and forwarding e-mails.

By creating your own actions and defining who is available which action and when, you can adapt the system to the workflow of your company and provide the agents with exactly the functions they need for daily service provision.

8.1.1 Types of Actions

Depending on the referenced reference object, KIX differentiates between ticket actions and article actions.

Ticket actions

Ticket actions are actions that are applied to existing tickets. For example, they enable the processing of tickets depending on the current ticket status and thus provide the basis for a process-oriented use of KIX. The editing options on the ticket can depend on the content, editing status and the calling user.

Ticket actions can be dialogue-based or non-dialogue-based:

- **Non-dialogue-based actions** trigger a background action when clicked. With these so-called one-click actions, tickets can be filled with predefined values and/or closed immediately. Non-dialogue-based actions can be, for example:
 - Approve leave request (set approval status and close ticket)
 - Move back to the service desk team (set team "service desk")
 - Forward a service request including attachments to the manufacturer and close the ticket at the same time.
 - Classify as spam and move to junk team (set team "Junk")
 - Wait 2 days (set wait status with time difference of 2 days)
 - Wait until next working day (set wait status with time difference of 1 day and target time "Start service time").
- **Dialogue-based actions** open an additional dialogue window for entering ticket information. For example, to enter further ticket information in a specific incident or to enter the target time. You can individually configure which form fields the dialogue [contains](#) (see page 145) .

Ticket actions can be configured for both the agent portal and the self-service portal. They are then available as buttons of the ticket detail view.

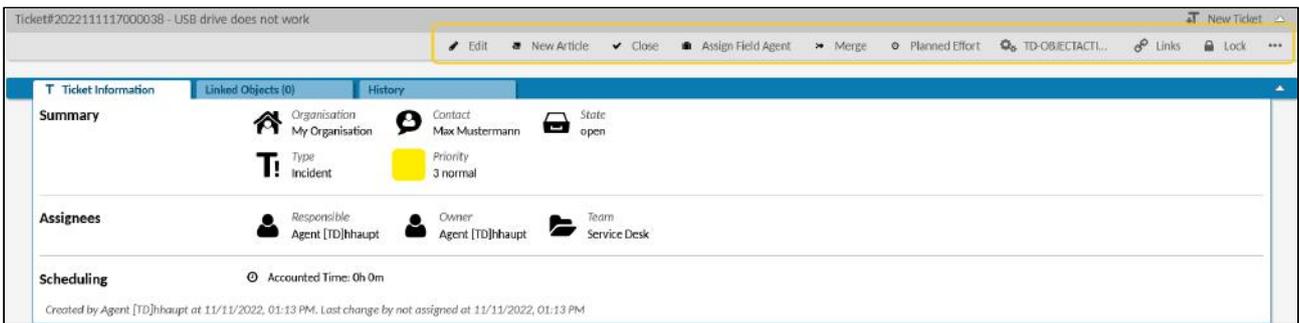


Fig.: Ticket actions in the ticket zoom view of the agent portal

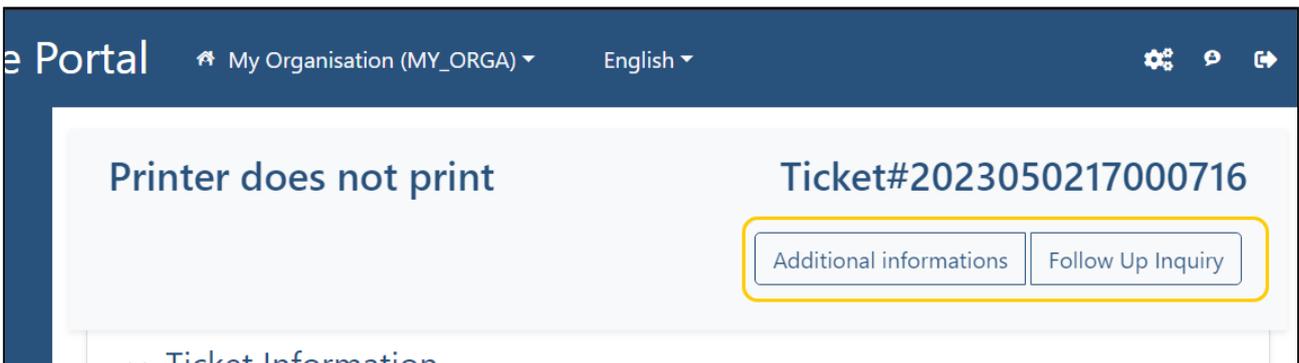


Fig.: Ticket actions in the ticket zoom view of the self service portal

In order for ticket actions to be available in the Self Service Portal, they must be assigned to the context "Customer" and the role "Customer". The further configuration is done in the same way as the ticket actions for the [agent portal](#) (see page 145) . Further information on the Self Service Portal can be found in the chapter: [Self-service Portal](#) (see page 93) .

8.1.1.1 Article actions

Article actions are always applied to an article and are therefore available directly on the article. For example, the actions for answering and forwarding e-mails or for sharing the article. These actions are initially preconfigured and can be adapted as needed. For example, to include the asset concerned in the reply email.

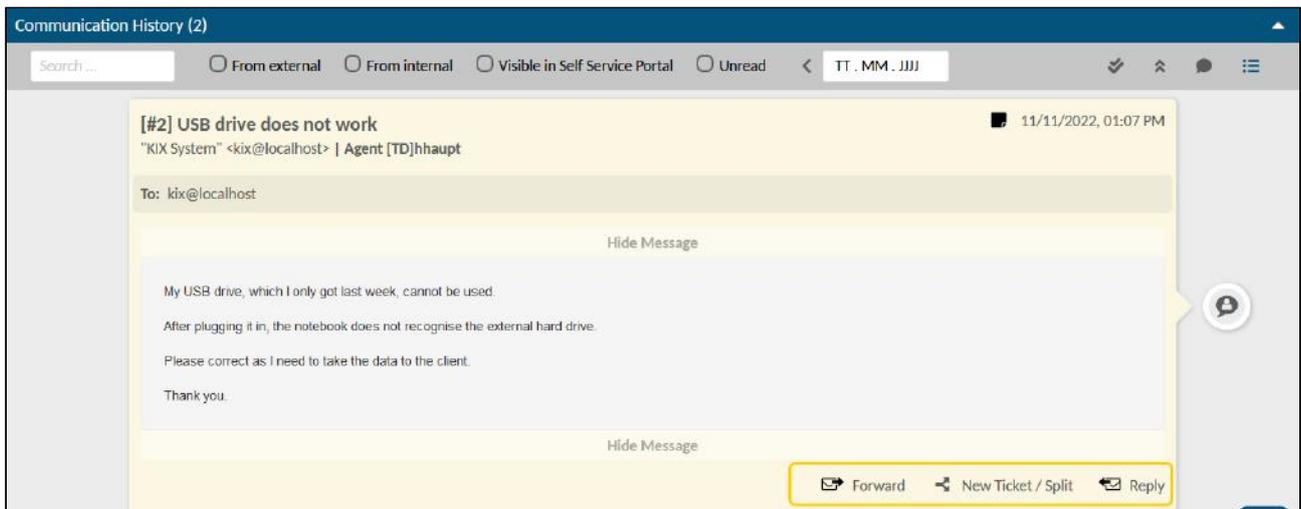


Fig.: Article actions on the article

The actions for editing and deleting an item are available, but initially deactivated (invalid). You can activate and reconfigure these actions as needed, for example, to define permissions.

You can create further article actions as required, e.g. to have different response options for forwarding or replying to an article. Further use cases for article actions can be

- Forwarding misdirected emails with feedback to the sender with just one click
- Forwarding emails to an external service provider with selection of the information to be attached or removal of internal information and simultaneous closing of the ticket
- Creation of multiple, different forwarding templates for different service providers or manufacturers. The recipient, affected service/SLA and the status of the ticket are already preselected, the variable waiting time is set by the agent.

Hint

- Article actions that are permitted on the ticket are displayed on all articles.
- Article actions can currently only be provided for the agent portal.

8.1.2 Actions initially delivered in KIX Pro

Action	Description	Hints
AppAssignWorkOrder	<p>The ticket action "Assign field agent" opens a dialog to add a job description including checklists to the ticket. This work description can be viewed and processed by the Field Agent in the Field Agent App.</p> <p>The dialog initially contains the following dynamic fields:</p> <ul style="list-style-type: none"> • RiskAssumptionRemark (transfer of risk) • WorkOrder (work instruction) • MobileProcessingChecklist010 (Checklist 01) • MobileProcessingChecklist020 (Checklist 02) <p>Note: Requires at least Field Agent App Version r1.2.x!</p>	<ul style="list-style-type: none"> • You can integrate additional dynamic fields in the dialog and remove dynamic fields that are not required. • You can adjust the checklists as needed. Change this directly in the configuration of the respective dynamic field.
ArticleDelete	<p>The item action "Delete" can be used to delete individual items of the <u>note</u> type., e.g. to delete redundant or spam articles.</p> <p>The deletion cannot be undone. Therefore, a confirmation prompt is displayed.</p>	<ul style="list-style-type: none"> • The action is initially deactivated (invalid). To activate it, open the action and set it to "valid". The action is then available on the item. • You can change the configuration of the action, e.g. to restrict the group of authorised persons.
ArticleForward	<p>Article action to "forward" emails. The configuration of the action determines which information is sent when forwarding.</p>	<p>You can reconfigure the action, e.g.:</p> <ul style="list-style-type: none"> • integrate additional (dynamic) fields and information into the e-mail. • add additional content to the text of the e-mail using placeholders. • control the sending of article attachments.

Action	Description	Hints
ArticleReply	<p>article action to "reply" to emails. The configuration of the action determines which information is sent when answering an e-mail.</p>	<p>You can reconfigure the action, e.g.:</p> <ul style="list-style-type: none"> integrate additional (dynamic) fields and information into the answer. add additional content to the text of the e-mail using placeholders.
ArticleUpdate	<p>The article action "Edit" can be used to edit individual articles of the <u>note</u> type, e.g. to correct or add information.</p>	<ul style="list-style-type: none"> The action is initially deactivated (invalid). To activate it, open the action and set it to "valid". The action is then available on the item. You can change the configuration of the action, e.g. to restrict the group of authorised persons.
Customer Feedback	<p>Ticket action to provide feedback. Is made available in the Self Service Portal on closed tickets. Customers can use it to give feedback on a ticket. Uses the initial dynamic fields "SatisfactionPoints" and "SatisfactionRemark".</p>	<p>You can reconfigure the action, e.g. to integrate additional (dynamic) fields and information into the evaluation.</p>
Follow Up Inquiry	<p>Is required for the Self Service Portal to add information to a request or to answer a question that has been asked.</p>	<p>You can change the configuration.</p>

Action	Description	Hints
Merge	<p>The ticket action "Merge" opens a dialog for combining two tickets into one ticket. For example, to attach a misdirected e-mail (source) to a ticket (destination).</p> <p>The dynamic field "MergeToTicket" is integrated in the dialog. It references all tickets, one of which can be selected as a destination and takes the ID of the selected ticket as a value.</p> <p>When the action is saved, the initial job "TicketMerge" is triggered, which</p> <ul style="list-style-type: none"> • sets the source ticket to the status "merged" • the event "TicketMerge" triggers on the source ticket • transfers the properties configured in the action to the target ticket • saves the article created in the action on the target ticket (channel selection) <p>If a ticket has the status "merged", no admin-configurable ticket and article actions are displayed on this ticket.</p>	<ul style="list-style-type: none"> • You can reconfigure the action and specify the ticket properties that are to be transferred from the "TicketMerge" job to the target ticket. • You can specify additional roles and filters to limit the availability of the action. The campaign can therefore only be made available to a certain group of users and/or under certain conditions. • You can define pre- and post-actions that are carried out before and after the action.
New Ticket / Split	<p>Via this article action, agents can create a new ticket from the article of a source ticket, e.g. to extract subtasks and assign them to the respective teams.</p> <p>Unless otherwise configured, the ticket template on which the source ticket is based is used for the child ticket and the article content and attachments are transferred. You can adjust this individually in step 3 of the configuration ("input fields").</p>	<ul style="list-style-type: none"> • The "pending time" set in the parent ticket is not adopted and must be set manually.

Action	Description	Hints
Planned Effort	<p>The ticket action "Planned effort" opens a dialog for entering and displaying the target time.</p> <p>The target time forms the basis for calculating the time on the ticket. The total of the times booked in the ticket is subtracted from the target time. The resulting time difference is displayed in the sidebar.</p> <p>The dialog contains the dynamic field "PlannedEffort" and shows the value saved in it. This value is initially set by the jobs "Auto Set Planned Effort (Incident)" and "Auto Set Planned Effort (Service Request)". You can deactivate the jobs if you do not want the time to be set automatically.</p> <p>When using the action, the value set in the field can be changed or entered by the person responsible for the ticket.</p>	<p>You can change the configuration of the action, for example to</p> <ul style="list-style-type: none"> • to integrate further input fields into the dialog • limit the use of the action by setting additional filters • and more <p>Note: Do not remove the "Planned Effort (min)" input field in step 4 of the assistant! Otherwise the time cannot be calculated correctly.</p>
Ticket Close	<p>The "Close" action opens a dialog for closing a ticket.</p> <p>The configuration defines which fields are included in the dialog and which values are set when the ticket is closed.</p> <p>The initial configuration contains the dynamic field "CloseCode", which provides the selection list "Closing code".</p>	<p>You can change the configuration, e.g. to integrate additional fields into the ticket close dialog.</p>

Action	Description	Hints
Ticket Edit	<p>The action "Edit" opens a dialogue for editing a ticket.</p> <p>The dialogue does not initialise the fields "Owner" and "Responsible", i.e. the fields are displayed empty after opening the dialogue. However, the agent/responsible person set on the ticket will be retained, as long as it is not changed when editing the ticket.</p> <p>Based on the initially set filters, the ticket edit action is only available on the ticket if</p> <ul style="list-style-type: none"> • the ticket is not locked • the ticket is locked and the executing user is the same as the editor (Ticket.Owner) or the responsible of the ticket (Ticket.Responsible). 	<p>You can change the configuration, e.g. to define the filter conditions for the provision of the action.</p> <p>Note: The configuration in the action has priority over the configuration in the SysConfig. The SysConfig key "<i>ticket-edit-form-group-data</i>" serves as a fallback.</p>
Ticket New Article	Ticket action "New" opens a dialog for creating a new article.	You can change the configuration, e.g. to integrate additional input fields into the new article dialog.

8.1.3 Create and configure actions

You can create and configure your own ticket and article actions. You can specify under which conditions which action is available to whom. For example:

- The person responsible for a ticket must be identical to the registered user.
- Certain assets must be affected.
- The ticket must be assigned to a specific customer.
- and much more

In addition, you can reconfigure the initial standard actions such as "New article", "Edit", "Close", "Forward" and "Reply" individually.

Content on this page:

- [Create, edit, duplicate and delete an action \(see page 145\)](#)
- [Configuration of an action \(see page 146\)](#)
 - [Action Information \(see page 147\)](#)
 - [Filter \(see page 150\)](#)
 - [Pre Actions \(see page 154\)](#)
 - [Input Fields \(see page 155\)](#)
 - [Post Actions \(see page 160\)](#)

8.1.3.1 Create, edit, duplicate and delete an action

To create an action:

1. In Explorer, navigate to *Workflow > Actions*. A table is opened in the content area, which lists all actions created in the system.
2. Click on "New Action" in the table. A form dialog opens in which you can create and configure a new action step by step (see below).
3. Finally save the action with "Save". The action is now available in the corresponding ticket interfaces.

To edit an action:

1. In the Explorer, navigate to *Workflow > Actions*. A table opens in the content area, which lists all actions created in the system.
2. Click on the action to be edited in the table. A form dialogue opens in which you can edit the configuration of the action step by step (see below).
3. Finally, save the action by clicking "Save". The changed action is now available in the corresponding ticket interfaces.

To duplicate an action:

1. In the explorer, navigate to *Workflow > Actions*. In the content area, a table opens that lists all actions created in the system.
2. Mark the action to be duplicated with a tick.

3. Click on "Duplicate" in the table. An independent duplicate of the selected action is created and opened.
The configuration of the duplicate corresponds to its source. You can change the configuration of the duplicate as required (see below - Configuring of an action).
The name of the duplicate is "Copy of [action name]" by default. You can keep this name or name the duplicate differently.
4. Finally, save the action with "Save". The action is now available in the corresponding ticket interfaces.

To delete an action:

1. In Explorer, navigate to *Workflow > Actions*. A table is opened in the content area, which lists all actions created in the system.
2. In the table, select the action to be deleted. Put a tick on this action. You can choose multiple actions.
3. Click on the "Remove" button in the table header and answer the security question with "Yes" if you are sure you want to delete the action.
All macros and sub-macros (pre- and post-actions) are deleted recursively. However, if internal references still exist, the action cannot be deleted. You will then receive a corresponding message.

As an alternative to deleting, you can set an action to "invalid" or "temporarily invalid". The action is then not available in the ticket interfaces. An invalid action can be set to "valid" again at any time.

 **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 pre- or post-action 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.

8.1.3.2 Configuration of an action

An action is created and edited step by step. 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. When you have finished entering all parameters, please click on the "Save" button.

Action Information

⚙ Edit Action

Action Information

< ● ● ● ● ● >

*** Usage Context:** ? KIX Agent × ▼

*** Reference Object:** ? 📄 Article × ▼

*** Name:** ?

Label: ?

Icon: ? 📁 Select image file 🗑

*** Behavior:** ? 📄 New Ticket × ▼

Rank: ?

User input required: ?

Comment: ?

*** Validity:** ? valid × ▼

Field	Description
Usage Context	<p>With the selected context of use, you define in which portal the action is available.</p> <ul style="list-style-type: none"> • Agent: The action is available in the agent portal. • Customer: The action is available in the Self Service Portal. The role "Customer" is also required. <p>If an action is to be available in both portals, duplicate a fully configured action and adjust the usage context and the roles accordingly in the duplicate.</p> <p>Note: As of version 29, only customer OR agent can be selected as the context of use. For already existing actions, this means: If an action is edited that was once created for both usage contexts, only the Agent usage context is adopted. Therefore, duplicate the action(s) and set the usage context "Customer" and the role "Customer" in the duplicate.</p>



Field	Description
Reference Object	<p>Choose whether to apply the action to a ticket or an item.</p> <ul style="list-style-type: none">• Ticket: Creates a ticket action. Ticket actions change and add properties on the ticket.• Article: Creates an article action. Item actions change and add properties to the item. Article actions are currently not supported by the Self Service Portal. Therefore, this option is only available in the "Agent" usage context.
Name	Internal name for the action. This is the name of the action in the system.
Label	Labeling of the button in the program interface.
Icon	Optionally, select an icon with which the action is graphically identified. The icon is displayed on the button to the left of the label.

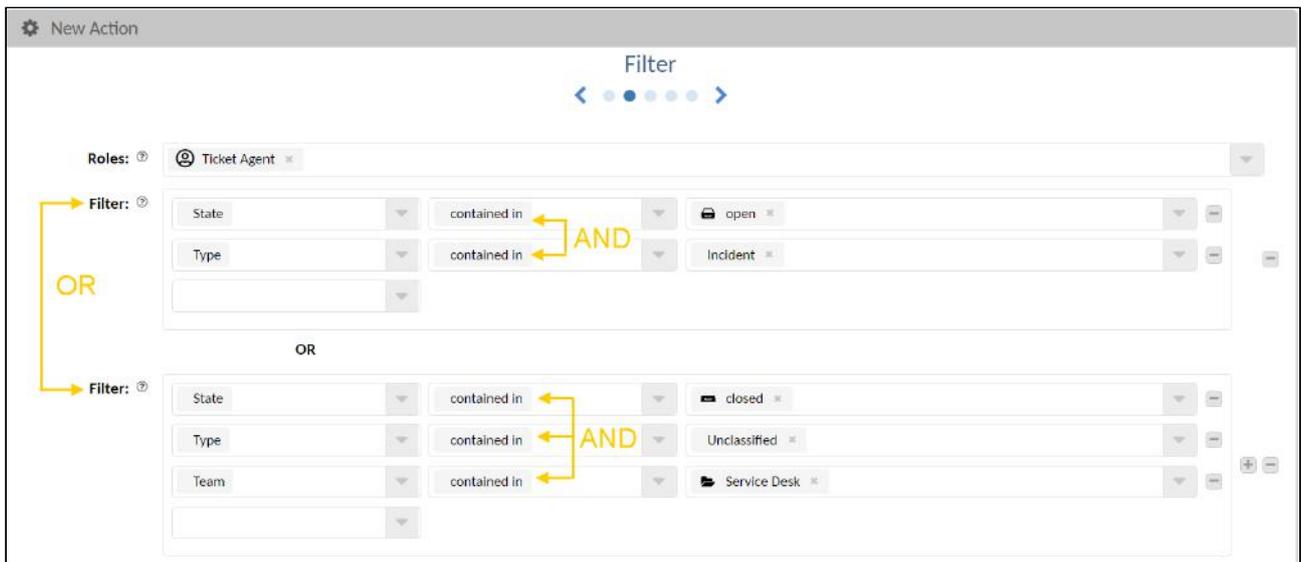
Field	Description
Behavior	<p>The selected behaviour controls what should happen when the action is executed. The selection options depend on the selected reference object.</p> <ul style="list-style-type: none"> • New Ticket: Creates a new ticket when the action is executed. <ul style="list-style-type: none"> • The new ticket can be linked to the source ticket as a child ticket. To do this, enter the placeholder "<KIX_TICKET_TicketID>" in the input field "ParentTickets" (see also Configuration of the item action "New ticket/split"). The resulting child tickets are displayed in the ticket details of the parent ticket. • Pre- and post-actions are not executed. They are therefore deactivated in the configuration of the action. • Unless configured otherwise, the ticket template on which the source ticket is based will be used for the (child) ticket and the item contents and attachments will be transferred. A "Pending time" set on the parent ticket will not be transferred and must be set manually. • If this option is deactivated in existing actions, any pre- and post-actions set will be deleted and no longer executed. • Edit Ticket: Makes changes to an existing ticket when the action is executed. Use this option if values are to be set or further entries are to be made on an existing ticket. • New Article: Creates a new article on the ticket when the action is executed. • Edit Article: Makes changes to the respective article (see also initial inactive action "ArticleUpdate"). <p>Note: When configuring your own actions, make sure that the new tickets contain at least the required mandatory information on the ticket.</p>
Rank	<p>Defines the order of the actions button in the ticket detail view. An action with a lower rank is placed further to the left. "0" is a valid value. If no value is specified, the rank is given the value NULL and the action is placed on the left. If the priority is the same, the actions are displayed in alphabetical order. The default order of actions is:</p> <ol style="list-style-type: none"> 1. To edit 2. New article 3. Conclude 4. Other actions without templates

Field	Description
User input required	<p>Select whether the action requires user input.</p> <p><input checked="" type="checkbox"/> Generates a <u>dialog-based action</u>. If the user selects the action, a dialog opens in which further information can be entered. These are only adopted when the user clicks on "Save".</p> <p>In the following steps of the actions configuration you create the template for the dialog and specify which fields are displayed in the dialog and define their behavior and values.</p> <p><input type="checkbox"/> Generates an action <u>without a dialog</u>. Use this option if no user input is required, e.g. to archive a ticket by pressing a button.</p> <p>So that it is recognisable that the action has been carried out, you can optionally define a system confirmation. To do this, activate the option "Confirmation request" and store the text to be displayed under "Confirmation content". Then a message with the same text appears as soon as the action has been carried out.</p>
Comment	Save notes about the action here (optional)
Validity	Set the action to "valid". Actions set to "invalid" or "temporarily invalid" cannot be used by the user.

Filter

By setting filters, you define the conditions under which an action is available.

You can combine the filters. The filter blocks are OR-linked. The filters within a filter block are AND-linked.



The screenshot shows the 'New Action' configuration page. At the top, there is a 'Filter' section with navigation arrows. Below it, the 'Roles' section is set to 'Ticket Agent'. There are two filter blocks, each starting with 'Filter:'. The first filter block contains three conditions: 'State' (contained in), 'Type' (contained in), and 'open' (selected). The second filter block contains three conditions: 'State' (contained in), 'Type' (contained in), and 'closed' (selected). The word 'AND' is placed between the 'Type' and 'open' conditions of the first filter, and between the 'Type' and 'closed' conditions of the second filter. A large yellow 'OR' is placed between the two filter blocks, indicating that either filter must be satisfied. The 'Service Desk' condition is also present in the second filter block.

Examples can be:

1. The ticket action "Approve leave request" is only available to those responsible for the ticket if the subject of a ticket contains "leave request". This button is hidden in all other tickets.
2. The article action "Forward to manufacturer" is only available on the article if the owner belongs to a certain team and if there is an incident with selected assets.
3. The ticket action is only available if the registered user has the role of ticket agent and is also the person responsible for the ticket.

Field	Description
Roles	<p>Select which roles you want to grant permission for this action.</p> <p>For an action to be available in the Self Service Portal, at least the "Customer" role is required.</p> <p>Use this selection only for roles to extend their permissions with regard to actions. Otherwise the authorizations will be restricted (e. g. for the roles of "Super User" and "System Admin").</p> <p> Note the authorizations within the roles!</p> <p>Example: You configure a ticket edit action and select the role "Ticket Reader". The action is not displayed because the "Ticket Reader" role is not authorized to process tickets.</p> <div style="border: 1px solid red; padding: 10px; margin-top: 10px;"> <p> Important!</p> <p>A "READ" is explicitly set for templates & actions for the selected roles. Therefore, never assign this authorization to roles that have more than read rights to actions & templates! Otherwise the authorizations will be restricted (e. g. superuser, admin).</p> </div>

Field	Description
Filter	<p>You can optionally further restrict the availability of the action by setting filters. In doing so, you determine the conditions under which an action is available.</p> <p>The basis of the filter is the selected role. In the above For example, the action is only available to members of the "Ticket Agent" role who are also set as responsible for the ticket.</p> <p>The filters are specified in the same way as the logic in the complex search (see user manual):</p> <ol style="list-style-type: none"> 1. Column: Filter attributes for tickets 2. Column: search operators (e.g. :. contained in, starts with, ends with, is equal, etc.) 3. Column: Form fields and values. <p>The action is available on all tickets that match the parameters specified here. If you specify several filters, the filters are linked to one another by a so-called logical AND link. You can for example, specify that an action is only available in tickets that have the type "unclassified" AND the status "closed" AND are from the "Support" team.</p> <p>If the reference object "Ticket" is selected, all ticket-based filters and dynamic fields (except for the 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). You can select valid and invalid objects and attributes. The use of KIX placeholders (e. g. <KIX_CURRENT_UserID>) is possible. Placeholders must be entered manually in the selection field and confirmed with ENTER. In addition, relative time information can be defined (e. g. created within the last 24 hours).</p> <p>Note: If dynamic fields set to "invalid" are specified, they will not be used during execution.</p>
	Removes the filter
	Inserts further filters

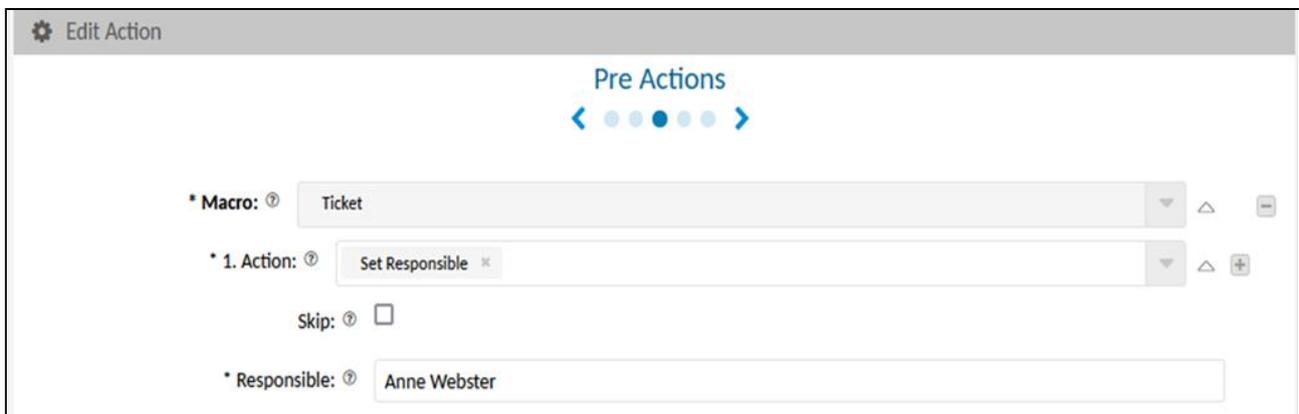
Pre Actions

Pre-actions are configured macros that are executed in the background as soon as the action is clicked:

- In dialogue-based actions:
 - Pre-actions are executed before the dialogue window is opened.
 - If a user cancels an action, he or she receives a notice that an action was executed in the background.
 - Skip this step if you do not want to define any upstream actions.
- In non-dialogue-based actions (one-click action):
 - Define what should happen in the background as soon as the user clicks on the action. For example, move the ticket to the "Junk" team, set a wait status or the person responsible for the ticket.
 - After the action has been carried out, the agent receives a corresponding message.
 - **Note:** In non-dialogue-based actions, the "input fields" in step 4 are not taken into account; their control is exclusively via the pre- and/or post-actions.

Pre actions are not supported by the Self Service Portal. Therefore, they cannot be configured if "Customer" has been selected as the context of use.

The use of placeholders (e.g. <KIX_CURRENT_UserID>) is possible. Placeholders must be entered manually in the selection field and confirmed with ENTER.



The screenshot shows the 'Edit Action' configuration window. At the top, there is a 'Pre Actions' section with a progress indicator showing the current step. Below this, the configuration is as follows:

- * Macro:
- * 1. Action:
- Skip:
- * Responsible:

Hints

- Pre actions are carried out with the user ID of the calling user. This means that the actions cannot be used directly using a backend API call.
- Please enter a difference to the waiting time in seconds if you set a waiting status (e.g. "Waiting for reminder" or "Waiting for successful closing"). This will automatically set the date so that the ticket can be closed without manually specifying the time.

✓ **Tip**

The "Loop" macro action can be used to execute macros on referenced tickets and to nest them as required.

Input Fields

Specify which fields should be included in the dialog. You use this to design the structure and content of the dialog window in a dialog-based action. This step can be skipped for non-dialog-based actions.

Note: This configuration is only taken into account if the option "User input required" is activated in the action information (step 1).

The screenshot shows the 'Edit Action' interface with the 'Input Fields' configuration. The configuration is divided into several sections:

- Input Fields:** A list of fields to be included in the dialog. Three fields are highlighted with yellow boxes and numbered 1, 2, and 3:
 - Field 1: State (dropdown), Channel (checkbox), CreateFAQSuggestio... (checkbox).
 - Field 2: fix in mask (dropdown), set in mask (dropdown), set in mask (dropdown).
 - Field 3: closed (dropdown), Note (checkbox).
- Advanced Options:** Two sections labeled '>> Advanced Options' are visible below the first field.
- Field Options:** A section labeled '<< Field Options:' is visible below the first field.
- Hint:** A section labeled 'Hint' with the text 'Customised information text for the field'.
- Code Editor:** A code editor at the bottom shows a JSON configuration for a field:

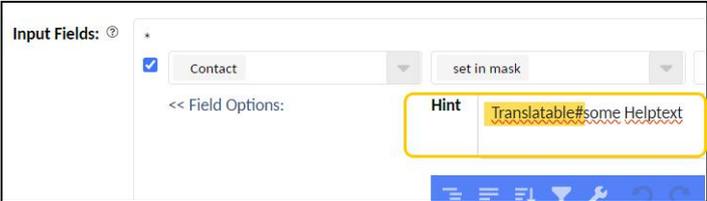

```

1 - [
2 - {
3   "option": "FIELD_NAME",
4   "value": "CreateFAQSuggestion"
5 - }
6 - ]
      
```

Column		Description
<p>1</p>	<p>Field selection</p>	<p>Select the fields here that are to be placed in the dialogue. You can choose from all of the ticket-relevant input fields as well as the dynamic fields.</p> <p><input checked="" type="checkbox"/> A check mark in front of the input field declares the field as a mandatory field.</p> <p>Notes on field selection:</p> <ul style="list-style-type: none"> • Channel-specific fields (To, CC, BCC, Subject, Article Content, Attachments) additionally require the "Channel" field in the template definition. Only when a channel has been selected when using the template, the channel-specific fields will be displayed and filled with default values. • Take permissions into account: If fields are configured in a ticket action to which the user (agent or customer) does not have permission, their use will fail. • If a ticket receives the status "Waiting for [...]", a dynamic time in the future will be displayed as a default value for the target waiting time. This preassignment is initially determined from the current time plus the value stored in the SysConfig. You can adjust the default value for the target waiting time by changing the key "<i>Ticket::Frontend::PendingDiffTime</i>" in the menu <i>System > SysConfig</i>. <p>Please note: A target waiting time must be specified in the ticket so that it can also be saved without manually changing the time specification.</p> <ul style="list-style-type: none"> • Please also note the instructions for providing ticket actions in the chapter: Templates, Actions, Rule Sets (see page 111) . • Dynamic field "Affected Services" in the Self Service Portal: There is no check of permissions. <p>This means: If services are set "as a fixed value" or "in the background" in an action, the service will be set even if the Self Service Portal user does not have the corresponding permissions.</p>

Column		Description
<p>2</p>	<p>Properties</p>	<p>Specify how the field or its value is integrated into the dialogue:</p> <p>Set in mask: The field is displayed in the ticket form and initialised with the value already set on the ticket. This value can be overwritten by a default value in the dialogue.</p> <p>Fix in mask: The field and its value will be displayed in the ticket mask. The value cannot be edited. E.g. to display a non-editable value in the ticket.</p> <p>Set hidden: The field and its value are not displayed in the ticket screen. For example, to specifically exclude fields from editing or to add invisible values to the ticket.</p> <p>Clear in mask: Shows the field in the dialogue, but initialises it with an empty value. This enables the resetting of values. In connection with mandatory entries, a conscious decision by the user is forced.</p> <p>Relative time:</p> <p>Date/time fields can be provided with a relative initialisation. The following options are possible:</p> <ul style="list-style-type: none"> • Set in mask (relative time) • Fix in maks (relative time) • Set hidden (relative time) <p>Behaviour of the dynamic field as described above, but information as relative time value (e.g. 10 minutes). Available for dynamic fields of the types Date and DateTime. Can be used, for example, for waiting times with relative time differences or tickets with relative times for plan start/plan end.</p> <p>Based on the time of use, the specified time value is added to the current time and set as a date (each time the action is called again).</p> <p>The use of KIX placeholders is supported so that a number specified in a dynamic field (field type Text) can be read out and used as a time value.</p> <p>The defined waiting times have a higher priority than the setting in the SysConfig key "<i>Ticket::Frontend::PendingDiffTime</i>".</p>

Column		Description
<p>3</p>	<p>Define values</p>	<p>Optionally, specify a value with which the field will default when the template is used. Leave the value blank if you do not want to give the field a value.</p> <p>The choices in dropdowns are in direct context to your field selection in column 1.</p> <p>Notes on using placeholders:</p> <ul style="list-style-type: none"> • You can specify placeholder variables to put variable, ticket-specific values in the field. Enter the placeholder manually in the selection field and confirm with ENTER. Placeholders for dynamic fields are specified according to the following scheme: <KIX_TICKET_DynamicField_NameDesDynamicField>. • When editing tickets, the substitution of placeholder variables refers to the source ticket; for new tickets, the fields are empty (for ticket placeholders). Fields that reference other objects such as contacts, agents or priority expect an ID as value. This means that when using KIX placeholders, the ID must be specified (e.g. KIX_CURRENT_UserID>). For dynamic fields, the value specified under "Key". • In order for placeholders in ticket forms to reference correctly, it is important to distinguish between user placeholders (KIX_xxx_UserID) and contact placeholders (KIX_xxx_ContactID) and to use them correctly. For example, the contact field needs a placeholder that references CONTACTS - even if the contact can be an agent and thus a user. This means, for example, for placeholders in the template "Default - New Ticket Dialog": <ul style="list-style-type: none"> • Placeholder for the contact in the ticket: <ul style="list-style-type: none"> • correct: <KIX_CURRENT_ContactID> • wrong: <KIX_CURRENT_UserID> • Placeholder for the currently logged in user as agent or responsible: <ul style="list-style-type: none"> • correct: <KIX_CURRENT_UserID> • Fields that reference other objects such as contacts, agents or priority expect an ID as value. This means that when using KIX placeholders, the ID must be specified (e.g. KIX_CURRENT_UserID>). For dynamic fields, the value specified under "Key".

Column	Description
<p>4</p> <p>Advanced Options</p>	<p>"Note" field: You can store a customised help text for the field. This allows you, for example, to inform the agents which data a dynamic field expects.</p> <p>If a help text is specified, a question mark symbol is displayed in the field. The help text is displayed after clicking on the question mark.</p>  <p><i>Fig.: Customised information text in the "Contact" field</i></p> <p>The help text is saved in the field configuration of the respective action so that a different help text can be noted for the field for each action.</p> <p>The initial fields of a template already contain localised help texts, which are provided via the weblate. If required, you can replace these with your own customised help text.</p> <p>You can maintain the translation of your texts yourself in the <i>KIX > Internationalisation > Translation</i> menu (see Translations). Mark the help texts to be localised with the preceding <code>Translatable#</code> :</p>  <p><i>Fig.: Texts to be translated are labelled with a preceding "Translate#"</i></p> <p>Note: Fields added by workflow rule are currently not considered.</p> <hr/> <p>Editor: You can specify an array of further configuration options for the field in JSON format, e.g. specific loading options, the object type to be loaded, etc.</p> <p>Different options are supported depending on the selected field. Some configuration options are set automatically in order for the fields to work correctly, e.g. for the dynamic field "Affected Asset".</p> <ul style="list-style-type: none"> • Option: Specification of the attribute • Value: value of the attribute

Column	Description
	<p>With field options, for example, the users available for selection in the field can be limited to certain roles.</p> <p>Further information can also be found in the Templates (see page 191) chapter (sections: Input fields, Notes on field selection, Notes on field options).</p>

Post Actions

Here you can define which actions are to be carried out immediately after the action has been carried out. For example, to close the ticket after the vacation request has been rejected. The configuration is carried out in the same way as the "Pre Actions" (see above).

Post actions are carried out as soon as the "Save" button is clicked in dialog-based actions. In non-dialog-based actions, post actions are carried out as soon as the action is clicked.

Post actions are not supported by the Self Service Portal. Therefore, they cannot be configured if "Customer" has been selected as the context of use.

You can use KIX placeholders (e.g. <KIX_CURRENT_UserID>). Enter the placeholder manually in the selection field and accept the entry with ENTER.

Skip this step if you do not want to define any subsequent actions.

Hints

- Post actions are carried out with the user ID of the calling user. This means that the actions cannot be used directly using a backend API call.
- Please enter a difference to the waiting time in seconds if you set a waiting status (e.g. "Waiting for reminder" or "Waiting for successful closing"). This sets the date automatically so that the ticket can also be closed manually.

8.2 Rule Sets

You can control the behaviour of input and selection fields and thus better support the workflow within dialogues. This is done in the menu "*Workflow > Rule Sets*" by means of individually designed rulesets. Each ruleset contains any number of freely definable rules. The rules define the conditions for the existence and behaviour of selection and input fields in ticket forms, templates and action dialogues.

For example, you can control that fields are dynamically shown or hidden in a specific dialogue under certain conditions.

- fields are dynamically shown or hidden
- values are set or removed from fields
- only permissible input and selection values are available
- a field becomes a mandatory field or not
- etc.

The rules directly affect and influence the initial form configurations and the templates and action dialogues you have configured. For example, if you have included a dynamic field in a template and hide it with a rule, the dynamic field is not displayed, even if it is included in the template. By specifying conditions, you can determine under which circumstances the field is hidden. In this way, you can control the behaviour, appearance and content of the forms/dialogues in a granular and very targeted way.

Content on this page:

- [Structure](#) (see page 163)
- [Sequence of processing](#) (see page 164)
 - [Enable/disable processing of rulesets](#) (see page 165)
- [Create or edit a Ruleset](#) (see page 166)
- [Rulebook](#) (see page 167)
- [Overview of the commands](#) (see page 169)
 - [Properties of the transaction object](#) (see page 171)
 - [Restricting the context of use](#) (see page 172)
 - [Operators](#) (see page 172)
 - [Conditions \(if\)](#) (see page 173)
 - [Statements \(then\)](#) (see page 177)
- [Functions](#) (see page 186)
 - [DynamicFields.contains](#) (see page 186)

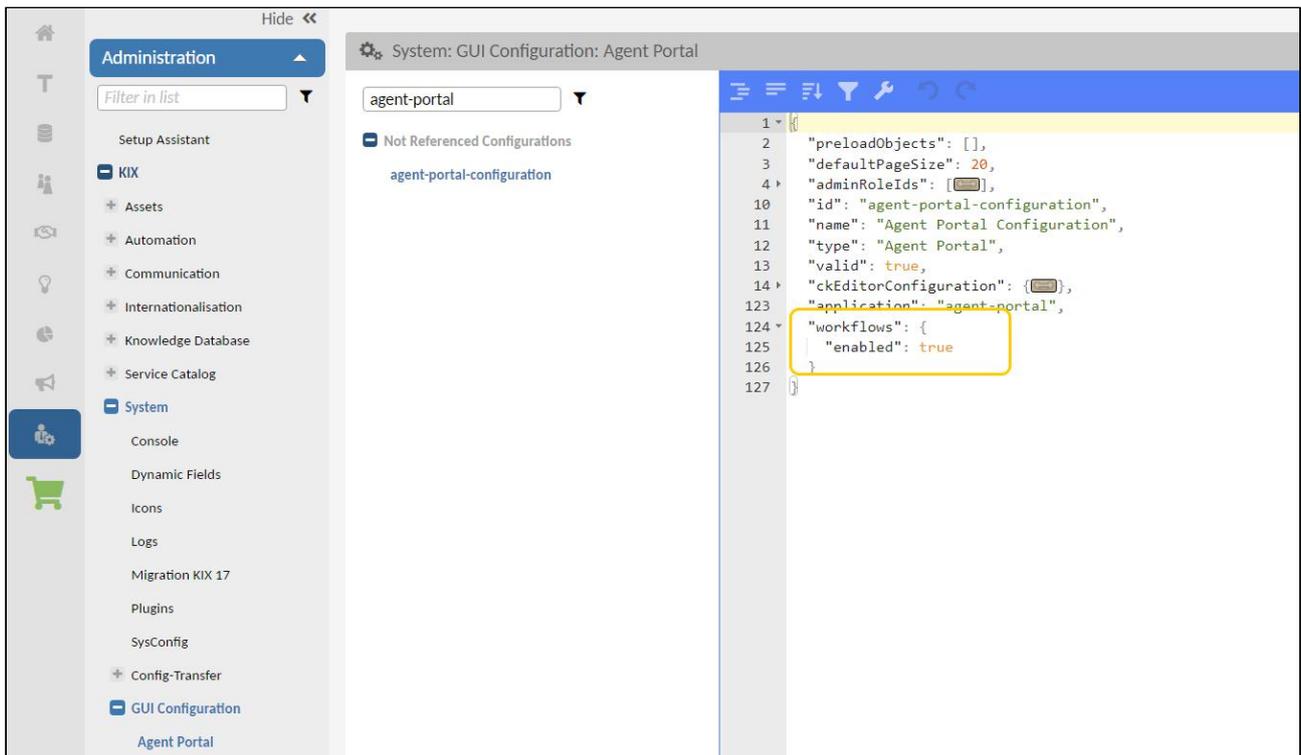


Fig.: Activation workflow evaluation

Attention!

Rules can - depending on the configuration - be mutually exclusive and collide with each other! Therefore, consider the complexity and work out the rules and rule sets conscientiously in advance to avoid this.

8.2.1 Structure

A ruleset is a list of individual rules. Each rule is structured as follows:
(parts in round brackets are optional)

Syntax

```
Rule "<Name>" (on <Object>) (if <Condition>)
  (<Commands>)
End
```

It applies:

- Each rule begins with the keyword `Rule` .
- Each rule ends with the keyword `End` .
- A rule is separated from the following rule by a blank line.
- Lines may contain indentations.
- For better readability, actions within a rule can be separated from the operands by any number of spaces (see example 2).
- `<Commands>` is a list of commands (see below "Overview of commands").
- Each command is on one line.
- Comments can be written between the rules and between the `<Commands>` .
- Comments begin with the `#` character.
- The `<Name>` of the rule can be arbitrary and consist of several letters, words and numbers (usable to designate the rule in a meaningful way).
- The `<Name>` of the rule may exist identically in several rulesets, but only once per ruleset.
- Rules can optionally be applied to a specific `<object>` (currently only on tickets; keyword: `on`).

Example 1

```
# this is a comment on the following rule
Rule "stop here, on one condition" if DB.TicketID == 10
  Stop
End

Rule "stop all rules" on Ticket
  # stop here!
  Stop
End
```

Example 2

```
# Rule is executed when type "Incident" is set on the ticket
Rule "Type" on Ticket if TR.TypeID eq "Incident"
  Set Priority "1 very high"
  Set StateID 2
  Set Owner "mmuster"
  Hide DynamicFields.TestField
End
```

8.2.2 Sequence of processing

The **rulesets** are processed from top to bottom in alphanumeric order of naming.

Dynamic Access Control: Rulesets (6)					
<input type="checkbox"/>	Name	Comment	Rule Count	Validit	
<input type="checkbox"/>	1 Ruleset		3	valid	
<input type="checkbox"/>	A_Ruleset		6	valid	
<input type="checkbox"/>	RuleSet 2		4	valid	
<input type="checkbox"/>	RuleSet 1		8	valid	
<input type="checkbox"/>	RuleSet A		2	valid	
<input type="checkbox"/>	RuleSet B		5	valid	

Fig.: Sequence of processing rulesets

By specifying a preceding or following letter or number, you can control the order of processing, e.g.:

- A_Ruleset, B_Ruleset, C_Ruleset
- Ruleset1, Ruleset2, Ruleset3
- 1_RulesetA, 2_RulesetA, 1_RulesetB, 2_RulesetB
- etc.

We recommend that you create your own naming convention in advance.

The **rules** defined in a ruleset are processed from top to bottom - both the individual rules in the rule set and within each rule. The name of the rules is irrelevant for the order of processing. What counts is the order of the rules in the ruleset.

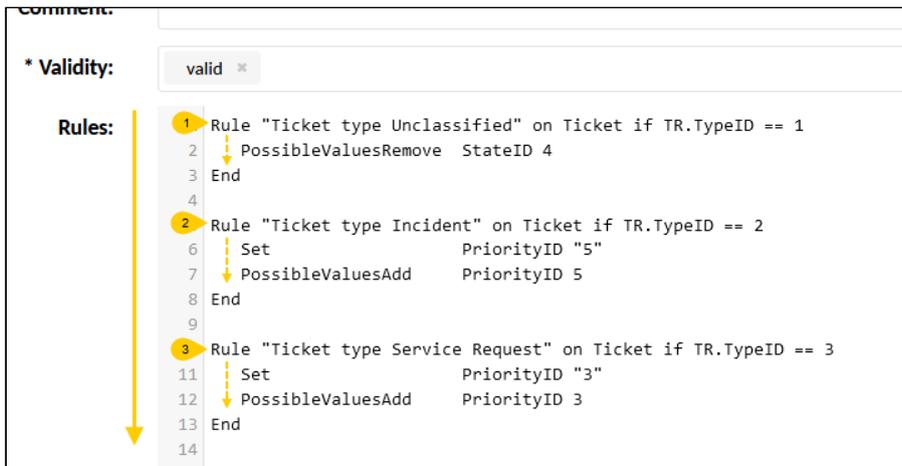


Fig.: Sequence of processing rules within a ruleset

Both rulesets and rules cascade. This means that if a property of an object is specified more than once, the property specified last applies (except for "PossibleValuesAdd" and "PossibleValuesRemove" - see below). In your configuration, make sure that rules do not unintentionally influence or override each other.

In the following example, priority 5 wins, priority 2 is replaced by priority 3, which in turn is replaced by priority 5.

Example: Cascading rules

```
Rule "alpha" on Ticket if TR.TypeID == 1
  Set PriorityID 2
End

Rule "beta" on Ticket if TR.TypeID == 1
  Set PriorityID 3
End

Rule "gamma" on Ticket if TR.TypeID == 1
  Set PriorityID 5
End
```

8.2.2.1 Enable/disable processing of rulesets

SysConfig key

agent-portal-configuration

The processing of workflow rules creates an input lock for checking the input values after each input that starts a rule. The check takes place after an input or selection field has lost its focus. In multi-select fields and in dynamic fields of the type "Table", the check takes place after clicking on "Apply". Entering the item content does not start an evaluation of the workflow rules.



The behaviour is only active if valid rulesets are defined AND workflow evaluation is activated. The workflow evaluation is activated in the menu "System > GUI configuration > Agent portal" in the SysConfig key "agent-portal-configuration".

8.2.3 Create or edit a Ruleset

To create or edit a ruleset, navigate in the Admin module to the Workflow > Rulesets menu. Click on "New Ruleset" to create a new ruleset or click on an existing ruleset to open it for editing. Define the ruleset in the dialogue that opens:

Field	Description	Example
Name	Enter a meaningful name for the ruleset here.	
Comment	Optionally, write a comment on the ruleset, e.g. to explain its function.	
Condition	Optionally, specify a global condition for the entire ruleset. The entire ruleset is only executed if this condition applies.	<ul style="list-style-type: none">• <code>User.isCustomer</code> (entire ruleset applies only to the Self Service Portal)• <code>User.isAgent</code> (entire ruleset applies only to the agent portal)• <code>User.isCustomer && TR.TemplateID == 3</code> (entire ruleset only applies in the Self Service Portal when using the ticket template with ID 3) <p>(see also below: Restriction the context of use).</p>

Field	Description	Example
Rules	<p>Define the individual rules here in accordance with the above structure and note the order in which they are processed (from top to bottom).</p> <p>The rules can be defined in a very targeted and granular way.</p>	<pre>Rule "<Name>" on Ticket if User.isCustomer && TR.TemplateID == 3 <Commands 1,2,3> End Rule "<Name>" on Ticket if User.isAgent && TR.TemplateID == 4 <Commands 4,5,6> End</pre>

 Edit Ruleset

*** Name:** myNewRuleset

Comment: Displays various dynamic fields in the SSP when option 2 is selected in the select box.

*** Validity:** valid ✕

Condition: User.isCustomer

Rules:

```

1 Rule "myRule1" on Ticket if TR.DynamicFields.contains(DFSelectionSingle, 1)
2 Show DynamicFields.DFTextArea
3 Enable DynamicFields.DFTextArea
4 Show OwnerID
5 Enable OwnerID
6 Show ResponsibleID
7 Enable ResponsibleID
8 Show SLAID
9 Enable SLAID
10 Show ChannelID
11 Enable ChannelID
12 Show TypeID
13 Enable TypeID
14 End
15
16 Rule "myRule2" on Ticket if TR.DynamicFields.contains(DFSelectionSingle, 2)
17 Required DynamicFields.DFTextArea
18 Show DynamicFields.DFTextArea
19 Enable DynamicFields.DFTextArea

```

Fig.: Example of a ruleset

8.2.4 Rulebook

All data stored in KIX is in the database. When a form is opened, the data is read from the database and temporarily displayed in the form 1. In the form, this data can now be changed and supplemented with further data. The form data is not written back to the database until it is saved 2.

Therefore, the set of rules distinguishes between:

- **Database object:** the (actual) values already set in the database or in the backend.

- **Transaction object:** the values displayed or entered in an open dialogue.

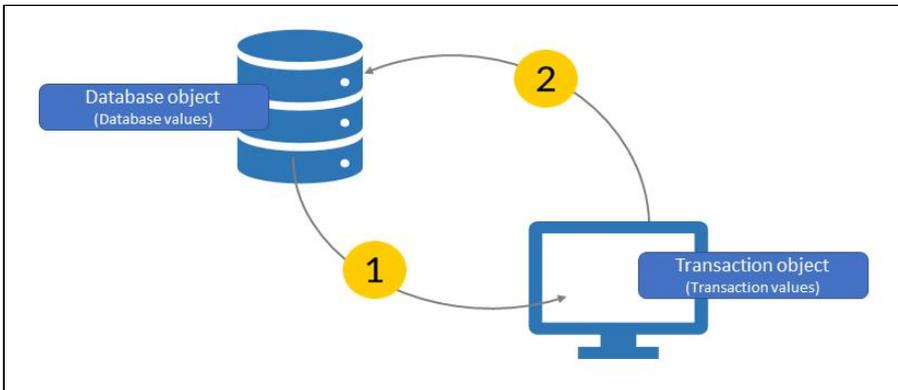


Fig.: Illustration of difference between database and transaction values

This means:

- If you are addressing values that are stored in the database or in the backend, then use the prefix " DB. "
- If you are addressing values that are displayed or entered in an open form, use the prefix " TR. "

Example of transaction values

```
# If the ticket type "2" (Incident) is selected in the opened ticket dialogue, then
set the priority to "1" (very high).
Rule "Type" on Ticket if TR.TypeID == 2
  Set PriorityID 1
End
```

Example of database values

```
# If the ticket type "2" (Incident) is saved on the ticket, set the priority to
"1" (very high).
Rule "Type" on Ticket if DB.TypeID == 2
  Set PriorityID 1
End
```

8.2.5 Overview of the commands

Legend for the following overviews

Placeholder	Description	Example
	OR separator Used in the overview when different possibilities exist, For example, when a value OR a list of values can be specified.	"<Value> <List>"
<Array>	An array of values can be specified.	Set BCC "john.doe@nomail.org", "manfred.lobster@example.com", "anothermailaddress.example.com"
<Attribute>	A KIX attribute can be specified, see also: Overview of KIX attributes	Set StateID 2
<AttributList>	A list of attributes can be specified.	InputOrder PriorityID, StateID, DynamicFields.DFText
<Commands>	A list of any commands. Each command must be on a separate line.	Rule ... Show PriorityID Set PriorityID 5 ReadOnly PriorityID End
<Condition>	The expression of a condition can be specified. Keyword: if (see below: " Conditions (see page 173) ") The condition must be fulfilled for the rule to be executed.	Rule "<Name>" on Ticket if TR.TypeID == 2 && TR.StateID == 3 <Commands> End

Placeholder	Description	Example
<Count>	Numerical value for a number	MinSelectable DynamicFields.DFSelection 2
<ErrorMessage>	Text of an error message after checking the user input by means of RegEx.	RegExp DFText "[A-Za-z0-9]+\$" "Only numbers and letters possible"
<List>	A list of values can be specified.	PossibleValuesAdd PriorityID 4,5,6
<Name>	The name of the rule (alphanumeric). Must be set in inverted commas. May contain special characters and spaces.	Rule "Set values 3, 4 & 5" on Ticket if TR.TypeID == 2
<Object>	A KIX object can be specified to which the rule is applied. Keyword: on see also: Overview of KIX objects  Currently only "Ticket" is possible.	Rule "<Name>" on Ticket if TR.TypeID == 2 && TR.StateID == 3 <Commands> End
<RegEx>	A regular expression can be specified to validate user input.	RegExp DFText "[A-Za-z0-9]+\$" "Only numbers and letters possible"
<Value>	Indication of the value to be set.	Set StateID 2

8.2.5.1 Properties of the transaction object

The transaction object can not only access KIX attributes (e.g. TR.TypeID), but also templates and actions. This way, rules can be restricted in such a way that they are only applied when using a certain ticket template or after selecting a ticket/article action (ObjectAction).

This is done by reference to the ID or name of the template/action. Therefore, note the context of use set in the template/action. If a rule is applied to a template/action shared by the Agent Portal and Self Service Portal (usage context Customer + Agent), the rule applies to both portals. However, if this rule is only to apply to the agent portal or to the self-service portal, you can either create 2 separate templates/actions, each with a different context of use, or restrict the context of use (see "Restricting the context of use" below).

Example: Extended properties for transaction object

```
Rule "<Name>" on Ticket if TR.ObjectActionName eq "Ticket Edit" && TR.DynamicFields.contains(DFSelection, 1)
  <Commands>
End
```

```
Rule "<Name>" on Ticket if TR.TemplateID == 3 && TR.DynamicFields.contains(DFSelection, 1)
  <Commands>
End
```

Property	Description
TR.TemplateID	ID of the template used
TR.ObjectActionID	ID of the ObjectAction used
TR.ObjectActionName	Name of the ObjectAction used Note: For comparisons use operator <code>eq</code> instead of <code>==</code> !

✓ Tip

You can find out the ID of an object by moving the mouse pointer over the desired object in an overview. In the footer of your browser you will find the ID as the last/right parameter in the displayed URL (browser-dependent).

8.2.5.2 Restricting the context of use

You can create separate rules for ticket templates and actions that are shared by the agent portal and the self-service portal (usage context customer + agent). It is thus possible to restrict the execution of rules to the respective usage context. The restriction is done by specifying the property of the user:

- **User.isCustomer:** Rule is applied to the Self Service Portal.
- **User.isAgent:** Rule is applied to the agent portal.

In the following example, the ticket template with the ID 3 is shared between the agent portal and the self-service portal. Depending on the portal in which the template is used, different rules apply.

Example: Restriction Context of use

```
#Rule for ticket template 3 is restricted to the usage context 'Customer' (Self
Service Portal):
Rule "UsageContext Customer" on Ticket if User.isCustomer && TR.TemplateID == 3 && TR.
DynamicFields.contains(DFSelectionSingle, 1)
  <Commands 1,2,3>
End

#Rule for ticket template 3 is restricted to the usage context 'Agent' (agent
portal):
Rule "UsageContext Agent" on Ticket if User.isAgent && TR.TemplateID == 3 && TR.Dynam
icFields.contains(DFSelectionSingle, 2)
  <Commands 4,5,6>
End
```

8.2.5.3 Operators

The use of operators makes it possible to link and compare conditions. The following are the most important Perl operators.

Example for AND-Operator

```
# If ticket type "3" (Service Request) AND status 3 (Waiting) are selected on the
ticket, set the priority to "4" (low).
Rule "Type" on Ticket if TR.TypeID == 2 && TR.StateID == 3
  Set PriorityID 4
End
```

Operator		Description
1st Rank	2nd Rank	
&&	and AND	Establishes a logical AND operation between 2 arguments. Rule is executed if both arguments are true .
	or OR	Establishes a logical OR operation between 2 arguments. Rule is executed if one of the two arguments is true .
==	eq	<p>Comparison operator "is equal" Compares 2 values. Returns true if both values are exactly equal.</p> <ul style="list-style-type: none"> Use <code>eq</code> to compare strings. <code>eq</code> performs a comparison of strings. Both arguments must be lexically equal (case sensitive). Example: <code>"3.0" eq "3"</code> returns <code>false</code> Use <code>==</code> to compare numbers. <code>==</code> performs a numerical comparison. Both arguments are converted to a number and compared. Example: <code>"3.0" == "3"</code> returns <code>true</code>
!=	ne	<p>Comparison operator "is not equal" Compares 2 values. Returns true if both values are not equal.</p> <ul style="list-style-type: none"> Use <code>ne</code> to compare strings. Use <code>!=</code> to compare numbers.

Further information on Perl operators can be found, for example, at: <https://beginnersbook.com/2017/02/perl-operators-complete-guide/>

8.2.5.4 Conditions (if)

The keyword `"if"` defines an IF-condition that must be fulfilled in order for the commands specified in the rule to be processed. If no condition is specified, the rule is executed in any case.

Example "if": If the TicketID is equal to 10, then stop

```
Rule "stop here" if DB.TicketID == 10
  Stop
End
```

Keyword/ Command	Description	Example
DB.<Attribute>	Processing takes place on the basis of the database values (see above).	<pre>Rule "<Name>" if DB.TypeID == 1 <Commands> End</pre>
TR.<Attribute>	Processing takes place on the basis of the transaction values (see above).	<pre>Rule "<Name>" if TR.TypeID == 1 <Commands> End</pre>
if !Rule "<Name>" ...	Checks whether a particular rule has NOT been applied.	<pre>Rule "<Name>" if !Rule "<RuleName>" <Commands> End</pre>
if Rule "<Name>" ...	Checks whether a certain rule has been applied.	<pre>Rule "<Name>" if Rule "<RuleName>" <Commands> End</pre>
if <Attribute> .isEmpty(<Field>)	Checks whether a field has no value or an empty list.  [NULL] or [" "] are not empty lists. This is a value, even if it is not defined.	<pre>Rule "<Name>" if TR.DynamicFields.isEmpty(Category) <Commands> End</pre>
if <Attribute>! isEmpty(<Field>)	Checks whether a field has <u>any value</u> or not an empty list.	<pre>Rule "<Name>" if TR.DynamicFields.! isEmpty(Category) <Commands> End</pre>



Keyword/ Command	Description	Example
if <Attribute> .contains (<Field>,<List of values>)	Checks whether any of the specified values are contained in a field. A list of values can be specified.	<pre>Rule "<Name>" if TR.DynamicFields.contains(Category, [1,2,3,23]) <Commands> End</pre>

✓ **Tip**

By combining " if Rule " with " if !Rule " a simple "IF - ELSEIF - ELSE" can be generated:

Example: Simple IF - ELSEIF - ELSE

```
Rule "alpha" on Ticket if TR.ObjectActionName eq "Ticket Edit"
  <Commands>
End

Rule "beta" if Rule "alpha"
  Show DynamicFields.DFText
End

Rule "gamma" if !Rule "alpha"
  Hide DynamicFields.DFText
End
```

The check of conditions can be done on an object ID as well as on the unique name of objects. Accepted are the names for:

- Team/Queue: the **full** team/queue name (e.g. "TeamA::SubteamOne")
- State: the name of the state
- Priority: the name of the priority
- Type: the name of the ticket type
- Owner: the login name of the user
- Responsible: the login name of the user
- Contact: mail address (without real name, only "localpart@domainpart")
- Organization: the customer number

Example: Conditions with Names

```
Rule "Queue" on Ticket if TR.Queue eq "Service Desk"
  Set Priority "1 very high"
  Set State "new"
  Set Type "Incident"
  Set Owner "mamu"
  Set Responsible "mamu"
  Set Contact "max.mustermann@example.org"
  Set Organisation "MY_ORGA"
End
```

8.2.5.5 Statements (then)

Statements are a single command or a list of commands within a rule. They follow an IF-condition and correspond to a "then" statement according to the following statement: IF a certain condition is met, **THEN** execute the following command(s).

Each line contains exactly one statement. The statements are processed in the specified order from top to bottom.

Beispiel: Liste von Anweisungen

```
# If the ticket ID is equal to 10,
# - THEN: Display the Priority field
# - THEN: Set the priority to 3 (Normal)
# - THEN: Stop the processing of the rules and rulesets
Rule "stop here" if DB.TicketID == 10
  Show PriorityID
  Set PriorityID 3
  Stop
End
```

Restrict the range of values:

Keyword/ Command	Description	Example
PossibleValues <Attribute> <List>	Defines the possible values in a field. Only the values specified here can be set.	<pre>Rule "Tickettyp Incident" on Ticket if TR.TypeID == 2 PossibleValues PriorityID 1,2,3 End</pre>

Keyword/ Command	Description	Example
PossibleValuesAdd <Attribute> <List>	<p>Adds the specified values to a field.</p> <p>For example, adds further selection values to a select field if the possible selection values were previously limited with "PossibleValues".</p> <p>Note: If the command is used several times, the previously defined values are <u>not replaced but added</u> (combination of commands)!</p>	<pre>Rule "Tickettyp Incident" on Ticket if TR.TypeID == 2 PossibleValuesAdd PriorityID 4,5,6 End</pre>
PossibleValuesRemove <Attribute> <List>	<p>Removes the specified values from a field.</p> <p>Reduces the possible selection values in a select field, for example, if the possible selection values were previously expanded with "PossibleValuesAdd".</p> <p>Note: If the command is used several times, the previously defined values are <u>not replaced but further reduced</u> (combination of commands).</p>	<pre>Rule "Tickettyp Unclassified" on Ticket if TR.TypeID == 1 PossibleValuesRemove StateID 4 End</pre>

Info

For the commands `PossibleValues`, `PossibleValuesAdd`, `PossibleValuesRemove`, and `Set`, strings must be enclosed in double quotes. Commas may be included within a string. These do not count as separators of the value list.

Restrict the selectability of values:

Note that some of the following commands are mutually exclusive. For example, the command "Hide" reverses the command "Show". This means that if a field is displayed with "Show" and then a "Hide" is applied to the field, this will end the display of the field. The last command wins.



Keyword/Command	Description	Example
MinSelectable <Attribute><Count>	Defines the minimum number of values that the user must select, E.g. at least 2 of n values.	<pre>Rule ... MinSelectable DynamicFields.DFSelection 2 End</pre>
MaxSelectable <Attribute><Count>	Defines the maximum number of values that the user may select, e.g. maximum 2 of n values; after that no further selection is possible ($x \leq n$)	<pre>Rule ... MaxSelectable DynamicFields.DFSelection 2 End</pre>

Keyword/Command	Description	Example
<p>Set <Attribute> "<Value>" "<List>" "<Array>"</p>	<p>Sets a fixed value in a field. Optionally allows a list of values or an array. Strings must be placed in double inverted commas. The combination of strings and values is possible - if supported by the attribute.</p> <p>The "Set" parameter supports object IDs as well as object names. Accepted for:</p> <ul style="list-style-type: none"> • Team/Queue: the full team/queue name (e.g. "TeamA::SubteamOne"). • State: the name of the state • Priority: the name of the priority • Type: the name of the ticket type • Owner: the login name of the user • Responsible: the login name of the user • Contact: mail address (without real name, only "localpart@domainpart") • Organization: the customer number <p>If a parameter is specified multiple times in a rule (e.g. as Queue or QueueID), the last entry wins.</p>	<pre>Rule ... Set PriorityID 5 Set OrganisationID 2,5,10 Set DynamicFields.DFText "foo", "foo bar", "foo, bar & 2 foobar" Set DynamicFields.DFDateTime "2022-09-01 12:34:56" End</pre>

Keyword/Command	Description	Example
ReadOnly <Attribute>	Field is set to "read only". Counterpart and reversal of the "Writable" command Tip: In conjunction with Set, a value can be set permanently in a field.	<pre>Rule ... Set PriorityID 5 ReadOnly PriorityID End</pre>
Writable <Attribute>	Field is set to "writable" Counterpart and reversal of the "ReadOnly" command.	<pre>Rule ... Writable DynamicFields.DFText End</pre>
Clear <Attribute>	A value set in the field is removed.	<pre>Rule ... Clear DynamicFields.DFSelection End</pre>
Show <Attribute>	Field is displayed/not hidden Counterpart and reversal of the "Hide" command .	<pre>Rule ... Show PriorityID End</pre>
Hide <Attribute>	Field is hidden. Counterpart and reversal of the "Show" command. Note: Hidden fields are still active and are taken into account when saving. That is: <ul style="list-style-type: none"> • a set value is transferred in the background. • a hidden mandatory field also expects a value. Without a value, the form is not saved. <p>To ensure that a field is not taken into account when saving, it must be deactivated with "Disable".</p>	<pre>Rule ... Hide PriorityID End</pre>

Keyword/Command	Description	Example
Enable <Attribute>	<p>Sets a field to "active" so that it can be used normally according to its configuration. Counterpart and reverse of the "Disable" command.</p> <p>If the field is hidden, a value set (in the background) is transmitted.</p> <p>Requires additionally the command "Show" to show again a field hidden with "Hide" or deactivated with "Disable".</p> <p>⚠ Always execute the "Enable" command at the end of a rule. Otherwise, inconsistent states may occur. So first manipulate the field, then activate it.</p>	<pre>Rule ... Show DynamicFields.DFText CountMax DynamicFields.DFText 3 Enable DynamicFields.DFText End</pre>
Disable <Attribute>	<p>Field is deactivated and hidden in the frontend (implicit "Hide"). (Value is NOT transmitted in the background). Counterpart and reversal of the "Enable" command.</p>	<pre>Rule ... Disable DynamicFields.DFText End</pre>
Optional <Attribute>	<p>Field is optional, i.e. not mandatory Counterpart and reverse of the "Required" command.</p>	<pre>Rule ... Optional DynamicFields.DFTextArea End</pre>
Required <Attribute>	<p>Field is mandatory Counterpart and reverse of the "Optional" command.</p>	<pre>Rule ... Required DynamicFields.DFTextArea End</pre>

Influencing the field properties:

Keyword/Command	Description	Example
<p>CountMax <Attribute> <Count></p>	<p>Sets a (different) value for the property "Number (min)" of a dynamic field.</p> <p>Note: CountMax only applies to all <u>non</u>-multiselect fields. To limit the selection in multiselect fields (e.g. Selection or Reference Dynamic Fields), use the MaxSelectable or MinSelectable command.</p>	<pre>Rule ... CountMax DynamicFields.DFTText 3 End</pre>
<p>RegExp <Attribute> "<RegEx>" "<ErrorMessage>"</p>	<p>Regular expression for validation of user input in a field and text for error message in case of invalid input, e.g. conditions for entering a valid IP4 and IP6 address</p>	<pre>Rule ... Show DFTText RegExp DFTText "((^\s*((([0-9] [1-9][0-9] 1[0-9]{2} 2[0-4][0-9] 25[0-5])\.[0-9]{3}([0-9] [1-9][0-9] 1[0-9]{2} 2[0-4][0-9] 25[0-5]))\s*\$) (^\s*((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4} :)) (([0-9A-Fa-f]{1,4}:){6}(:[0-9A-Fa-f]{1,4} ((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) ((([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2}) :(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) ((([0-9A-Fa-f]{1,4}:){4}(((:[0-9A-Fa-f]{1,4}){1,3}) ((:[0-9A-Fa-f]{1,4})?:((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) ((([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]{1,4}){2}) ((:[0-9A-Fa-f]{1,4})?:((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) ((([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,2}) ((:[0-9A-Fa-f]{1,4})?:((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) ((([0-9A-Fa-f]{1,4}:){1}((:[0-9A-Fa-f]{1,4}) ((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3}) :)) (: ((25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d 1[0-9]?\d))){3} :))\s*\$))\s*\$)"</pre>

Keyword/Command	Description	Example
		<pre>{3}(((:[0-9A-Fa-f]{1,4}){1,4}) (::[0-9A-Fa-f]{1,4}){0,2}: ((25[0-5] 2[0-4]\d 1\d\d [1-9]? \d)(\.(25[0-5] 2[0-4]\d 1\d\d [1-9]?\d)){3})) :)) ((:[0-9A-Fa- f]{1,4}:){2}(((:[0-9A-Fa-f] {1,4}){1,5}) (::[0-9A-Fa-f] {1,4}){0,3}:((25[0-5] 2[0-4]\d 1\d\d [1-9]?\d)(\.(25[0-5] 2[0-4]\d 1\d\d [1-9]?\d) {3})) :)) ((:[0-9A-Fa-f]{1,4}:) {1}(((:[0-9A-Fa-f]{1,4}){1,6}) (::[0-9A-Fa-f]{1,4}){0,4}: ((25[0-5] 2[0-4]\d 1\d\d [1-9]? \d)(\.(25[0-5] 2[0-4]\d 1\d\d [1-9]?\d)){3})) :)) (:(((:[0-9A-Fa-f]{1,4}){1,7}) ((: [0-9A-Fa-f]{1,4}){0,5}: ((25[0-5] 2[0-4]\d 1\d\d [1-9]? \d)(\.(25[0-5] 2[0-4]\d 1\d\d [1-9]?\d)){3})) :)))?(%.+)? \s*\$))" "Gib eine gültige IP Adresse ein" End</pre>
<p>InputOrder <AttributeList></p>	<p>Defines the order of the fields in the form, e.g. 1st field: Priority 2nd field: Status 3rd field: Dynamic field "DFText</p> <p>The named fields are placed at the top of the form in the specified order. All other form fields are placed below, according to the order defined in the template.</p>	<pre>Rule ... InputOrder PriorityID, StateID, DynamicFields.DFText End</pre>



General:

Keyword/Command	Description	Example
Stop	<p>The processing of the rules is stopped completely. Subsequent rules and rule sets are not taken into account.</p> <p>With the next user action, the processing of the rules starts again with the first rule set.</p>	<pre>Rule ... Stop End</pre>

8.2.6 Functions

8.2.6.1 DynamicFields.contains

The function `"DynamicFields.contains"` can be used to check whether a certain value is set in a dynamic field. The function searches the list of dynamic fields for the name of the dynamic field and checks whether the specified value is contained in the values of the dynamic field. If yes, the function returns 1. If no, the function returns 0. Only 1 value can be queried at a time. If several values of the same field are queried, further rules or conditions must be created.

Syntax:	TR.DynamicFields.contains(<Name of the DynamicField>, <Value of the DynamicField>)
	DB.DynamicFields.contains(<Name of the DynamicField>, <Value of the DynamicField>)

Example: DynamicFields.contains

```
Rule "Alpha" on Ticket if TR.DynamicFields.contains(DFSelectionSingle, 0)
  MaxSelectable DynamicFields.DFSelectionMulti 1
  PossibleValues DynamicFields.DFSelectionMulti 0,1,2
End

Rule "Beta" on Ticket if TR.DynamicFields.contains(DFSelectionSingle, 1)
  MaxSelectable DynamicFields.DFSelectionMulti 1
  PossibleValues DynamicFields.DFSelectionMulti 3,4,5
End

Rule "Gamma" on Ticket if TR.DynamicFields.contains(DFSelectionSingle, 2)
  MaxSelectable DynamicFields.DFSelectionMulti 1
  PossibleValues DynamicFields.DFSelectionMulti 6,7,8,9
End
```

8.3 Template groups

In the *Ticket > Template Groups* menu, you can create categories to structure [templates](#) (see page 191) thematically. A template is assigned to a template group when a template is created.

By assigning a template group to a superordinate template group, you can create a tree structure from template groups. The sorting is done alphanumerically. You can create as many levels as you want. You can create a service catalog for individual user groups (agents and customers).

Template groups can be duplicated (see below). If necessary, the directly and indirectly subordinate template groups can also be included.

Agents can find the template tree including the templates assigned to the groups in the left sidebar. There the desired template can be selected when creating a new ticket.

As a result of the grouping, the templates provided in the Self Service Portal are also available in an easy-to-find structure.

Only valid template groups to which a template has been assigned are displayed. A user only sees the templates for which he has authorizations.

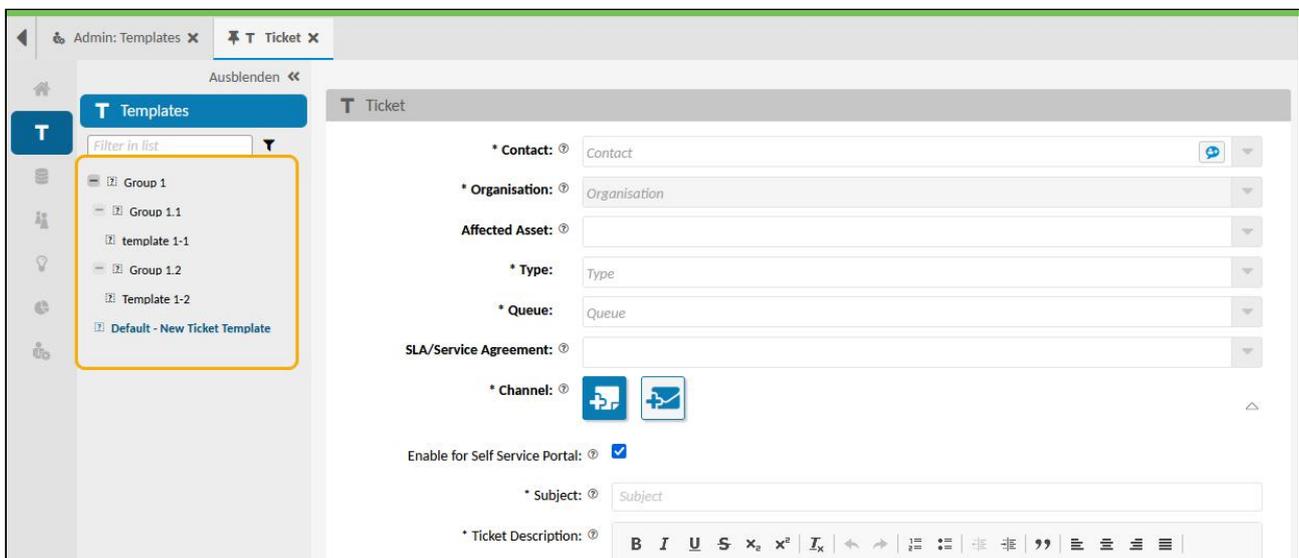


Fig: The template groups including templates in the sidebar when creating a new ticket

To create a new template group, navigate to *Ticket > Template Groups* and click on "New Group". Fill in the dialog that opens (see table below) and save your entries.

To edit a template, click on a template group and change the information as required.

To delete a template group, select the template group with a check mark and click "Delete". A template group can only be deleted if no template is assigned to it. If necessary, assign the relevant templates to a different template group beforehand.

The form dialog has the following input fields:

 <p>Incident Report ¹  Use this template if you encounter an incident or some ² station of a usually working service or function</p> <p><i>Fig.: Template in the SSP</i></p>	Field	Description
	¹ Name	Name of the template group
	² Icon	Optional icon to identify the template group. The icon is placed in front of the template name in the agent portal.  A changed icon is only displayed after logging in and updating the browser cache.
	Rank	The rank determines the sort order in the Self Service Portal: <ol style="list-style-type: none"> 1. Template groups based on rank 2. Unranked template groups sorted by name Sorting in the agent portal is alphanumeric.
	Parent Group	Selection of a superordinate template group in order to create a structure tree of template groups (optional).
	³ Comment	Optional comment to describe the template group in more detail.
	Validity	<ul style="list-style-type: none"> • valid: Templates in this group are available to users according to their permissions. • (temporarily) invalid: If a template group is invalid, subordinate structures (subgroups and templates) are not offered for selection. The templates of this group are therefore not available, even if the template itself is valid.

8.3.1 Duplicating groups of templates

You can duplicate template groups, e.g. to create new service catalogs based on existing ones. You can specify whether the subordinate template groups should also be duplicated.

If templates are assigned to the template group to be duplicated, the templates are also duplicated.

To duplicate, select the relevant template group with a tick. You can then only click on "Duplicate" in the header of the overview. The button is only active when a template group is checked. A dialog opens with the following parameters:

Field	Description
Name	Display name of the template group in the agent portal and in the Self Service Portal
Copy Recursively	If checked, the copies of all directly and indirectly subordinate template groups are also created. If not checked, only the selected template group - without sub-elements - is duplicated.
Prefix for Duplicates	To identify the duplicates, they are given a prefix. The default is: "Copy of". You can set a different prefix.
Icon	You can assign an icon to the duplicate (optional)
Rank	The rank determines the sorting order in the Self Service Portal (see table above)
Parent group	You can specify which template group the duplicate (s) should be subordinated to. If nothing is defined, the duplicates are created on the same hierarchical level as their originals.
Comment	Optional comment



Field	Description
Validity	<ul style="list-style-type: none">• valid: Templates in this group are available to users according to their permissions.• (temporarily) invalid: If a template group is invalid, subordinate structures (subgroups and templates) are not offered for selection. The templates of this group are therefore not available, even if the template itself is valid.

8.4 Templates

You can configure templates for the creation of new tickets in KIX Pro, which are independent of the specifications in the SysConfig.

Templates are

application-dependent templates with which the ticket creation masks in the agent portal and in the self-service portal can be adapted to the respective context at the time of need. They are therefore also called ticket templates.

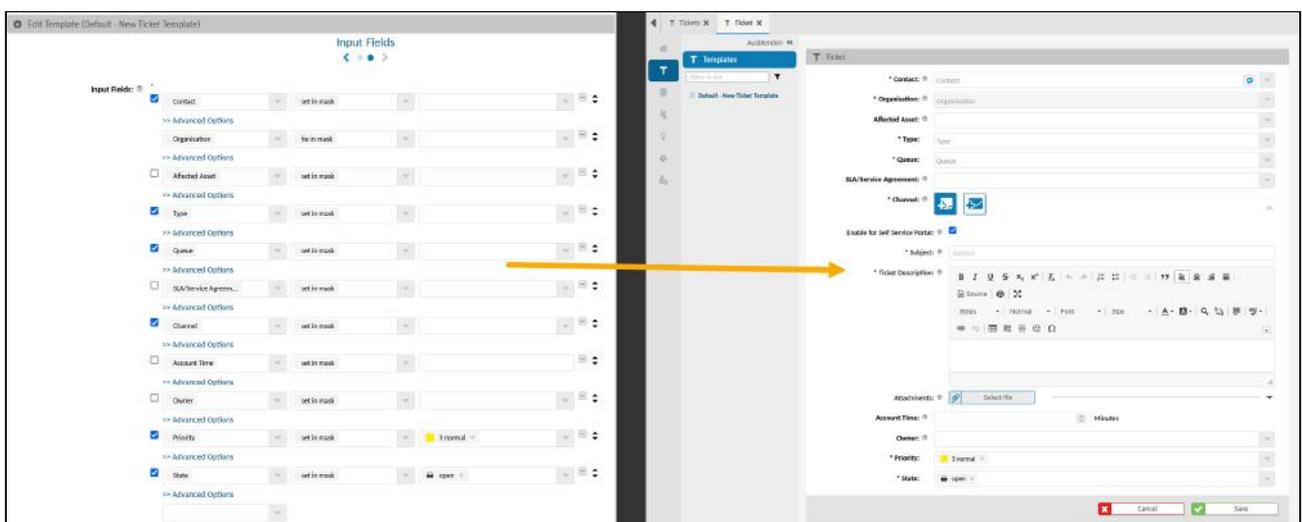
Use cases for templates are, for example, special ticket creation masks

- for hiring new employees
- for quick entries such as "Printer paper is missing"
- Vacation requests
- Permits
- for checklists
- and many more

It is thus possible to make entries adapted to the message type or to create special ticket templates for a selected group of customers.

Contents on this page:

- [Configure a Template](#) (see page 193)
 - [Notes on field selection](#) (see page 203)
 - [Notes on field options](#) (see page 204)
- [The initial standard template "Default - Ticket New Template"](#) (see page 207)
 - [Initial configuration of the standard template](#) (see page 207)



The administration of the ticket templates can be found under *Workflow > Templates*. The overview contains all ticket templates available in the system. The column "Usage Context" shows in which portal the template can be selected:

- **Customer:** The template is available in the Self Service Portal (customer portal) (requires the "Customer" role on the template).
- **Agent:** The template is available in the agent portal.

The "Template groups" column indicates which [template group](#) (see page 187) the template is assigned to.

You can **duplicate** templates to create new templates based on existing templates. To do this, click on "Duplicate" in the overview. A duplicate of the selected template is opened. The configuration of the duplicate corresponds to its source. You can change the configuration of the duplicate as needed (see below - Configure a template). The name of the duplicate is "Copy of [name of template]" by default. You can keep this name or name the duplicate differently.

The templates are available for agents to choose from in the agent portal when creating a new ticket in the left sidebar. The templates are available for selection in the Self Service Portal above the editor. Based on the selected template, the ticket creation mask is changed according to the template configuration. Only the fields defined in the template are displayed, regardless of the specifications of the SysConfig key.

By default, the user's standard ticket template is opened. Each user can define the standard ticket template himself in his personal settings under "My ticket template". He can only choose from the valid templates to which he has access according to his role authorization.

In the Self Service Portal, ticket templates can also be opened via **URL direct call**, e.g.: <https://kix-domain-ssp.kix.cloud/tickets/new?templateId=79>. This enables the template to be called up even from a third-party system. The template is loaded directly after the user has successfully logged in. If a template that does not exist or is not authorised by the user is called up, the "New ticket" dialogue will be opened. The template to be loaded is determined by the templateID (URL parameter "templateId"). The ID of a template can be found in the footer of your browser if you hold the mouse pointer over the name of a template in the template overview in the Admin module.

Hints

- Templates do not currently check dependencies. If, for example, a customer is set via the template that does not match the contact, the customer set in the template is overwritten.
- Templates only set values in existing fields or hide existing fields in the ticket.
- In the Self Service Portal:
 - The selection or entry of the ticket contact is always fixed and cannot be made by the customer user.
 - It is not possible to select a waiting queue in the Self Service Portal, as no waiting time can be set.
- In order for templates to be available in the Self Service Portal, they must be assigned the "Customer" role.

- Manually changed standard templates are not adapted in the update in order not to compromise the changed configuration. Any input fields added with the update must be added manually.

8.4.1 Configure a Template

To configure or create a template, navigate to *Workflow > Templates* . A table is opened in the content area, which lists all the ticket templates created in the system. Click on "New template" in the table or select an existing template with one click. A assistant will open in which you can configure the template step by step. Click either the blue arrow buttons or dots to switch between the individual steps. When you are finished, please click on "Save" to apply your changes.

Template information (step 1):

⚙ Edit Template (Onboarding)

Template Information

< ● ● ● >

Template Information

* Usage Context: 🔍 Agent ⌵

* Name:

Icon: 📁 Select image file T

Rank: ⌵

Template Group: 🔍 Intern ⌵

Comment:

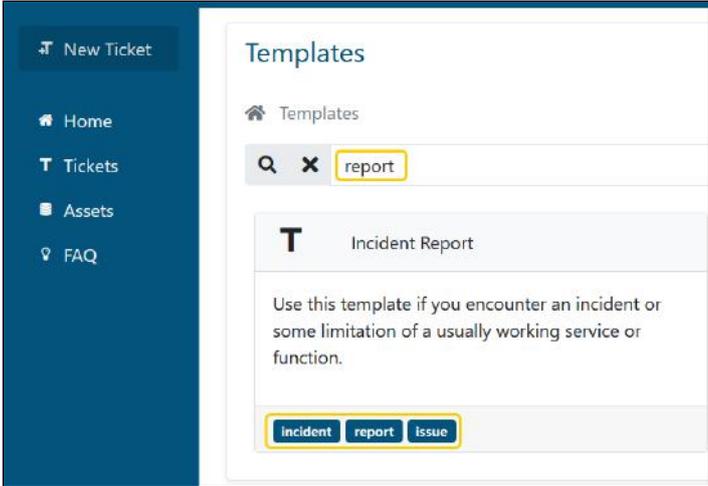
Keywords:

* Validity: valid ⌵

Required Role(s)

Roles: 👤 Ticket Agent ⌵

Field	Description
Usage Context	<p>The context of use defines in which portal the template is available for selection (multiple selection possible).</p> <ul style="list-style-type: none"> • Agent: The template can be selected in the agent portal. • Customer: The template can be selected in the self-service portal ("Customer" role required). • System: The template cannot be used by users of the agent or self-service portal. It is used to create tickets that are generated exclusively by the backend. <p>The template can be used by macro actions (e.g. "Create Ticket from System Template") and other automatisms, e.g. by the add-on "KIX Maintenance Plan" for the automated creation of tickets for maintenance tasks.</p> <p>Templates with the exclusive usage context "System" are not available in the Agent Portal and the Self Service Portal.</p> <p>System templates support the use of KIX placeholders to resolve the information of reference objects when they are specified.</p>
Name	<p>Enter a meaningful name for the template. Agents and customers can later select the template under this name.</p>
Icon	<p>Optionally, you can choose an icon to symbolically identify the template.</p> <p> A changed icon is only displayed after logging in and updating the browser cache.</p>
Rank	<p>The rank determines the sort order in the Self Service Portal:</p> <ol style="list-style-type: none"> 1. Templates based on rank 2. Unranked templates sorted by name <p>Sorting is alphanumeric in the agent portal.</p>
Behaviour	<p>Set the behaviour of the template here:</p> <ul style="list-style-type: none"> • New (Default): Select this option for all templates that are not used by the add-on "KIX Maintenance Plan Wartungsplan". Also possible for system templates that are used by the macro action "Ticket Create from System Template", for example. • Maintenance: This option is only available after purchasing the add-on "KIX Maintenance Plan Wartungsplan". Select this option if the template is to be used by the add-on to generate the maintenance tickets. <p>In the configuration of maintenance plans, only templates with the usage context "System" AND the behaviour "Maintenance" can be selected.</p>

Field	Description
Template Group	<p>You can assign the template to a template group and thus structure the templates. Only templates from valid template groups are available, regardless of whether the template itself is valid.</p>
Comment	<p>Optionally, write down a detailed description of the template. The comment is displayed in the label of the template in the Self Service Portal.</p>
Keywords	<p>You can keyword the template (optional). Separate the individual keywords with a space.</p> <div data-bbox="338 725 743 801" style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p>Keywords: incident report issue</p> </div> <p>The templates in the Self Service Portal are marked with these keywords. Users of the Self Service Portal can search for templates using the keywords.</p> <div data-bbox="338 920 1046 1406" style="border: 1px solid gray; padding: 10px; margin: 10px 0;">  <p>The screenshot shows the 'Templates' page in the Self Service Portal. A search bar at the top contains the word 'report'. Below the search bar, a template titled 'Incident Report' is displayed. At the bottom of the template card, three keyword tags are visible: 'incident', 'report', and 'issue'.</p> </div>
Validity	<p>Set the validity to "valid" so that the template can be selected. Templates set to "invalid" are not available for selection in the portals. The template used is retained in existing tickets.</p>

Field	Description
Roles	<p>Select one or more roles from which the template can be used. The template is only available to members of the roles selected here. To deselect roles, click on the small cross next to the role description.</p> <p>The "Customer" role must be selected so that a template is available in the Self Service Portal.</p> <div data-bbox="336 651 1422 1070" style="border: 1px solid red; padding: 10px;"><p> Important!</p><p>Never assign authorization to the ticket template to roles that (explicitly or implicitly) have more than read rights to templates (e.g. superuser, admin, etc.)! Only use the roles to extend permissions to the use of the ticket template! Background: A "READ" is explicitly set for templates for the roles selected here. If you select the role "Superuser", for example, the UPDATE authorization for templates will be <u>overwritten</u> so that the user can no longer edit the template. Should this still happen, there is the option of manually changing the rights in the user administration.</p></div>

Input fields (step 2):

Specify here which input and selection fields should be included in the template. When selecting a template, only the fields defined here are available in the ticket form.

You can also specify which values should be set in the ticket. As soon as the template is selected by the user, the values in the ticket form fields are replaced by the values defined in the template. Any standard values will be overwritten.

The form fields in the ticket form are displayed in the order specified here. You can change the order by dragging and dropping a field to the desired position using the double arrows on the right edge . To remove a form field, click the minus button . You can add further form fields by selecting another input field in the drop-down field.

Input Fields:

Input Fields

1 2 3

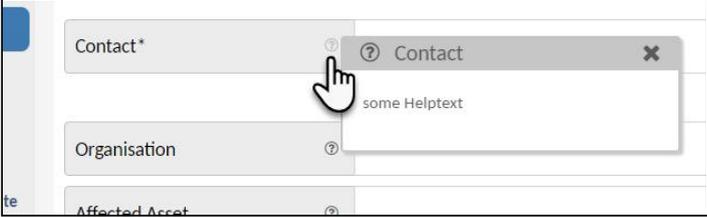
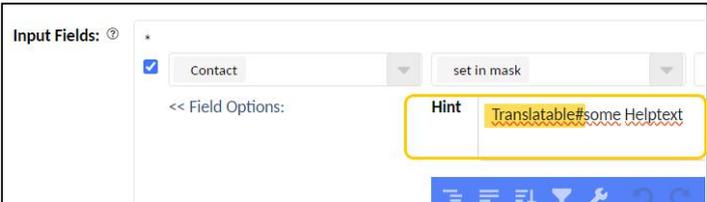
4

```
1 [
2   {
3     "option": "OBJECT",
4     "value": "SLA"
5   },
6   {
7     "option": "SHOW_INVALID",
8     "value": 0
9   }
10 ]
11
```

Column		Description
1	Field selection	<p>Select a field on which field the action should be applied. You can choose from all ticket and article attributes as well as the dynamic fields. Please also note the notes on field selection (see page 203) below.</p> <p><input checked="" type="checkbox"/> A check mark in front of the input field declares the field as a mandatory field.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Infos</p> <ul style="list-style-type: none"> • Channel-specific fields (To, CC, BCC, Subject, Article Content, Attachments) additionally require the "Channel" field in the template definition. Only when a channel has been selected when using the template are the channel-specific fields displayed and filled with default values. If the channel is set hidden or fix in mask, the channel must already be selected in the template definition. • Take permissions into account: If fields are configured in a ticket action to which the user (agent or customer) does not have permission, their use will fail. • If a ticket receives the status "Waiting for [...]", a dynamic time in the future is displayed as a default for the target waiting time. This pre-assignment is initially determined from the current point in time plus the value stored in the SysConfig. You can adjust the pre-assignment of the target waiting time by changing the key <code>"Ticket::Frontend::PendingDiffTime"</code> in the menu <i>System > SysConfig</i>. <p>Please note: A target waiting time must be specified in the ticket so that it can be saved without manually changing the time.</p> </div>

Column	Description
<p>2</p>	<p>Properties</p> <p>Define how the field is displayed when using the template and with which values the ticket is initialised:</p> <ul style="list-style-type: none"> • Set in mask: The field is displayed in the ticket form and initialised with the value already set on the ticket. This value can be overwritten by a default value in the dialogue. A value set in the template can be changed in the form (unless configured otherwise under Field Options). • Fix in mask: The field and its value are displayed in the ticket mask. The value cannot be edited. E.g. to display a non-editable value in the ticket. • Set hidden: The field and its value are not displayed in the ticket screen. For example, to specifically exclude fields from editing or to assign invisible values to the ticket. <ul style="list-style-type: none"> • Important: Item text and subject must not be set to "in background" for plausibility reasons. The aim is that all communication should be visible to the user before sending. • Clear in mask: shows the field in the dialogue, but initialises it with an empty value. This enables the resetting of values. In conjunction with mandatory entries, a conscious decision by the user is forced. <p>Relative time:</p> <p>Date/time fields can be provided with a relative initialisation. The following options are possible:</p> <ul style="list-style-type: none"> • Set in form (relative time) • Set as fixed value (relative time) • Set in background (relative time) <p>Behaviour of the dynamic field as described above, but information as relative time value (e.g. 10 minutes). Available for dynamic fields of the types Date and DateTime. Can be used, for example, for waiting times with relative time differences or tickets with relative times for plan start/plan end. Thus, no waiting status has to be stored in the template itself.</p> <p>Based on the time of use, the specified time value is added to the current time and set as a date (each time the template is called up again).</p> <p>The use of KIX placeholders is supported so that a number specified in a dynamic field (field type Text) can be read out and used as a time value.</p> <p>The defined waiting times have a higher priority than the setting in the SysConfig key "<i>Ticket::Frontend::PendingDiffTime</i>".</p>

Column		Description
<p style="text-align: center;">3</p>	<p>Define values</p>	<p>Optionally, specify a value with which the field will default when the template is used. Leave the value blank if you do not want to give the field a value.</p> <p>The choices in dropdowns are in direct context to your field selection in column 1.</p> <p>Notes on using placeholders:</p> <ul style="list-style-type: none"> • You can specify placeholder variables to put variable, ticket-specific values in the field. Enter the placeholder manually in the selection field and confirm with ENTER. Placeholders for dynamic fields are specified according to the following scheme: <KIX_TICKET_DynamicField_DynamicFieldName>. • When editing tickets, the substitution of placeholder variables refers to the source ticket; for new tickets, the fields are empty (for ticket placeholders). • Fields that reference other objects such as contacts, agents or priority expect an ID as value. This means that when using KIX placeholders, the ID must be specified (e.g. KIX_CURRENT_UserID>). For dynamic fields, the value specified under "Key". • In order for placeholders in ticket forms to reference correctly, it is important to distinguish between user placeholders (KIX_xxx_UserID) and contact placeholders (KIX_xxx_ContactID) and to use them correctly. For example, the contact field needs a placeholder that references CONTACTS - even if the contact can be an agent and thus a user. This means, for example, for placeholders in the template "Default - New Ticket Dialog": <ul style="list-style-type: none"> • Placeholder for the contact in the ticket: <ul style="list-style-type: none"> • correct: <KIX_CURRENT_ContactID> • wrong: <KIX_CURRENT_UserID> • Placeholder for the currently logged in user as agent or responsible: <ul style="list-style-type: none"> • correct: <KIX_CURRENT_UserID> • Fields that reference other objects such as contacts, agents or priority expect an ID as value. This means that when using KIX placeholders, the ID must be specified (e.g. KIX_CURRENT_UserID>). For dynamic fields, the value specified under "Key".

Column	Description
<p>4</p> <p>Advanced Options</p>	<p>"Hint" field: You can store a customised help text for the field. This allows you, for example, to inform the agents which data a dynamic field expects.</p> <p>If a help text is specified, a question mark symbol is displayed in the field. The help text is displayed after clicking on the question mark. The help text has the same line breaks as the stored text.</p>  <p><i>Fig.: Customised help text in the "Contact" field</i></p> <p>The help text is saved in the field configuration of the respective action so that a different help text can be noted for the field for each action.</p> <p>The initial fields of a template already contain localised help texts, which are provided via the weblate. If required, you can replace these with your own customised help texts.</p> <p>You can maintain the translation of your help texts yourself in the <i>KIX > Internationalisation > Translation</i> menu (see Translations). Mark the help texts to be localised with the preceding <code>Translatable#</code> :</p>  <p><i>Fig.: Texts to be translated are labelled with a preceding "Translatable#"</i></p> <p>Note: Fields added by workflow rule are currently not considered.</p> <hr/> <p>Editor: You can specify an array of further configuration options for the field in JSON format, e.g. specific loading options, the object type to be loaded, etc.</p>

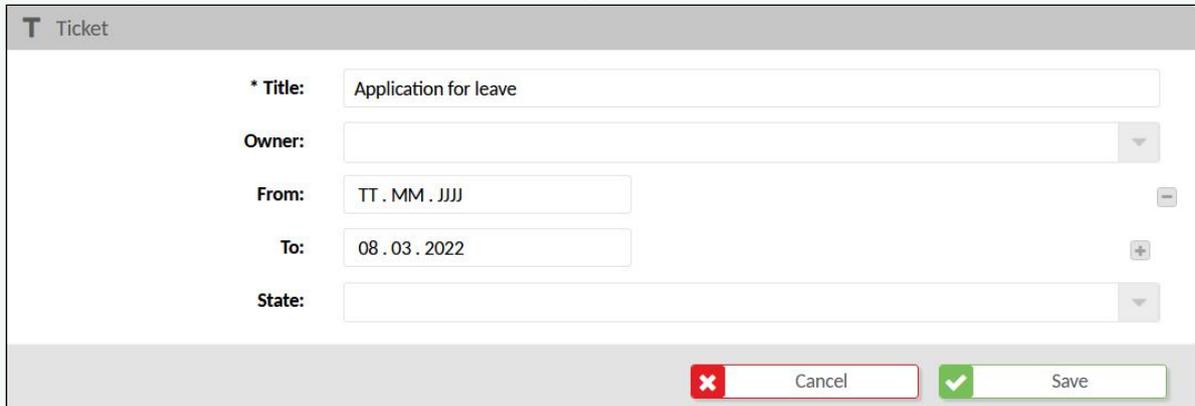
Column	Description
	<p>Depending on the selected field, various options are supported. (see below: Notes on field options). Some configuration options are set automatically so that the fields function correctly, e.g. for the dynamic field "Affected asset".</p> <ul style="list-style-type: none"> • Option: specification of the attribute • Value: value of the attribute <p>Put the field options in square brackets: [...]. Put each option block in curly brackets {...} and separate multiple options with commas.</p> <pre data-bbox="555 788 1423 1164"> [{ "option": "SHOW_NEW_CONTACT", "value": 1 }, { "option": "SHOW_INVALID", "value": 0 }] </pre> <p>With field options, for example, the users available for selection in the field can be limited to certain roles.</p> <pre data-bbox="555 1290 1423 1541"> [{ "option": "Role", "value": "TicketAgent, Customer" }] </pre> <div data-bbox="555 1550 1423 1729" style="border: 1px solid green; padding: 5px;"> <p> Tip</p> <p>Use the configuration of the "Default - New Ticket Template" template as an example.</p> </div> <div data-bbox="555 1738 1423 1917" style="border: 1px solid orange; padding: 5px;"> <p> Note</p> <p>KIX placeholders used in the field options are not resolved by the Self Service Portal.</p> </div>

8.4.1.1 Notes on field selection

- In the "agent" context of use, contact, customer and title are automatically always mandatory fields in the configuration, as these fields are required to create tickets.
- Contact and customer cannot be configured in the context of use "Customer". These fields are not available for selection.
- If no contact is configured in the template, the user who created it is entered as a contact.
- If no customer is configured in the template, the PrimaryOrg of the contact is entered as the customer.
- If title AND article subject are not configured or specified in the template, this is automatically generated from "<template title> <YYYY-MM-DD hh: mm of current time>".
- If the template configuration contains the channel, the selected channel value determines the input fields available in the form:
 - If the channel is set with a value (e.g. email or note), the channel-specific input fields such as subject, ticket description, attachment or from and CC are available in the ticket form.
 - If the channel has no value or is set in the background, the channel-specific input fields are not available (similar to channel selection "without article"; see also tip below).
However, the title must be specified (fixed, hidden or visible)!
- If no channel is selected in the template or is set in the background,
 - the channel-specific input fields are not available in the form
 - the article-specific input fields "Subject", "Body" and "Attachments" are not available in the template configuration
- The following SysConfig keys are used as fallback for mandatory ticket fields that are not specified in the template:
 - **Type:** *Ticket::Type::Default*
 - **Team:** *PostmasterDefaultTeam*
 - **Channel:** *API::Operation::V1::ArticleCreate###Channel*
 - **Priority:** *PostmasterDefaultPriority*
 - **Status:** *PostmasterDefaultState*
- Dynamic field "Affected Services" in the Self Service Portal: There is no check of permissions. If services are set "as a fixed value" or "in the background" in a template, then the service is set even if the Self Service Portal user does not have the corresponding permissions.
- If the template does not contain a title, the title is set automatically, consisting of: Name of the template used + date + time (e.g.: "Holiday request 11.11.2022 14:15")

✓ **Tip**

If the template configuration does not contain a channel specification or if the specifications are set in the background, no ticket/item-specific input fields are available. This allows tickets to be created without subject, item content, attachment, etc., e.g. holiday request.



8.4.1.2 Notes on field options

Depending on the input field, different attributes are supported in the Options field. Some attributes are set automatically in the background and cannot be configured. (e.g. Organisation is automatically set according to the selected contact.) These are not listed below.

Specific attributes for **dropdown fields** (status, priority, processor, responsible person, team, type, organization and contact):

Attribute	Description	Example
<p>AUTOCOMPLETE</p>	<p>Controls the autocomplete behaviour for text input in selection fields (e.g. in reference fields such as "Related tickets", "Contact", etc.)</p> <ul style="list-style-type: none"> <code>charCount</code> : 1 Defines how many characters must be entered for the search to start (default: 3). <code>delay</code> : Waiting time in milliseconds until the search starts after the last character has been entered. The value can be changed if the search starts too early or too late. <code>limit</code> : Display limit in the selection field. The limit can be increased if necessary if too few or no entries are loaded due to restricted user rights. <code>noResultsObjectName</code> : 2 Object name to be displayed if no objects were found. <p>This designation is automatically translated by prefixing it with "Translatable#" if you have stored a corresponding pattern in the "Internationalisation > Translation" menu. If nothing is specified, the designation "Objects" is used.</p> 	<pre>{ "option": "AUTOCOMPLETE", "value": { "charCount": 3, "delay": 250, "limit": 25, "noResultsObjectName": "Translatable#Contacts" } }</pre>
<p>AUTOCOMPLETE_PRELOAD_PATTERN</p>	<p>A search is made within the preloaded values using the character string entered under "value" guided.</p>	<pre>{ "option": "AUTOCOMPLETE_PRELOAD_PATTERN", "value": "*Meier*" }</pre>

Attribute	Description	Example
MULTISELECT	Enable selection of multiple items value: 1 0 or true false	<pre>{ "option": "MULTISELECT", "value": 1 }</pre>
FREETEXT	Allow entry of free text (e.g. for email fields) value: 1 0 or true false	<pre>{ "option": "FREETEXT", "value": 1 }</pre>
COUNT_MIN	Minimum number of items that must be selected. With values > 1, the field becomes a mandatory field.	<pre>{ "option": "COUNT_MIN", "value": 1 }</pre>
COUNT_MAX	Maximum number of items that can be chosen (e.g. maximum number of affected assets)	<pre>{ "option": "COUNT_MAX", "value": 10 }</pre>
SHOW_NEW_CONTACT	Specific option for the "Contact" field. Decides whether to display the New Contact action. value: 1 0 or true false	<pre>{ "option": "SHOW_NEW_CONTACT", "value": 1 }</pre>
USE_OBJECT_SERVICE	Specific option for the "Team" field. Makes teams appear as a tree. value: 1 0 or true false	<pre>{ "option": "USE_OBJECT_SERVICE", "value": 1 }</pre>

8.4.2 The initial standard template "Default - Ticket New Template"

In KIX Pro, the "New Ticket" dialogue is based on the standard template "Default - Ticket New Template". Using this template, you can customise the "New Ticket" dialogue and easily integrate your own dynamic fields (if necessary, refresh the browser window with F5 afterwards). The integration of dynamic fields into other ticket interfaces like ticket details or ticket dashboard is done via SysConfig (see KIX start manual).

The configuration of the default template is initially based on the configuration of the SysConfig key "*ticket-new-form-group-data*". Changes to the default template do not affect the configuration in the SysConfig key. If the default template is set to invalid or cannot be loaded, the configuration of the SysConfig key serves as fallback.

Note: The "Edit ticket" dialogue is a ticket action. Changes to this dialogue are therefore made in the menu *Workflow > Actions > Action "Ticket Edit"*.

Attention

It is currently not possible to reset the configuration of the "Default - Ticket New Template" template to the delivery status. You should therefore change the template configuration carefully! If necessary, you will find the output configuration at the end of the chapter.

8.4.2.1 Initial configuration of the standard template

Template information

Field	Value
Name	Default - New Ticket Template
Context of use	Agent
Comment	Default new ticket template for agents with role Ticket Agent.
Validity	valid
Roles	Ticket Agent

Input fields



Field	Properties	Value	Field Options
<input checked="" type="checkbox"/> Contact	Set in the form	[empty]	<pre>[{ "option": "SHOW_NEW_CONTACT", "value": 1 }, { "option": "SHOW_INVALID", "value": 0 }]</pre>
Organisation	Set as a fixed value	[empty]	[empty]
<input type="checkbox"/> Affected Asset	Set in the form	[empty]	<pre>[{ "option": "FIELD_NAME", "value": "AffectedAsset" }]</pre>



Field	Properties	Value	Field Options
<input checked="" type="checkbox"/> Type	Set in the form	[empty]	<pre>[{ "option": "LOADINGOPTIONS", "value": { "filter": [{ "filterType": "AND", "operator": "EQ", "property": "ValidID", "type": "NUMERIC", "value": 1 }] } }]</pre>
<input checked="" type="checkbox"/> Team	Set in the form	[empty]	<pre>[{ "option": "USE_OBJECT_SERVICE", "value": 1 }]</pre>



Field	Properties	Value	Field Options
<input type="checkbox"/> SLA criterion - SLA / service agreement	Set in the form	[empty]	[{ "option": "OBJECT", "value": "SLA" }, { "option": "SHOW_INVALID", "value": 0 }]
<input checked="" type="checkbox"/> Channel	Set in the form	[empty]	[empty]
<input type="checkbox"/> Account time	Set in the form	[empty]	[{ "option": "STEP", "value": 1 }, { "option": "EXCEPTS_EMPTY", "value": 1 }, { "option": "UNIT_STRING", "value": "Translatable#Minutes" }]



Field	Properties	Value	Field Options
<input type="checkbox"/> Owner	Set in the form	[empty]	<pre>[{ "option": "AUTOCOMPLETE", "value": 1 }, { "option": "LOADINGOPTIONS", "value": { "filter": [{ "filterType": "AND", "operator": "EQ", "property": "ValidID", "type": "NUMERIC", "value": 1 }], "query": [["requiredPermission", "TicketRead,TicketCreate"]] } }]</pre>



Field	Properties	Value	Field Options
<input checked="" type="checkbox"/> Priority	Set in the form	3 normal	<pre>[{ "option": "LOADINGOPTIONS", "value": { "filter": [{ "filterType": "AND", "operator": "EQ", "property": "ValidID", "type": "NUMERIC", "value": 1 }] } }]</pre>



Field	Properties	Value	Field Options
<input checked="" type="checkbox"/> Status	Set in the form	open	<pre>[{ "option": "LOADINGOPTIONS", "value": { "filter": [{ "filterType": "AND", "operator": "EQ", "property": "ValidID", "type": "NUMERIC", "value": 1 }] } }]</pre>



9 Additional configuration options

- [Configuring the Team View Mode \(see page 215\)](#)
- [Time Recording \(see page 226\)](#)
- [KIX Pro REST API \(see page 231\)](#)
- [Show child ticket tab in ticket details \(see page 232\)](#)
- [Configuring object history \(see page 235\)](#)

9.1 Configuring the Team View Mode

Configuration key	tickets
--------------------------	---------

In the ticket dashboard, tickets are displayed sorted by team. Agents in KIX Pro can choose between different view modes here:

- List
- Kanban
- Calendar
- Map

Inhalte auf dieser Seite:

- [List view](#) (see page 217)
- [Kanban view](#) (see page 217)
- [Calendar view](#) (see page 219)
- [Map view](#) (see page 220)
- [Enabling/disabling team view modes](#) (see page 224)

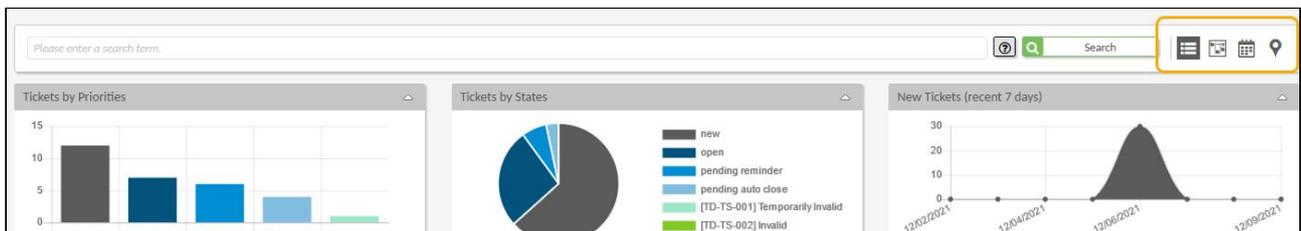


Fig.: The team view modes in the ticket dashboard

As an administrator, you can disable/enable the provision of view modes and configure their contents. This is done in the SysConfig key "tickets":

- Section "content[]" : contains the widgets displayed in the list view.
- Section "others[]" : contains the configuration blocks for the kanban, calendar and map view as well as the configuration block for de-/activating the view modes.

```

1  {
2    "id": "tickets",
3    "name": "Ticket Dashboard Configuration",
4    "type": "Context",
5    "contextId": "tickets",
6    "sidebars": [],
7    "explorer": [{}],
8    "lanes": [],
9    "content": [{}],
10   "generalActions": [],
11   "actions": [],
12   "overlays": [],
13   "others": [
14     {
15       "instanceId": "maps",
16       "configurationId": null,
17       "configuration": [{}],
18       "permissions": [],
19       "size": "large"
20     },
21     {
22       "instanceId": "kanban",
23       "configurationId": null,
24       "configuration": [{}],
25       "permissions": [],
26       "size": "large"
27     },
28     {
29       "instanceId": "calendar",
30       "configurationId": null,
31       "configuration": [{}],
32       "permissions": [],
33       "size": "large"
34     },
35     {
36       "instanceId": "ticket-views",
37       "configurationId": "ticket-views",
38       "configuration": [{}],
39       "permissions": [],
40       "size": "large"
41     }
42   ],
43   "dialogs": [],
44   "customizable": false,
45   "valid": true
46 }

```

Fig.: Configuration blocks of team view modes

Info

After making changes to the configuration, click on "Reload frontend configuration" to update the display in the frontend.

9.1.1 List view

The list view is the default view in the ticket dashboard. The chart widgets and ticket table are integrated in the section "content[]". You can configure their contents in the SysConfig key of the respective widget (see parameter "instanceID"). You can find more details on this under:

- The chart widget
- The table widget
- Configuration of dashboard tables



Fig. : List view mode

9.1.2 Kanban view

The Kanban view enables the tickets to be visually separated according to their processing status. The tickets are displayed as individual cards, which the processor can move to another column of the Kanban board using drag & drop. The processing status of the ticket changes automatically. The cards display various, configurable ticket information. As an administrator, you can specify which ticket information should be displayed on the cards.

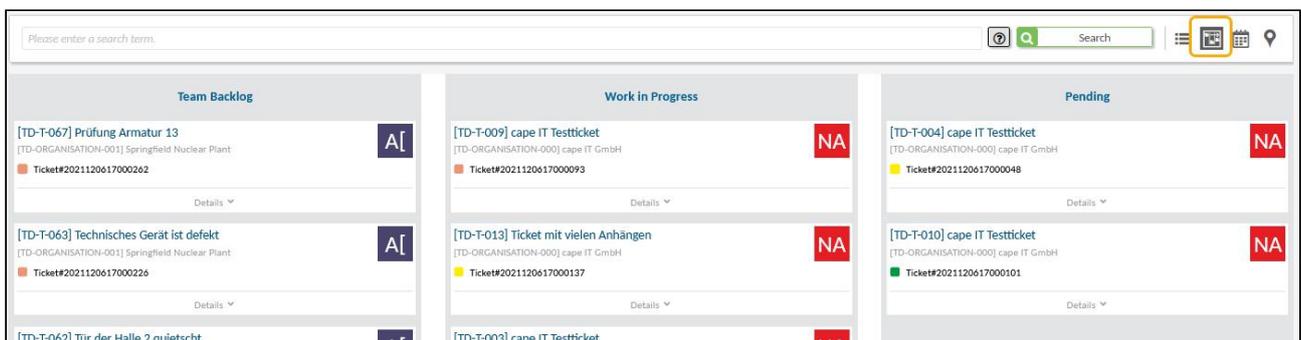


Fig. : Kanban view mode

The map contents are configured in the section "others[]" in the configuration block "{instanceID": "kanban" ...}.

Parameter	Description	Value
cardProperties	<p>Ticket properties to be displayed in the ticket marker overlay.</p> <p>You can remove unnecessary information from the card or add additional ticket information, e.g. the planned time expenditure or other dynamic fields.</p>	<p>Default: "ContactID", "StateID", "QueueID", "ResponsibleID", "Changed"</p> <p>Example: "ContactID", "StateID", "ResponsibleID", "DynamicFields.PlannedEffort", "DynamicFields.DFxyz"</p>
columns	<p>Columns of the Kanban board Default: Team Backlog, In Process (WIP), Waiting</p> <p>You can add additional columns to the Kanban board and change the status.</p> <p>id: Defines one column each in the Kanban board. Possible are:</p> <ul style="list-style-type: none"> • team-backlog (Team Backlog) • wip (in progress) • pending • closed (recently closed) <p>dropState: Defines the status that the ticket should receive when the ticket is moved to the corresponding column. All statuses created in the system are possible (menu Ticket> Status)</p>	<pre>"columns": [{ "id": "team-backlog", "dropState": "new" }, { "id": "wip", "dropState": "open" }, { "id": "pending", "dropState": "pending reminder" }]</pre>

9.1.3 Calendar view

In the calendar view, the tickets are displayed according to their due date. If a ticket is clicked on in the calendar, an overlay opens with a map that contains detailed information about the ticket.

For the display of the tickets in the calendar, the dates stored under "Plan start" and "Plan end" are used by default.

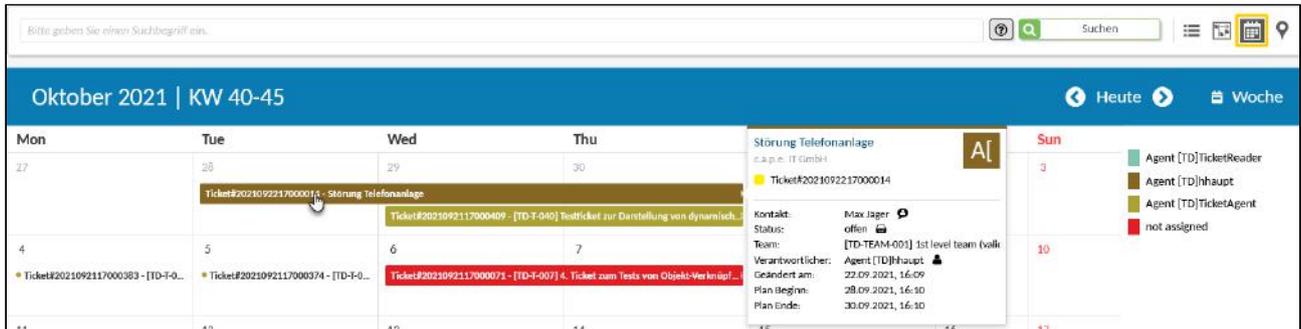


Fig. : Calendar view mode

You can configure the map content of the overlay. The configuration takes place in the section "others[]" in the configuration block "{instanceID": "calendar" ...}.

Parameter	Description	Example
properties	Ticket properties that should be displayed in the card content (overlay). You can remove unnecessary information from the card or add additional ticket information, e.g. the planned time expenditure or your own dynamic fields.	<pre>"properties": ["ContactID", "StateID", "QueueID", " DynamicFields.PlanBegin", "DynamicFields.PlanEnd", "DynamicFields.PlannedEffort", "DynamicFields.DFxyz"]</pre>

9.1.4 Map view

The map view visualises the assignment locations for tickets as well as the current position of the agents. It thus enables the planning and control of field operations. Incoming messages/tickets can thus be given directly to the service employee who is in the vicinity of the deployment location. This shortens effective response times and reduces travel times.

If the data changes, e.g. when the agent changes position, the map is automatically updated. To do this, the agent must use the Field Agent App with geo-tracking switched on on their mobile device.

You can also find instructions on how to use the map view in the KIX 18 Pro user manual.

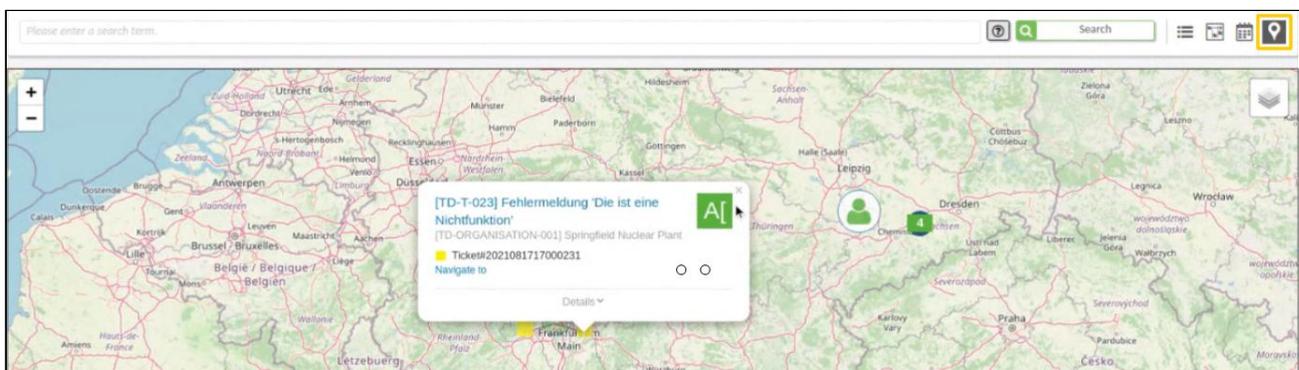


Fig .: Map view mode

The map uses Open Street Maps for display. <https://nominatim.org> is used as the geodata provider for resolving address information into geo-coordinates. The predefined bases are:

- for the representation of a ticket in the map: the address data of the organisation stored at the respective ticket.
- for the representation of the agent position: the address of the agent user.

The address data are specified in the configuration as KIX placeholders. This gives you the option of using alternative sources for address data, such as dynamic fields, the agent user's address, or similar. If, for example, the address of an asset is to be used, it can be placed in a dynamic field in the ticket by means of an automation job and the macro action "Fetch asset attributes". In the ticket configuration, you can then reference this dynamic field via a placeholder.

ⓘ Limitation of the default geodata provider

The resolution of geodata when using a nominatim.org service is limited to a maximum number of requests per time unit and IP address (see <https://operations.osmfoundation.org/policies/nominatim/>). Productive use is not recommended. There is a risk that address requests will be temporarily blocked, making the map function unavailable. We recommend the use of an alternative geodata provider such as <https://opencagedata.com>.

The following parameters can be configured for the integration of an alternative geodata provider:

Parameter	Description	Value
initialCoordinates	Initial coordination data with which the map is loaded.	<ul style="list-style-type: none"> • Default: 51.324572, 10.519871 (Middle of Germany) • Example: 49.939680368625325, 10.314198730887163 (Europe)
initialZoom	Initial zoom level with which the map is loaded.	<ul style="list-style-type: none"> • Default: 6 • Example: 5
addressDefinition	<p>Defines an address string for the ticket markers</p> <p>Based on this information, the location of the ticket is displayed on the map.</p>	<ul style="list-style-type: none"> • Default: Address of the organisation: "<KIX_ORG_Street>,
 <KIX_ORG_Zip>, <KIX_ORG_City>" • Example alternative: "<KIX_TICKET_DynamicField_Plac eofUse>"
layerGroups	Is currently not supported. Currently only layers for ticket owners are supported.	"OrganisationID", "OwnerID"
markerProperties	<p>Ticket properties to be displayed in the ticket marker overlay.</p> <p>You can add further ticket information to the overlay or remove information that is not required, e.g. the planned time expenditure or other dynamic fields.</p>	<ul style="list-style-type: none"> • Default: "ContactID", "StatelD", "QueueID", "ResponsibleID", "Changed" • Example: "ContactID", "StatelD", "ResponsibleID", "DynamicField.PlannedEffort", "DynamicField.DFxyz"



Parameter	Description	Value
geoDataProvider	<p>Specification of an alternative geodata server for KIX Cloud instances, e.g. https://opencagedata.com for the resolution of address data and the display of deployment locations (tickets) and agent positions in the map display.</p> <ul style="list-style-type: none">• url: URL of the API with placeholder <SEARCH_VALUE>, in which the defined address string is inserted.• latProperty: Array describing the property path to the latitude property in the response of the geo-data provider.• lonProperty: Array describing the property path to the longitude property in the response of the geo-data provider.• useTimeout: <code>true</code>, if only 1 request per second is to be sent to the geodata provider.	<pre>"geoDataProvider": { "url": "https://api.opencagedata.com/ geocode/v1/json? q=&key=YOUR_API_TOKEN_HERE", "latProperty": ["results","0","geometry","lat"], "lonProperty": ["results","0","geometry","lng"], "useTimeout": true }</pre>

Parameter	Description	Value
geoPositionTTL	<p>Expiry time of the position data (in minutes)</p> <p>The Field Agent app reports the geo-positions with a time stamp. Here you define after how many minutes the time stamp is considered "expired" and the position data expires.</p> <p>The agent's position is displayed on the map as long as the time stamp has not expired. If the agent changes its position, this is transmitted including a new time stamp. The basis for reporting the change of position is the information in the configuration conclusions (see below: Updating geo-position data). After the time stamp has expired, the agent position results from its address based on its stored contact or organisation data.</p>	Default: <code>"geoPositionTTL": 480</code>

 **Tip**

If geo-position data cannot be resolved or the geo-data provider delivers an error, you can investigate the reasons via the browser console (F12 key - Developer tools). You will receive a corresponding log entry there.

9.1.4.1 Updating geo-position data

If a service agent uses the Field Agent app, the app can transmit the geo-position data of the mobile device to the agent portal. KIX can display the agent position in the map view of the home dashboard based on the transmitted position data.

You can control the sending behaviour of the app in the *System > SysConfig* menu. This is done in the following configuration keys:

Key	Description	Values
KIXMobileApp::GeoPosition::ChangeThreshold	Defines from which position change the app sends the new position (distance in metres).	Natural numbers >0 The default value is used for invalid entries. Default: 1000
KIXMobileApp::GeoPosition::EnabledByDefault	Determines whether the position message is active or inactive after the user logs into the app for the first time. For further use, the settings in the app apply.	0 - inactive (Default) 1 - activ

9.1.5 Enabling/disabling team view modes

You can control the provision of team view modes globally via SysConfig. Team view modes that are not required can thus be deactivated and reactivated as required. The buttons of deactivated team view modes are hidden and thus cannot be selected.

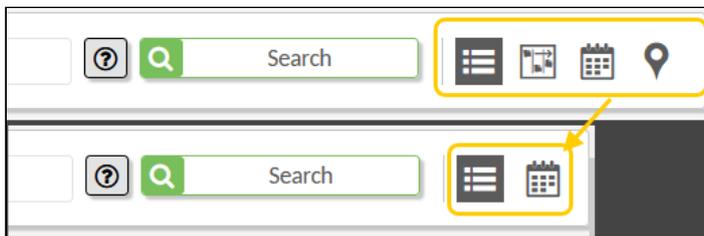


Fig.: Providing team view modes (all and partial).

The section "ticket-view s" in the code block "others[]" contains the parameter "widgets". Here you can determine for each view mode whether it is available. To do this, set the parameter "valid" to true or false.

```
101 "actions": [],
102 "overlays": [],
103 "others": [
104   {
105     "id": "ticket-views",
106     "name": "Ticket views",
107     "type": "Widget",
108     "widgetId": null,
109     "title": "",
110     "actions": [],
111     "subConfigurationDefinition": null,
112     "configuration": {
113       "id": "ticket-ticket-views-widget-config",
114       "name": "Ticket views Config",
115       "type": "Widget",
116       "widgets": [
117         {
118           "id": "ticket-view-list",
119           "name": "List",
120           "type": "Table",
121           "widgetId": null,
122           "displayName": "Translatable#List",
123           "icon": "kix-icon-legend",
124           "valid": true
125         },
126         {
127           "id": "ticket-view-kanban",
128           "name": "Kanban",
129           "type": "Kanban",
130           "widgetId": "kanban",
131           "displayName": "Translatable#Kanban",
132           "icon": "kix-icon-kanban",
133           "valid": false
134         },
135         {
136           "id": "ticket-view-calendar",
137           "name": "Calendar",
138           "type": "Calendar",
139           "widgetId": "calendar",
140           "displayName": "Translatable#Calendar",
141           "icon": "kix-icon-calendar",
142           "valid": true
143         },
144         {
145           "id": "ticket-view-maps",
146           "name": "Maps",
147           "type": "Maps",
148           "widgetId": "maps",
149           "displayName": "Translatable#Maps",
150           "icon": "kix-icon-tour",
151           "valid": false
152         }
153       ]
154     }
155   }
156 ]
157 "valid": true
```

Fig.: Enabling/disabling team view modes in the "tickets" SysConfig key

9.2 Time Recording

In KIX Pro, agents have the option of simply recording the time on the ticket. For this purpose, the planned target time (planned effort) is specified on the ticket. Each time the ticket is processed, an agent can indicate how much time he needed ("Book time" field). The total of the booked times is subtracted from the planned target time. The target time, the total of the booked times and the resulting difference are displayed in the sidebar widget "Time booking". In addition, the total time booked is displayed in the ticket detail view under "Accounted Time".

The target time can be set in different ways:

1. Manual setting of the target time using the "Planned effort" action
2. Automated setting via jobs
3. Automated setting using an [asset attribute](#) (see page 228).

The prerequisites and necessary components for simple time recording are already available in KIX Pro. You can use the time recording directly or configure it individually.

The following components are used for time recording:

1. The **dynamic field** "Planned Effort".
The dynamic field records the target time set on the ticket as a value and is initially included in KIX Pro. This value is set either automatically by a job or manually using the "Planned Effort" action. Positive numerical values can be entered in minutes. The target time is displayed in the "Time booking" sidebar and used to calculate the time difference. The configuration of the dynamic field cannot be changed. It is an internal dynamic field.
2. The "Planned Effort" **action**.
When used, the action opens a dialog in which the target time for the respective ticket can be specified. Unless otherwise configured, the action is only available to those responsible for the ticket. The action can be reconfigured individually.
3. **Jobs**
The dynamic field "Planned Effort" can be filled automatically via a job with the action "Set dynamic field" for a certain group of tickets. Using the filters in the job, you can define conditions for the execution of the job, e.g. B. a specific customer or ticket type. The jobs "Auto Set Planned Effort (Incident)" and "Auto Set Planned Effort (Service Request)" are initially delivered. For the ticket type "Incident" the default value is initially 30 minutes and for the ticket type "Service Request" 60 minutes. The jobs can be individually reconfigured if required.

You can optionally create additional jobs for setting the target time and configure under which conditions which value is set as the target time on the ticket.

It is also possible to read out the target time stored on an asset via a job and put it in the ticket. The

prerequisite for this is that an input field for the target time is configured in the asset class.

4. The **input field** "Time Accounting"

The field is offered in ticket forms and when creating articles. In this field, agents store the time that they worked on the ticket (in minutes). Posting negative values is possible. The entirety of the booked times is initially displayed in the "Ticket Information" lane as soon as times are booked. A time can also be booked by a job using the MacroAction "Book time".

5. The **sidebar widget** "Time Accounting".

In the sidebar widget, both the planned target time on the ticket and the total of the times booked on the ticket are displayed. The resulting time difference is also given. If the time difference is a negative value, the time difference is shown in red with a cross. If the result of the time difference is a positive value, it is displayed in green with a checkmark. This means that you can see at a glance how much of the planned time budget is still available for ticket processing.

Time booking scheme

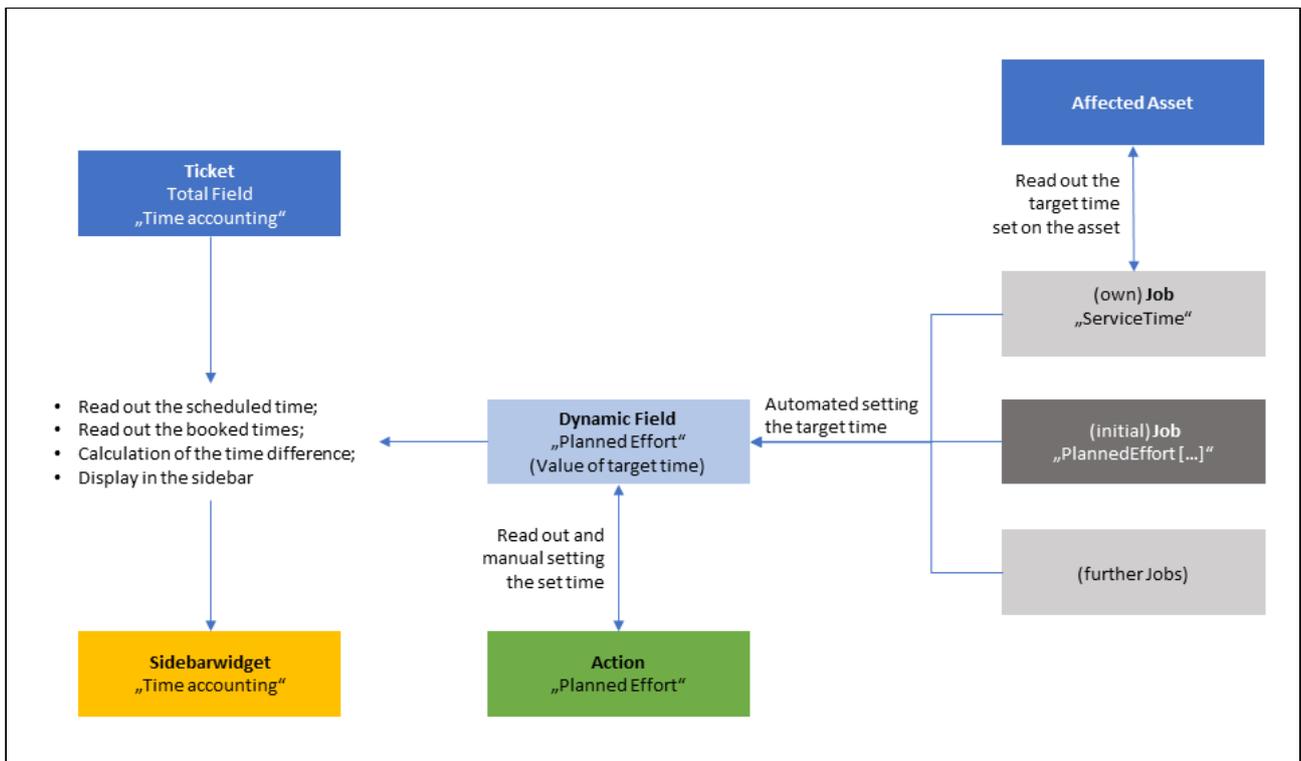


Fig.: Time booking scheme

9.2.1 Configuration example

This configuration example shows the use of an asset attribute to set the target time. The aim is that when a certain affected asset is selected, the target time stored on the asset is automatically set on the ticket. The example is

1. the service asset class has been expanded to include the "service time" attribute.
2. a service "Incident ACDC" is created and a service time of 60 minutes is stored there, as there is a corresponding service contract with the customer "Active Connect Data Center (ACDC)".
3. configured a job that reacts to the creation of a ticket of the type "Incident" in connection with the customer ACDC. The job sets the value stored in the asset attribute "ServiceTime" in the dynamic field "Planned Effort" on the ticket.
4. A ticket is created on which the time limit of the asset concerned is used.

1. Add the "Service time" attribute to the "Service" asset class:

1. In the Admin Module, navigate to *Assets > Asset Classes*.
2. Open the Service class and click Edit. A dialog opens in which you can change the class definition.
3. Drag the "Class Definition" text box a little larger to make it easier to edit the source code.
Alternatively, you can also use a JSON editor (<https://www.jsonformatter.io>).
4. Add the following code block to the desired location in the source code:

```
{
  'CountDefault' => 1,
  'CountMax' => 1,
  'CountMin' => 1,
  'CustomerVisible' => 0,
  'Input' => {
    'RegEx' => '^ \d + $',
    'RegExErrorMessage' => 'Please enter a positive integer.',
    'Type' => 'Text'
  },
  'Key' => 'ServiceTime',
  'Name' => 'Service time',
  'Searchable' => 1
},
```

5. Click on "Save" to apply the change to the class definition.
6. The dialog for creating / editing an asset now contains the input field "Service time". Agents can store the planned target time in it.

✓ **Tip**

- The "Service Time" attribute can also be added to other asset classes.
- You can create a pattern for the attribute designation ("Service Time") under *Internationalization > Translations*. This will automatically translate the name into the user language.

2. Create service "Incident ACDC":

1. Create a new asset of the "Service" class. To do this, click the "+ New" button.
Alternatively, you can also select the "Service" class in the asset module explorer and click the "New Asset" button there.
2. Select the "Service" class in the "Assets" tab.
3. Give the asset a meaningful name (in the example: "Incident ACDC").
4. Enter the usage and incident status and, optionally, other parameters.
5. Enter the value of the target time in minutes in the "Service time" field (in the example: 60).
The "Service time" field is the added field in the asset class.
6. Save the newly created asset.

3. Create job "AssetServiceTime":

1. In the Admin Module, navigate to *Automation > Jobs* and click on "New Job".
2. Use the following configuration in the job:
 - Job information
 - Job Type: Ticket
 - Name: e.g. "AssetServiceTime"
 - Validity: valid
 - Execution plan
 - Event: TicketDynamicFieldUpdate_ServiceTime, TicketCreate
 - Filter
 - a. Customer> included in> Active Connect Data Center (= customer ACDC)
 - Actions
 - 1st action: Set dynamic field
 - Dynamic Field Name: PlannedEffort (= name of the dynamic field that sets the target time on the ticket)
 - Dynamic Field Value: 60 (time in minutes).
3. Save the job.

Note: The initially delivered job "Auto Set Planned Effort (Incident)" must be set to "invalid".



4. Create a ticket

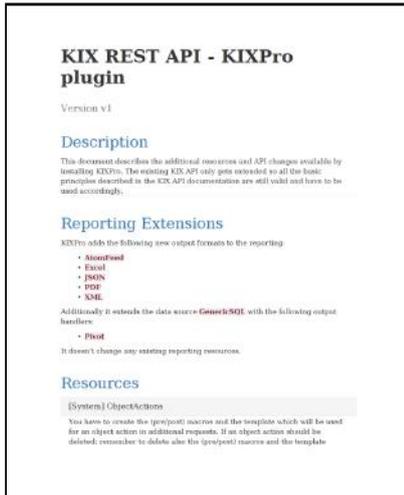
1. Create a new ticket. To do this, click the "+ New" button.
2. In the dialog that opens, enter the required information about the ticket and the following information:
 1. Type: incident
 2. Affected Asset: Incident ACDC.
3. After the ticket has been saved, the "Time booking" sidebar shows 60 minutes as "planned effort".



9.3 KIX Pro REST API

Below we offer the documentation for the REST API of KIX Pro for download.

The document describes the additional resources and API changes available through the installation of KIX Pro. The existing KIX API is only extended, so all the basics described in the KIX API documentation are still valid and must be used accordingly.



Note: Please download file after opening.

9.4 Show child ticket tab in ticket details

Configuration key	ticket-details-parent-child-widget
--------------------------	------------------------------------

The lane "Ticket Information" of KIX Pro contains an initially deactivated tab in the ticket details for displaying the child tickets stored on the ticket, including extended information such as ticket type, status and agent. If required, you can activate/deactivate this tab in the SysConfig key "ticket-details-parent-child-widget" (see below).

The tab lists all tickets that are stored in the [dynamic field "Child tickets"](#) (see page 52) on the ticket. The dynamic field "ChildTickets" is delivered with KIX Pro, but is not initially integrated into the ticket interfaces. In order to save child tickets on the ticket, it has to be integrated into a ticket [template](#) (see page 191) (e.g. "Default - New Ticket Template") or [ticket action](#) (see page 145) (e.g. "Ticket Edit").

The child tickets saved on the ticket will be linked to the current ticket as a child ticket. The current ticket thus becomes their parent ticket. The child tickets are also listed in the tab "Linked objects" and in the sidebar widget "Subtasks" - but with different or reduced information.

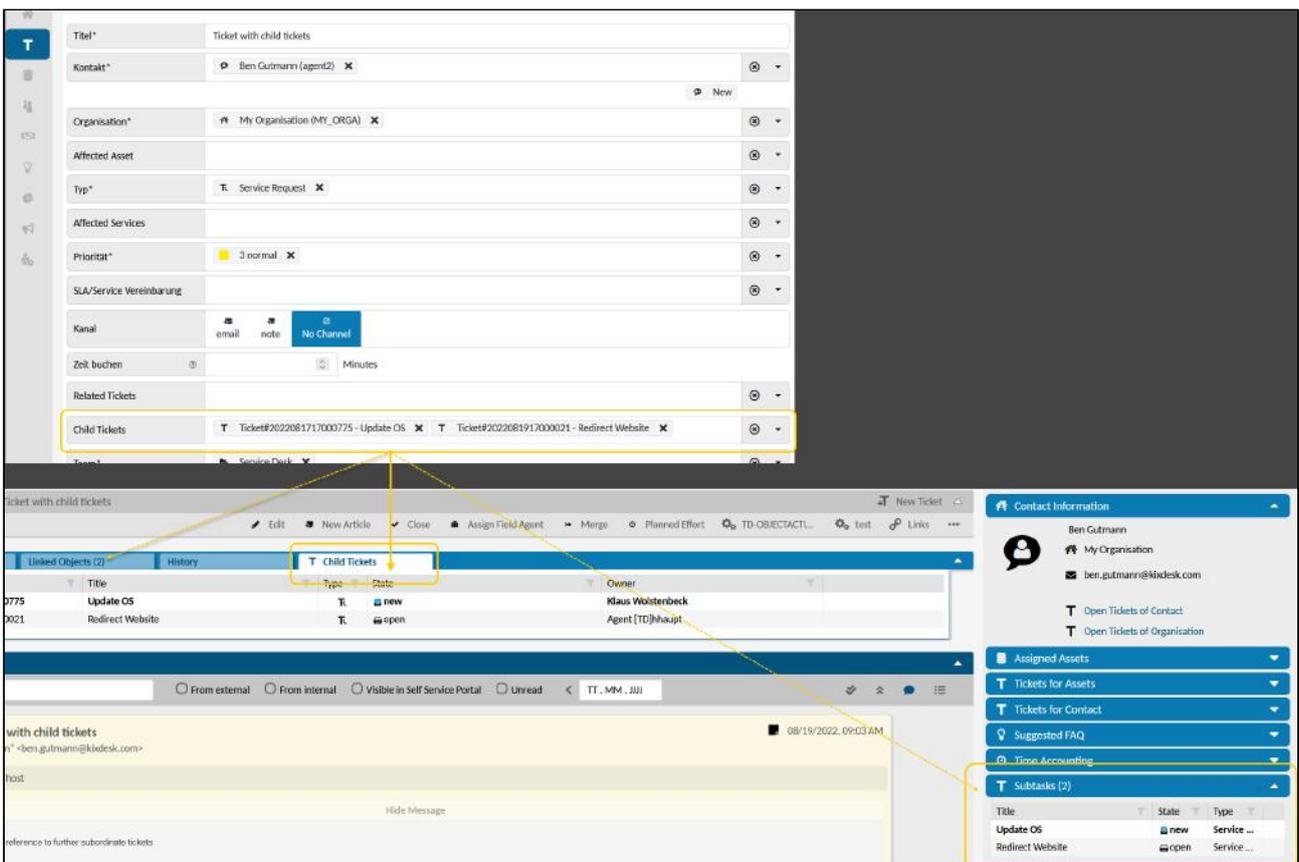


Fig.: Child tickets stored on the ticket in the ticket details

The following steps are necessary to display child tickets in the tab:

1. Integrate the dynamic field "ChildTickets" into the ticket so that agents can add child tickets to the ticket (e.g. in the dialogue "New ticket" and/or in the dialogue "Edit ticket").
2. Activate the SysConfig key "*ticket-details-parent-child-widget*" so that the initially hidden tab "Child tickets" is displayed.

9.4.1 Integrate dynamic field "ChildTickets" into the dialogue "new ticket"

Please proceed as follows:

1. In the Admin module of KIX Pro, navigate to the menu *Workflow > Templates*.
2. Open the template "Default - New Ticket Template" for editing.
This template generates the input mask for creating a new ticket.
3. Go to step 2 - "Input fields".
4. In the empty dropdown at the bottom of the page, select the dynamic field "ChildTickets" and the option "Set in mask".
Optionally, position the field at a different place in the form (via drag & drop).
5. Save the template.

Afterwards, the dynamic field "Child Tickets" is included in the "new ticket" dialogue, so that agents can select child tickets.

Alternatively, you can create your own [template](#) (see page 191) and include the dynamic field there. Agents can then save the child tickets to the ticket as soon as they use this template.

9.4.2 Integrating the dynamic field "ChildTickets" into the "Edit Ticket" dialogue box

Please proceed as follows:

1. In the Admin module of KIX Pro, navigate to the menu *Workflow > Actions*.
2. Open the action "Ticket Edit" for editing.
This action generates the input mask for editing a ticket.
3. Go to step 4 - "Input fields".
4. In the empty dropdown at the bottom of the page, select the Dynamic field "ChildTickets" and the option "Set in mask".
Optionally, position the field at a different place in the form (via drag & drop).
5. Save the action.

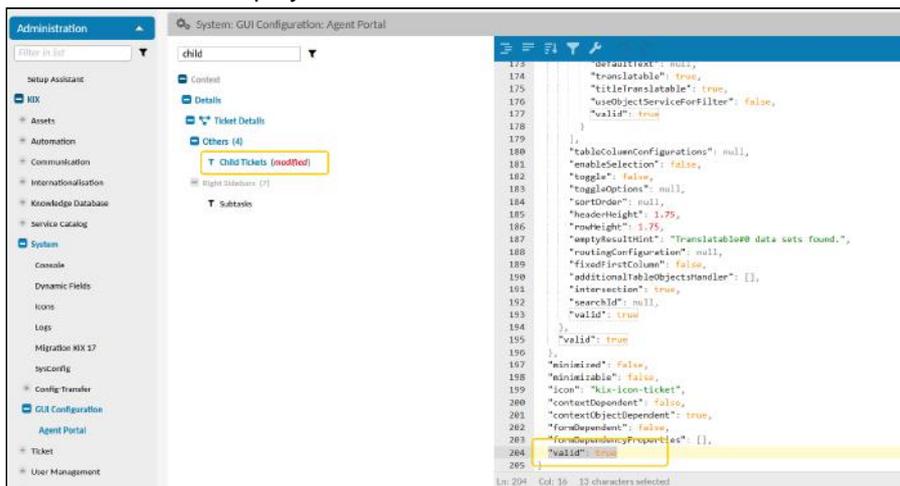
Afterwards, the dynamic field "Child Tickets" is included in the "Edit Ticket" dialogue, so that agents can select child tickets.

Alternatively, you can create your own [ticket action](#) (see page 145) and include the dynamic field there. Agents can then save the child tickets to the ticket as soon as they use this ticket action.

9.4.3 Activate SysConfig key

Please proceed as follows:

1. In the Admin module of KIX Pro, navigate to the menu *System > GUI Configuration > Agent Portal*.
2. Search for the SysConfig key *"ticket-details-parent-child-widjet"* (child tickets).
 For example, enter "child" in the search field.
3. In the editor, scroll down to the end of the JSON string.
4. Set the last parameter `valid` to `true`.
5. This activates the display of the tab "Child tickets".



6. Save the change and finally click on "Reload frontend configurations".

Afterwards, the tab "Child tickets" is included in the ticket details, so that the "Child tickets" saved on the ticket are listed in it.

To deactivate the display, set the parameter back to `false` (= default).

9.5 Configuring object history

Configuration key	ObjectHistoryTypes###Customer ObjectHistoryTypes###Organisation
--------------------------	--

The zoom view of contacts and organisations contains the "History" tab. Here you can see who made which changes to the contact or organisation. Logged are:

- the time of the change
- the user who made the change
- the change made.

If a change is made, KIX saves the parameters changed on the object in variables and creates a string from them, which is output in the history in prepared form as a comment: "\$1 changed (old: \"\$2\", new: \"\$3\")."

Variable	Value
\$1	Attribute which was changed
\$2	old value
\$3	new value

You can change this string as required, e.g. to output updates to a dynamic field specifically. This is done in the menu "System > SysConfig" in the above mentioned configuration keys.

There must be no entry for the individual attribute updates. Then the default setting is used: <WHAT> changed (old: "<OLDVALUE>", new: "<NEWVALUE>":

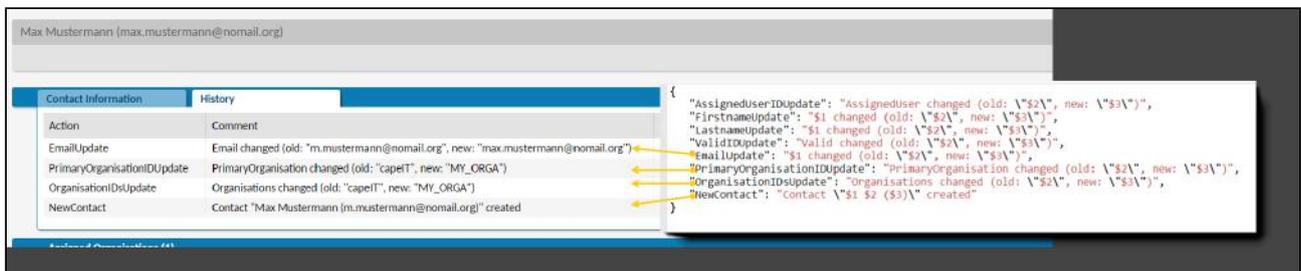


Fig.: Configuration of the object history

Please note:

If you use double quotation marks within the string, they must be masked with preceding backslash (e.g. "old: \"\$2\" "). Alternatively, you can use single quotation marks (e.g. "old: '\$2' ")

If the history is not to be displayed for selected users, you can define this via the permissions of the corresponding roles, e.g. role "Customer Reader":

Resource	/contacts/*/history	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Kommentar	-
Resource	/organisations	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Kommentar	-
Resource	/contacts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Kommentar	-

Fig.: Role has read permissions to contacts, but no permissions to the contact history.



10 Add-Ons

For KIX Pro you can purchase additional modules with which you can expand the basic functions of KIX. The additional modules allow u. bidirectional data exchange with other information systems such as Baramundi.

Product line: CONNECT

Connect Baramundi

Connect Database

[Connect Opsi \(see page 295\)](#)

Connect Webservice

Product line: ITIL Practices

[Add-on "ITIL Practices" \(see page 321\)](#)

[Implementation of ITIL practices by Means of Processes \(see page 325\)](#)

KIX Maintenance Plan

[Maintenance Plan \(see page 357\) \(Admin\)](#)

[Maintenance Plan \(see page 357\) \(User\)](#)

10.1 Connect

Under the KIX Connect product line, we offer you additional Add-on for KIX Pro that can be purchased individually. The Add-ons enable you to exchange data with third-party systems and use external database connections as a data source for further use in KIX.

Typical use cases for KIX Connect are e.g.

- Import of organizational, contact or asset data from CRM or ERP systems such as Navision, Axapta, Sugar CRM etc.
- Display of customer data, controlling information from CRM or ERP systems such as Navision, Axapta, Sugar CRM etc.
- Import of asset data from endpoint management/inventory/discovery systems such as Baramundi, i-doit, OPSI, etc.
- Display of stocks for article numbers ("Warehouse Integration")
- Transfer of ticket data from other systems
- Provision of selection values in form fields on tickets, organizations, contacts or FAQ entries from CRM or ERP systems such as Navision, Axapta, Sugar CRM etc.

The current data from other database sources are thus available to the agents for use in KIX. For example, the data will

- referenced in dynamic fields on tickets, contacts, organizations or FAQ entries
- displayed contextually in detailed views or dashboards
- used as the basis for data imports into the KIX Asset Management (CMDB).
- used to create tickets.

Content on this page:

- [Installation](#) (see page 239)
- [Common functional scope of the Connect Add-ons](#) (see page 239)
- [Field type "Data Source"](#) (see page 240)
 - [Parameter](#) (see page 241)
- [Advanced Macro Actions](#) (see page 242)
 - [XSL Transformation](#) (see page 242)
 - [Parameter](#) (see page 244)
 - [XSLT Functions](#) (see page 245)
 - [XSLT peculiarities in KIX](#) (see page 251)
 - [GetObjectData](#) (see page 252)
 - [Parameter](#) (see page 253)
 - [Get Item List From Data Source](#) (see page 253)
 - [Parameter](#) (see page 254)
 - [Get Item From Data Source](#) (see page 255)

10.1.1 Installation

After applying for KIX Connect, we will provide you with an updated image of your system - usually the next working day. Carry out a system update as described in the chapter [Installing KIX Pro \(see page 9\)](#) . After that, KIX Connect is integrated into your system. Depending on the specific extension package, activation in the admin area of your KIX is still required.

10.1.2 Common functional scope of the Connect Add-ons

The Add-ons of the Connect line partly use common core functions which are described below.

- **Field type "Data Source"**
 - Dynamic fields of the type "Data Source" make it possible to store contents from external data sources on all objects that support dynamic fields.
 - Replaces the existing field types "RemoteDB" and "Invoker" in KIX Pro version 17.
 - Is only used in [Connect Database \(see page 263\)](#) and [Connect Webservice \(see page 299\)](#) .

- **Advanced Macro Actions**

Macro Action	Description	Hints
Get Item From Datasource	Performs a detail query on a Connect data source.	Parameters and details see Macro Action "Get Item From Data Source" Application example see: Connect Database (see page 263) (Use in Macro Actions)
Get Item List From Datasource	Performs a list request to a Connect data source.	Parameters and details see Macro Action "Get Item List From Data Source" Application example see: Connect Database (see page 263) (Use in Macro Actions)
Get Object Data	Determines the needs-based content of KIX business objects. E.g. tickets with articles and attachments, assets with version data, organization and contact data	Parameters and details see Macro Action "Get Object Data" Application example see: Connect Webservice (see page 299) (Example Redmine Issue)

Macro Action	Description	Hints
XSL Transform	Preparation and processing of data for other macro actions. Allows the creation of complex data structures.	Parameters and details see Macro Action "XSL Transform" Application example see: Connect Webservice (see page 299) (Example Redmine Issue)

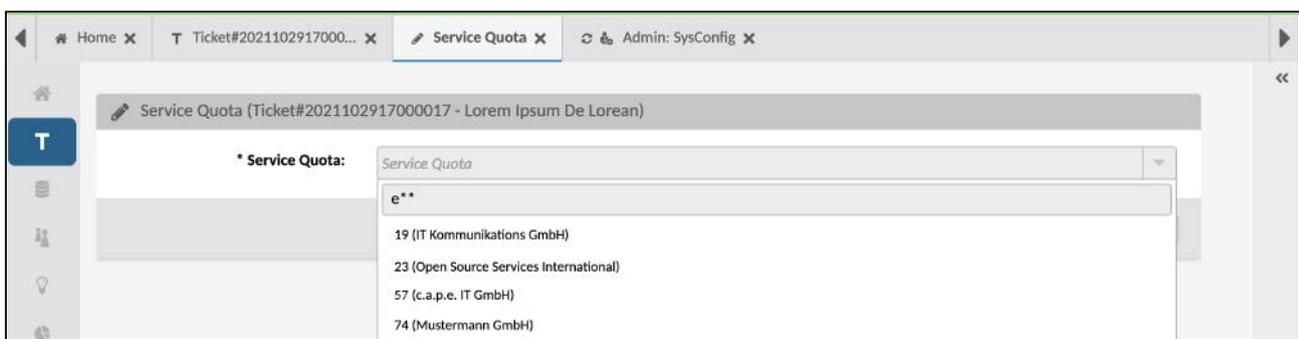
⚠ Please note

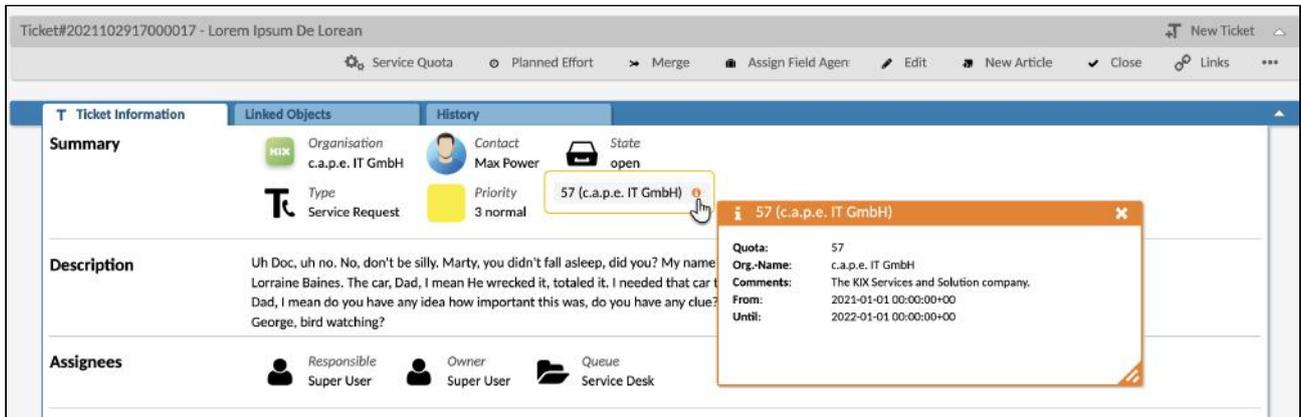
Some macro actions of the Connect line contain a debug option. If you activate debugging, "*Automation::MinimumLogLevel*" must be set to "`Debug`".

Otherwise, only the error messages delivered by the Macro Action will be captured and displayed in the job history, but all other debug information will not.

10.1.3 Field type "Data Source"

With dynamic fields of the "Data Source" type, form fields can be integrated whose value range is defined in Connect data sources. The functionality corresponds to that of the "Asset Reference" field type. In this way, single and multiple values can be stored in a form field and the information displayed in an overlay. It is also possible to provide and use these dynamic fields in templates and actions for the Self Service Portal - but please note the access restrictions (role assignment) stored at the data source assigned to the respective field. The desired customer user roles, e.g. "Customer", then also require access authorization to the data source.





10.1.3.1 Parameter

In addition to the general parameters, the behavior of a field of type "Data Source" is defined by the following parameters:

Parameter	Description	Example
CacheTTL	Defines how many seconds a data set that has been requested from the data source is kept in the KIX cache before a new request is sent to the data source.	3600
Data Source	The data source to query	"Servicequotas (DB)"
Default Display Column	<p>Array of column display pairs. Defines the content of the overlay when displaying the dynamic field in the detail views.</p> <ul style="list-style-type: none"> • Display: Defines the labeling of the single value. • Column: Defines which attribute of the detail query at the data source (with DB alias/DB column name) is used for the display value. 	<p>Column: quota - Display: Quota Column: cname - Display: Org.-name Column: startdate - Display: From Column: enddate - Display: Until</p>

Parameter	Description	Example
Display Pattern	Defines which attributes of the list request are used to display an individual entry (view mode and selection list). The attribute names (DB column names or aliases) are marked with "<" and ">".	<quota> (<cname>)

10.1.4 Advanced Macro Actions

10.1.4.1 XSL Transformation

XSL Transformation (XSLT for short) is a language for transforming XML documents. This macro action allows the conversion of an incoming data structure into another data structure. Value translations, value extractions and structural changes can thus be made. KIX uses XSLT version 1.1.

The purpose of the macro action is to prepare or process data for use in other macro actions, e.g. for "[Webhook Extended \(see page 299\)](#)". It can use the result of XSL transformations as content in the request. Likewise, results of web requests (responses) can be prepared for further use in other macro actions, e.g. for "CreateOrUpdateAsset". When using the XSL transformation, KIX's own functions are available to extend the possibilities of XSLT.

The macro action takes a data structure defined as a macro variable or character string, converts it into XML and applies the XSL transformation based on an XSLT template. The result of this conversion is then available for further use.

Examples of practical use cases are:

- the storage of external system IDs for tickets after their transmission (Redmine, Confluence, Jira, etc.)
- the preparation of asset information for storage in the KIX-CMDB (inventory)
- or error handling of Webhook Extended calls.

You can find an example of how to create a Redmine issue from a KIX ticket under [Connect Webservice \(see page 299\)](#) (example 2).



Parameter

Parameter	Description	Example
Data	<p>The source object or JSON string to transform.</p> <p>If a JSON string is specified, KIX placeholders are supported.</p>	<p><code>\${SomeMacroVariable}</code></p> <p>oder JSON-String, z.B.:</p> <pre>{ "issue": { "subject": "<KIX_TICKET_Title>", "description": "Lorem ipsum..." } }</pre>
Debug	<p>Enable/disable debug outputs.</p> <p>If activated, the debug information provided by the macro action is displayed in the job history.</p> <p>Requirement: The SysConfig key "<i>Automation::MinimumLogLevel</i>" must be set to the value " <code>Debug</code> ". Otherwise, only the error messages delivered by the macro action are recorded, but all other debug information is not.</p>	<p>yes no</p>

Parameter	Description	Example
XSL Template	<p>The XSL template for the transformation to be performed.</p> <p>The result of a transformation must be contained in the root path of the document for further use.</p> <p>The example shown includes an evaluation of the HTTP status code and processing of the HTTP response content. Error handling can be set up in conjunction with the "condition" macro action.</p>	<pre><?xml version="1.0" encoding="UTF-8"?> <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/ XSL/Transform" xmlns:KIX="urn:KIX"> <xsl:output method="xml" encoding="utf-8" indent="yes"/> <xsl:template match="RootElement"> <root> <xsl:variable name="HTTPCode" select="HTTPCode"/> <xsl:choose> <xsl:when test="\$HTTPCode='201'"> <RMIssueID><xsl:value-of select="Result/issue/id"/></RMIssueID> </xsl:when> <xsl:otherwise> <RMIssueID><xsl:text>Not transferred to Redmine.</xsl:text></ RMIssueID> </xsl:otherwise> </xsl:choose> </root> </xsl:template> </xsl:stylesheet></pre>
XSL TransformedData	Variable name for the transformation result structure	RMRequestPrepared

XSLT Functions

KIX provides some specific functions for resolving identifiers (determining contact, organization, asset IDs from names, login identifiers or object numbers) and for additional options in XSLT mappings. Calling the actions when using an XSL template leads to the function call being replaced with the result of the function.



Function	Description	Possible Values	Return Values
AccessTokenLookup (String)	Returns an OAuth2 access token for the OAuth2 profile named by parameter.	'Office 365'	String
AssetClassGetIDByName (String)	Asset Class ID	'Computer'	String
AssetGetIDByClassAndAttributeValue (String, String, String, String)	ID of asset based on asset class and any class spec. Attribute('AssetClass', 'AttributeKey', 'SearchValue', 'JoinSeparator', 'SplitSeparator')	'Computer', 'FQDN', 'chnb001.example.com', ',', ''	String
AssetGetIDByClassAndName (String, String)	ID of the asset based on asset class and name ('AssetClass', 'AssetName')	'Computer', 'chnb001'	String
AssetGetIDByName (String)	Asset ID based on asset name	'ddsrv009'	String
AssetGetIDByNumber (String)	Asset ID based on asset number	'634000001'	String
Base64Decode (String)	Base64 decoding of the given character string	'U29tZUJhc2U2NFN0cmZw=='	String
Base64Encode (String)	Base64 encoding of the transferred data/character string	'SomeDataString'	String
ConfigGet (String)	Value of the SysConfig entry of type "String" denoted by "String"	'SysConfigKey'	String
ContactGetIDByEmail (String)	ID of the contact with the given email	'max@example.com'	String



Function	Description	Possible Values	Return Values
ContactGetIDByUserLogin (String)	ID of the contact with the specified user login (only if there is a user)	'contact075'	String
DeploymentStateGetIDByName (String)	ID of deployment state	'Production'	String
GeneralCatalogGetIDByClassAndName (String,String)	ID of the General Catalog entry for the GC class and display value ('GC-Class', 'GC-Name')	'ITSM::ConfigItem::Computer::Type', 'Desktop'	String

Function	Description	Possible Values	Return Values
GenerateTOTP (String, String, String, String, String)	<p>Generates a "Timebased One Time Password" (TOTP) from the transferred parameters. This can be used via a macro variable in the header of the WebhookExtended to use TOTP MFA authentications in webhook calls.</p> <ul style="list-style-type: none"> Parameter 1: Base32Secret - The shared secret in Base32 encoding. Parameter 2: TimeStep - Specification for the validity period of a token. Default value: 30. Expects a natural number. Parameter 3: Digits - Length of the token. Default value: 6. Permitted are 6, 7 or 8. Parameter 4: Algorithm - Encryption algorithm for the token. Default value: SHA1. 'SHA1', 'SHA256' and 'SHA512' are permitted (capitalisation is relevant). Parameter 5: Previous - Generates a previous/future OTP. Default value: 0 	'KRUGKU3FMNZGK5CTORZGS3TH','30','6','SHA1','0'	String
IncidentStateGetIDByName (String)	Incident Status ID	'Operational'	String
MD5Sum (String)	MD5 sum of the character string passed	'SomeString'	String



Function	Description	Possible Values	Return Values
OrganisationGetIDByName (String)	ID of the organization with the given name	'Harmony Shoal Inc.'	String
OrganisationGetIDByNumber (String)	ID of the organization with the given number	'CN007'	String
PatternMatch (String, String)	Returns 1 if the given string contains the RegEx pattern. Perl regular expressions are used.	'String', 'REGEX'	String
PatternRemove (String, String)	Removes the part matching RegEx from the given character string. Perl regular expressions are used.	'String', 'REGEX'	String
PatternReplace (String, String, String)	Replaces the part that applies to RegEx from the given character string. Perl regular expressions are used.	'String', 'REGEX', 'replacement'	String
SystemDataDelete (String)	Removes the full entry for a passed key from KIX, e.g. for auth token.	'N2GBearerToken'	String
SystemDataGet (String)	Returns a value permanently stored in KIX and identified by the key. Allows a bpsw. use the auth token received in a previous "WebhookExtended" call in further calls.	'N2GBearerToken'	String



Function	Description	Possible Values	Return Values
SystemDataSet (String, String)	Stores a value for a given key permanently in KIX (e.g. auth token), which is generated by a user password request. Allows, for example, to save an auth token received via "WebhookExtended" in the system.	'N2GBearerToken', '__eyJ0eXAiOiJKV1QiLC...'	String
TimeStamp ([String])	Returns a timestamp (now+offset seconds) in the format 'YYYY-MM-DD hh:mm:ss' (optional).	'3600'	String
TimeString (String, [String])	Like "TimeStamp(OffsetInSeconds)", but the format can be defined according to strftime ⁷ .	'Format', '3600'	String
UserGetIDByLogin (String)	Retrieves the ID of the user identified by the given login name.	'mamu'	String
UserGetLoginByID (String)	Retrieves the login of the user identified by the passed ID.	'1'	String

⁷ <https://metacpan.org/pod/POSIX::strftime::GNU>

XSLT peculiarities in KIX

Return values from XSL functions

As a rule, XSLT functions return strings. These can be reused directly within the XSLT. If specific XSLT functions are called that return complex data structures, the individual elements of the structures can be accessed.

In the following example, a hypothetical function *MyTicketGet()* is called that returns a ticket hash. The example uses specific information from the hash for further processing.

Call in the XSLT template:

```
<xsl:variable name="Ticket" select="KIX:MyTicketGet(123)"/>
<Ticket>
  <ContactID><xsl:value-of select="$Ticket//ContactID"/></ContactID>
  <StateName><xsl:value-of select="$Ticket//State"/></StateName>
</Ticket>
```

Result of the JSON structure after XLS transformation:

```
"Ticket": {
  "ContactID": "1701",
  "StateName": "open"
}
```

XML-incompatible parameter names

XSLT is a variant of XML. Therefore, restrictions apply to the characters that can be used as names for tags. JSON is not subject to the same restrictions. For example, key strings such as "@name" are possible in JSON. If a prefix is to be used, it must be specified in the respective XML tag as an attribute "prefix".

For example, this creates:

```
<id prefix="@">SomeIdentifierString</id>.
```

creates the JSON structure:

```
"@id": "SomeIdentifierString", .
```

Typing of Outgoing Data

If certain data types are to be enforced in outgoing requests/webrequests and the JSON contained therein, e.g. because the interface to be addressed requires a strong typing (e.g. Boolean or Number), this can also be controlled via XSLT.

```
<eventSequenceNumber convert="number">
  <xsl:text>1701</xsl:text>
</eventSequenceNumber>
<someBooleanValue convert="boolean">
  <xsl:text>0</xsl:text>
</someBooleanValue>
<someOtherBooleanValue convert="boolean">
  <xsl:text>1</xsl:text>
</someOtherBooleanValue>
```

The following JSON structure is created:

```
"eventSequenceNumber": 1701,
"someBooleanValue": false,
"someOtherBooleanValue": true,
```

References

- https://en.wikipedia.org/wiki/XSL_Transformation⁸
- https://www.w3schools.com/xml/xsl_intro.asp
- <https://www.tutorialspoint.com/xslt/index.htm>
- <http://www.cheat-sheets.org/saved-copy/XSLT-1.0.pdf>

10.1.4.2 GetObjectData

The macro action "Get Object Data" carries out a KIX-internal query and returns the determined object with all details for further processing to further automation steps. This can be used in ticket jobs, for example, to obtain all ticket data - including the associated items - and to make them available for further processing in XSL transformations. Another use case is the reference to the data (organisation, contact or assets) referenced on a ticket. In this way, more information is available than is possible with the placeholders known from text modules and answer templates.

⁸ https://de.wikipedia.org/wiki/XSL_Transformation

Parameter

Parameter	Description	Example
Contains	subobjects to include or IDs to resolve - see backend REST API documentation	DynamicFields, Priority, Articles, Queue, State, Links
Expands	sub-objects to extend - see backend REST API documentation	Links
Item List	Identifier of the macro variable for further processing of the return	CompanyList
Objekt-ID	ID of the object to get	123 <code>\${SomeMacroVariable}</code>
Object Typ	Business object type to be obtained	Asset Ticket Organisation Contact

10.1.4.3 Get Item List From Data Source

The Macro Action "Get Item List From Data Source" performs a list request to a Connect data source. Free search criteria and values for fixed search attributes can be specified. The return structure is an "array of hashes", where the attribute/column names or aliases of the list request form the keys. A complex example is included in the documentation for [Connect Database](#) (see page 263).

Example return value

```
[
  {"ID":11,"cno":"EMA","cname":"EM Automotive GmbH & Co. KG","quota":24},
  {"ID":18,"cno":"HMS","cname":"Hypokrates Medical Services","quota":95}
]
```

Usage return value

As a rule, the return value is used in a further macro action "Loop". Since this expects an array, no processing of the return value is required. Within this loop, ID values can be accessed using the dot notation "`${LoopVariable.ID}`".

Parameter

Parameter	Description	Example
Data Source	The data source to query	"Servicequotas (DB)"
Item List	Identifier of the macro variable for further processing of the return	CompanyList
Parameters	JSON structure of the fixed parameters to be applied in the data source	{"Param1": 123, "Param2": "Text Pattern", "Param3": [1,2,3]}
Search	JSON structure of the search criteria to be applied in the data source. The structure corresponds to that of filter criteria in the KIX REST API	<pre>{ "Item": { "AND": [{ "Field": "id", "Operator": "GT", "Type": "NUMERIC", "Value": "5" }, { "Field": "id", "Operator": "LT", "Type": "NUMERIC", "Value": "9" }] } }</pre>

10.1.4.4 Get Item From Data Source

The macro action "Get Item From Data Source" carries out a detailed request to a Connect data source. The return structure is a hash, with the attribute/column names or aliases of the detail query forming the keys. A complex example is included in the documentation for [Connect Database](#) (see page 263) .

Example retrun value
<pre>{ "ID": 18, "cname": "Hypokrates Medical Services", "cno": "HMS", "quota": 95, "comments": "Some generic comment here.", "startdate": "2021-01-01 00:00:00+00", "enddate": "2022-01-01 00:00:00+00" }</pre>

Parameter

Parameter	Description	Example
Data Source	The data source to query	"Servicequotas (DB)"
Item	Identifier of the macro variable for further processing of the return	CurrCompanyData
Item ID	<p>ID of the record to retrieve. KIX placeholders and macro variables are supported.</p> <p>If the Macro Action is called within a "Loop" after a "Get Item List From Data Source", the ID value within the hash of the current loop variable can be accessed using dot notation (see Example column).</p>	<p>123 <KIX_TICKET_DynamicField_SQID> \$ {SingleValueMacroVariable} \$ {CurrCompany.ID}</p>

10.1.5 Connect Baramundi

Connect Baramundi is an additional module for KIX Pro. It offers the possibility to provide asset data in KIX based on the inventory solution Baramundi. The information available in a Baramundi can thus be used in the KIX Service Management System.

For this purpose, KIX provides an additional, separate asset class "Baramundi Device". The device data to be imported is obtained via the Baramundi bConnect interface. The synchronization is carried out by the "Baramundi - Device Sync" job. Both asset class and job are implemented when enabling Baramundi synchronization in KIX.

Content on this page:

- [Requirements](#) (see page 256)
- [Use](#) (see page 257)
 - [Activate Baramundi synchronization](#) (see page 258)
 - [Reset or change Baramundi synchronization](#) (see page 260)
- [Adjustments](#) (see page 260)
 - [Deactivation/activation of takeover "LastUser"](#) (see page 260)
 - [Limitation of synchronized Baramundi devices](#) (see page 261)
 - [Further adjustments](#) (see page 262)

10.1.5.1 Requirements

HTTP/S access to the bConnect interface is required to use Connect Baramundi. To do this, the FQDN and, if applicable, the port number of the bConnect interface must be specified. When using HTTPS, only correct SSL certificates may be used. The use of invalid or self-created certificates can negatively affect the function of the interface.

Furthermore, the username and password of a user authorized to log in to the bConnect interface are required. To use the username and password in the setup/configuration, an authentication token must be created from the username and password. Under Linux you can use the following command line (replace `<YourUserName>` and `<YourPassword>`)

```
SomeLinuxSystem:~# echo -n '<YourUserName>:<YourPassword>' | base64
PF1vdXJvc2VyTmFtZT46PF1vdXJQYXNzd29yZD4=
```

Under Windows, the string "`<YourUserName>:<YourPassword>`" must first be written to a file (here `UserNamePassword.txt`). This is then Base64 encoded. To display the authentication token, remove all lines containing "CERTIFICATE" from this file. The following two command lines reflect this:

```
C:\temp> certutil -encode -f UserNamePassword.txt UserNamePasswordB64.txt
C:\temp> type UserNamePasswordB64.txt | find /v "CERTIFICATE"
PF1vdXJvc2VyTmFtZT46PF1vdXJQYXNzd29yZD4=
```

10.1.5.2 Use

The plugin is configured and activated after installation (KIX On Premise: restart the KIX stack using "start.sh"; KIX Cloud: after feedback from support) in the admin area in the KIXConnect-Baramundi setup:

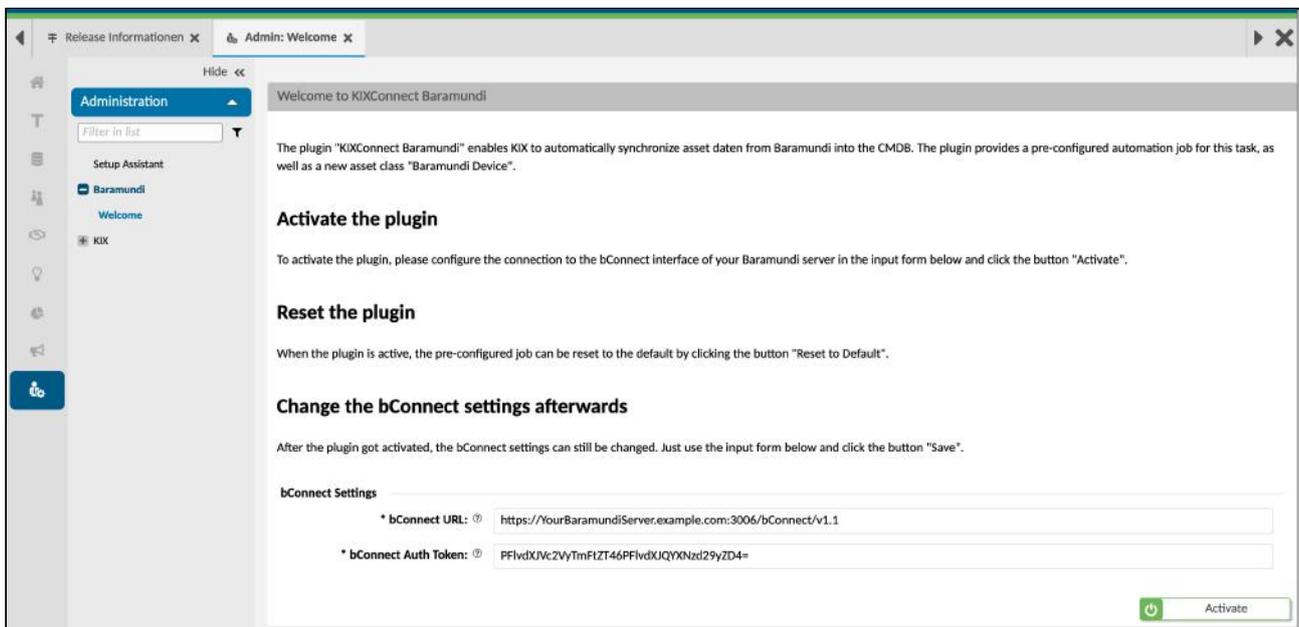


Fig.: Baramundi Setup

Activate Baramundi synchronization

To activate the additional module, store the bConnect URL (e.g. `https://YourBaramundiServer.example.com:3006/bConnect/v1.1`) in the Admin module under *Baramundi* > *Welcome* as well as the previously prepared authentication token (see Requirements). Then click on "Activate". The configuration is saved and creates the automation job "Baramundi - Device Sync".

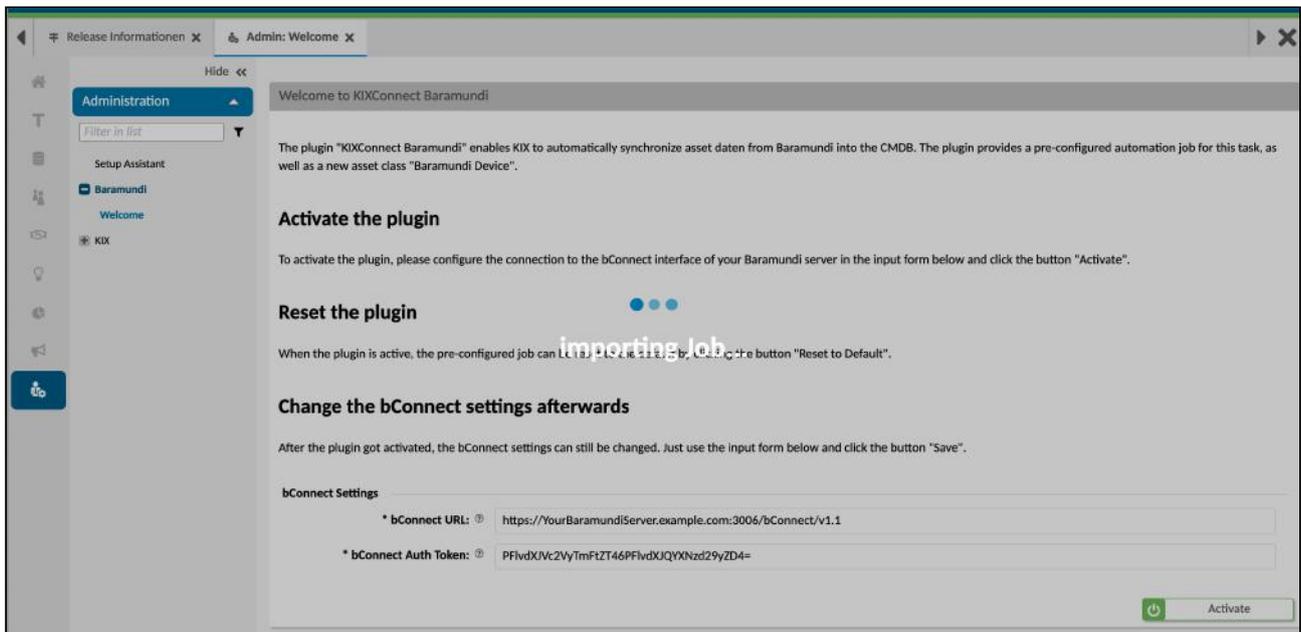


Fig.: Activate Baramundi

For a first Baramundi data import, we recommend that you then run the "Baramundi - Device Sync" job manually (admin module under *KIX* > *Automation* > *Jobs*). Depending on the content of the Baramundi data source, this can take a few minutes. If the Baramundi data import is to be carried out periodically in the future, assign a corresponding schedule to the job under "Execution plan", e.g. every day at 03:00 (see Creating or editing a job). Do not change the rest of the job configuration! After that, the configuration is complete.

Type	Name	Event ...	Time ...	Number of Actions	Last exe
<input type="checkbox"/>	Ticket	Anonymisation	✓	9	
<input type="checkbox"/>	Ticket	Auto Set Planned Effort (Incident)	✓	1	2022-09
<input type="checkbox"/>	Ticket	Auto Set Planned Effort (Service Request)	✓	1	2022-09
<input type="checkbox"/>	Synchronisation	Baramundi - Device Sync		2	
<input type="checkbox"/>	Ticket	Create FAQ Suggestion	✓	2	2022-09
<input type="checkbox"/>	Ticket	Customer Response - reopen from pending	✓	1	2022-09
<input type="checkbox"/>	Ticket	KIX Field Agent - Mobile Processing Rejected	✓	2	
<input type="checkbox"/>	Reporting	Periodic Reports		✓	1
<input type="checkbox"/>	Ticket	Put affected assets back into operational state	✓	1	2022-09
<input type="checkbox"/>	Ticket	Put affected assets into incident state	✓	1	2022-09

Fig.: Baramundi Synchronization Job

Baramundi - Device Sync

Job Information

Type:	Synchronisation	Changed at:	09/15/2022, 08:23 AM
Name:	Baramundi - Device Sync	Changed by:	not assigned
Last executed at:			
Comment:			
Validity:	valid		
Created at:	09/15/2022, 08:23 AM		
Created by:	not assigned		

Execution Plan

Time Based Execution

Day(s):

Time:

Actions (2)

Action	Skip	Comment
1 Baramundi Request		
2 Loop		

Fig.: Manually starting the "Baramundi - Device Sync" job

Reset or change Baramundi synchronization

If you have made adjustments to the configuration of the job, for example to synchronize specific, extended information, you can reset the default configuration to the delivery status at any time using the "Reset to default" button. The customizations will be removed.

After activating the synchronization, you can continue to change the settings for the bConnect interface, for example to store a changed URL or a new authentication token. To do this, make the necessary changes in the Baramundi setup and click on "Save". The changes are active the next time the job is run.

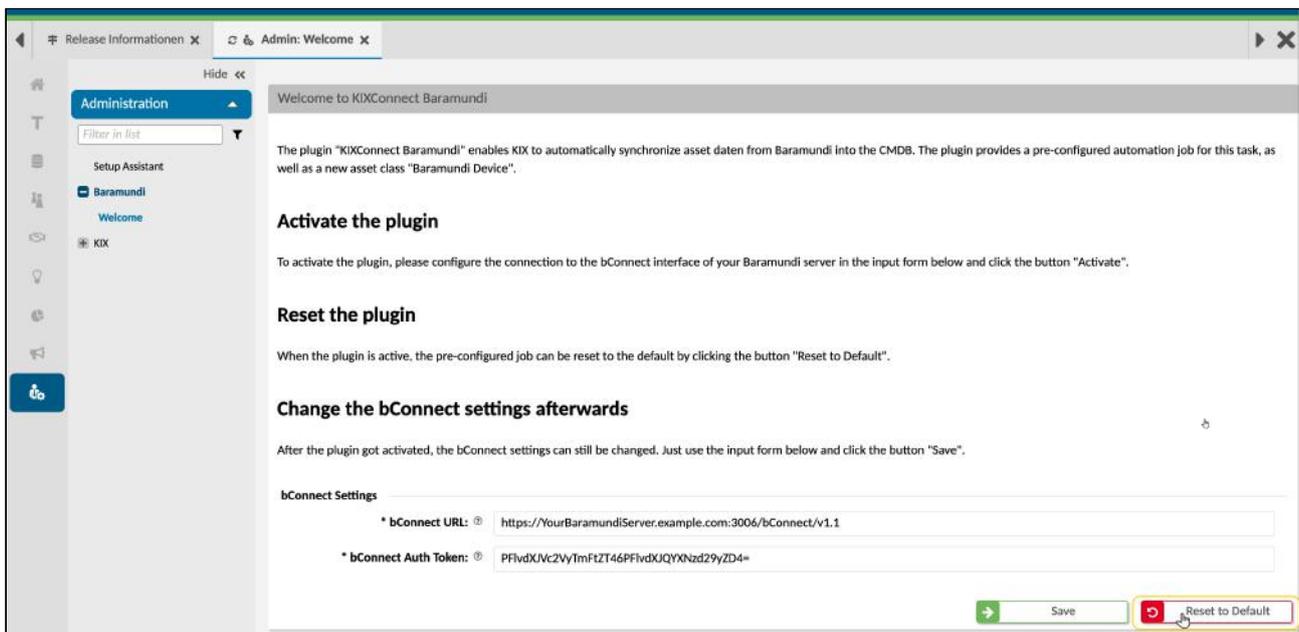


Fig.: Resetting the configuration

10.1.5.3 Adjustments

Deactivation/activation of takeover "LastUser"

The asset class "Baramundi Device" and the data comparison import not only the primary user but also the last user of a device. If this changes very frequently, it may make sense not to carry out the adjustment. In this case, adjust the configuration in KIX as follows:

1. Remove the relevant section from the asset class definition

LastUser section to remove (asset class)

```
{
  'Key'      => 'LastUserContact',
  'Name'     => 'Last User Contact',
  'Searchable' => '1',
  'Input'    => {
    'Type'    => 'Contact',
  },
},
{
  'Key'      => 'LastUser',
  'Name'     => 'Last User',
  'Searchable' => '1',
  'Input'    => {
    'Type'    => 'Text',
  },
},
},
```

2. Remove the relevant section from the data preparation in the "Baramundi - Device Sync" synchronization job (action 1 "XSL Transformation" contained in action 2 "Loop")

LastUser section to remove (data preparation)

```
<xsl:if test="string-length($LastUserContactID) > 0">
  <LastUserContact>
    <xsl:value-of select="$LastUserContactID"/>
  </LastUserContact>
</xsl:if>

<xsl:if test="string-length($LastUserString) > 0">
  <LastUser>
    <xsl:value-of select="$LastUserString"/>
  </LastUser>
</xsl:if>
```

Limitation of synchronized Baramundi devices

You can use the Baramundi feature "Dynamic Groups" to limit the devices to be synchronized from Baramundi. Define a corresponding group in Baramundi, assign the devices to this group and determine the "ID" of the group. As far as we know, this is only possible by making a request to the bConnect interface. If the query is to be limited in KIX, add the "DynamicGroup" parameter to the "/Endpoints" resource to be queried in the "Baramundi - Device Sync" job (example: "/Endpoints? DynamicGroup=BE7592D3-8E57-41DE-929A-FDA1D993C836")



Further adjustments

If you have any requests for adjustments to the data to be used in the "(Baramundi) Device" asset class or a separate consideration in different asset classes, please contact us.

10.1.6 Connect Database

The additional module Connect Database for KIX Pro enables the use of database connections as a data source for further use in KIX. A corresponding configuration is required for specific use. The database management systems are supported: PostgreSQL, MariaDB/MySQL or MS SQL via ODBC.

Inhalte auf dieser Seite:

- [Requirements](#) (see page 263)
- [Establishment](#) (see page 263)
 - [1. Set up Database Connections](#) (see page 264)
 - [Parameter](#) (see page 265)
 - [2. Set up Database Data Sources](#) (see page 267)
 - [Step 1 - Header Data](#) (see page 268)
 - [Step 2 - Database Queries](#) (see page 270)
- [Use of Data Sources](#) (see page 272)
 - [In Dynamic Fields](#) (see page 272)
 - [In sidebar tables](#) (see page 277)
 - [In Macro Actions](#) (see page 283)
- [Use of the KIX Database](#) (see page 294)

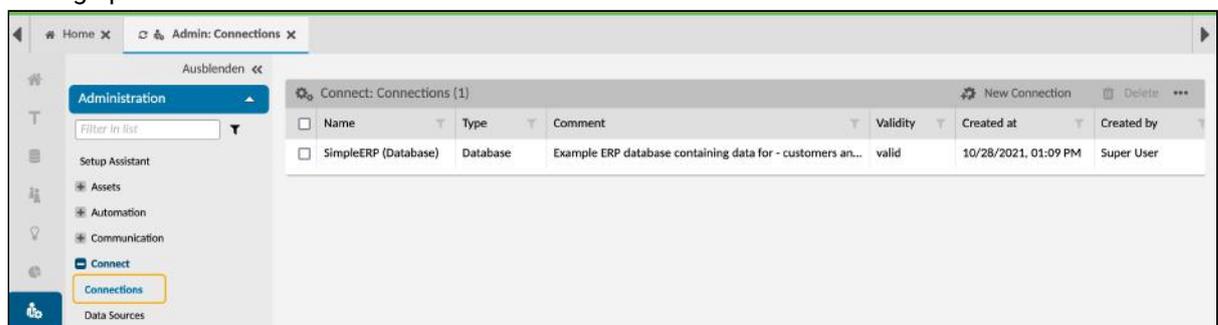
10.1.6.1 Requirements

The connection of a database data source requires direct access to the relevant database using authentication via user name and password. The relevant DBMS must be accessible via a database socket connection.

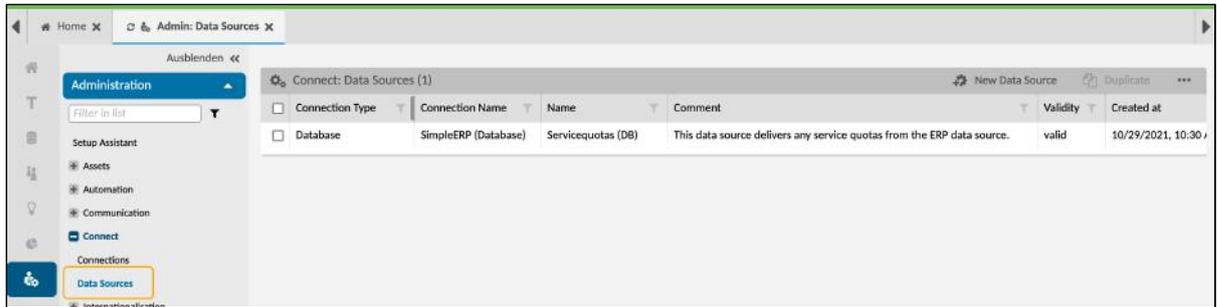
10.1.6.2 Establishment

The database connections and the data sources are set up in the explorer of the admin module under "Connect". There you will find the menu items for

1. Setting up a database connection and



2. Setting up the required database data sources.



Both must be set up to use an external data source.

1. Set up Database Connections

Database connections are set up and configured step by step in the *Connect > Connections* menu. Store the connection parameters to the database here. When saving, KIX tries to establish the database connection. If this is not successful, you will be informed and can correct the settings.

To access the database, a database connection requires information about the host, the database name, the user name and the password used to protect the database. This information must be provided and entered in the DSN field. Have this information ready.

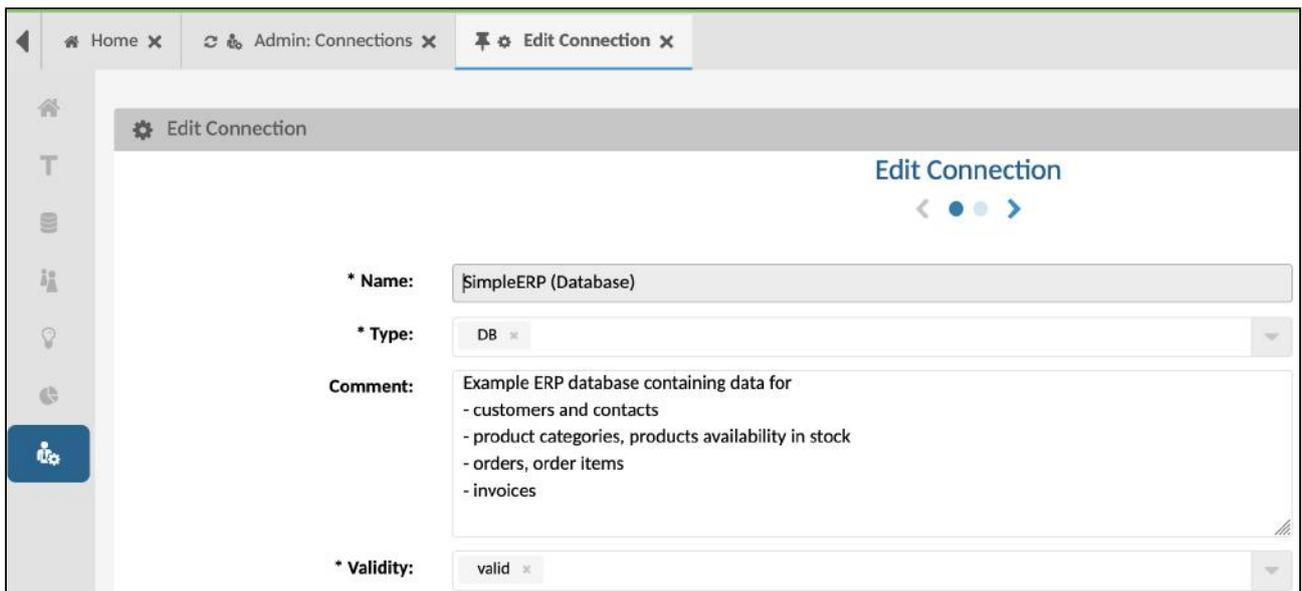


Fig. Setting up a database connection - step 1

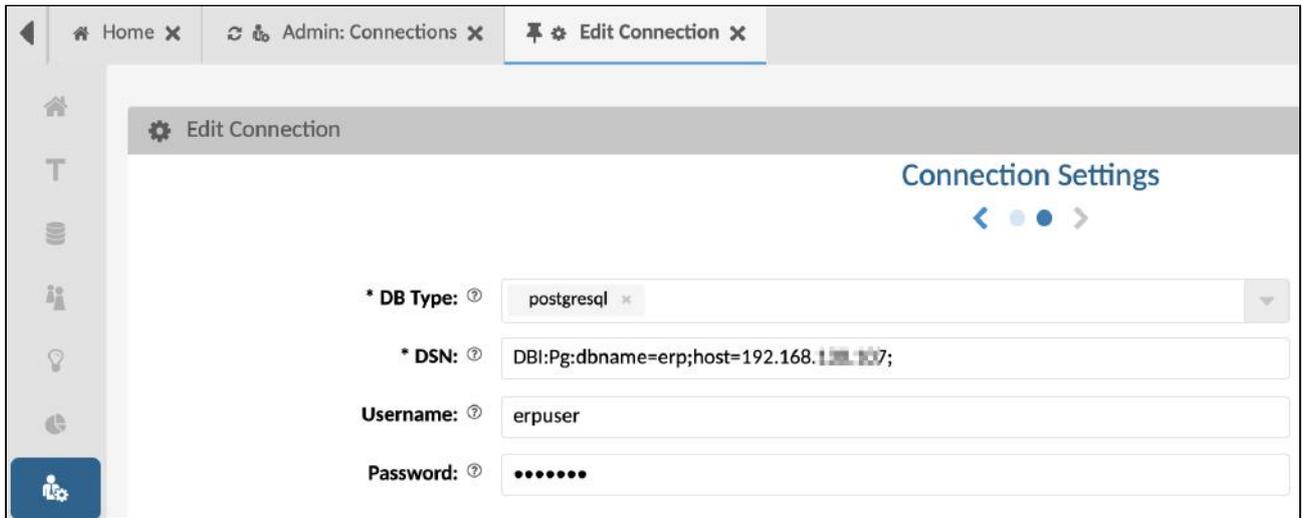


Fig. Setting up a database connection - step 2

Parameter

Parameter	Description	Example
Name	Name of data connection	SimpleERP (Database)
Type	Type of data connection (how is the data source accessed) The available selection depends on the installed additional modules.	DB
Comment	Optional comment	Example ERP database, contains data on: <ul style="list-style-type: none"> customers and contacts product categories, product availability in stock orders, item orders bills
Validity	disables/enables data source for further use	valid invalid

Parameter	Description	Example
DB-Type	Database type to connect to. This information is relevant if the type is not evident from the DSN.	postgresql mysql mssql
DSN	For database access, enter here: <ul style="list-style-type: none"> the host on which the database is located the database name the access data to the database: <ul style="list-style-type: none"> user and password. 	<ul style="list-style-type: none"> DBI :Pg:dbname=erp;host=192.168.130.107; DBI:Pg:dbname=kix;host=db; DBI:ODBC:Driver={ODBC Driver 17 for SQL Server}; Server=<FQDN/IP>, <Port>; Database=<Database>; UID=<login>; PWD=<password> <p>Replace in the above example the following parameters with their values:</p> <ul style="list-style-type: none"> <FQDN/IP> <Port> <login> <password> <p>Note: Access to an MS SQL server is via ODBC. Make sure that the user name and password are specified in the DSN. Both specifications are defined directly in the DSN for an ODBC connection. The individual parameters "DB user" and "DB password" are then not relevant.</p>
DB-User	Username/login used to access the database. This specification can be omitted when accessing an MS SQL server, since the user name is already specified in the DSN.	<ul style="list-style-type: none"> erpuser kix

Parameter	Description	Example
DB-Password	<p>Password of the user who accesses the database.</p> <p>This specification can be omitted when accessing an MS SQL server, since the password is already specified in the DSN.</p>	<ul style="list-style-type: none"> • SuperSecretPassword • kix

2. Set up Database Data Sources

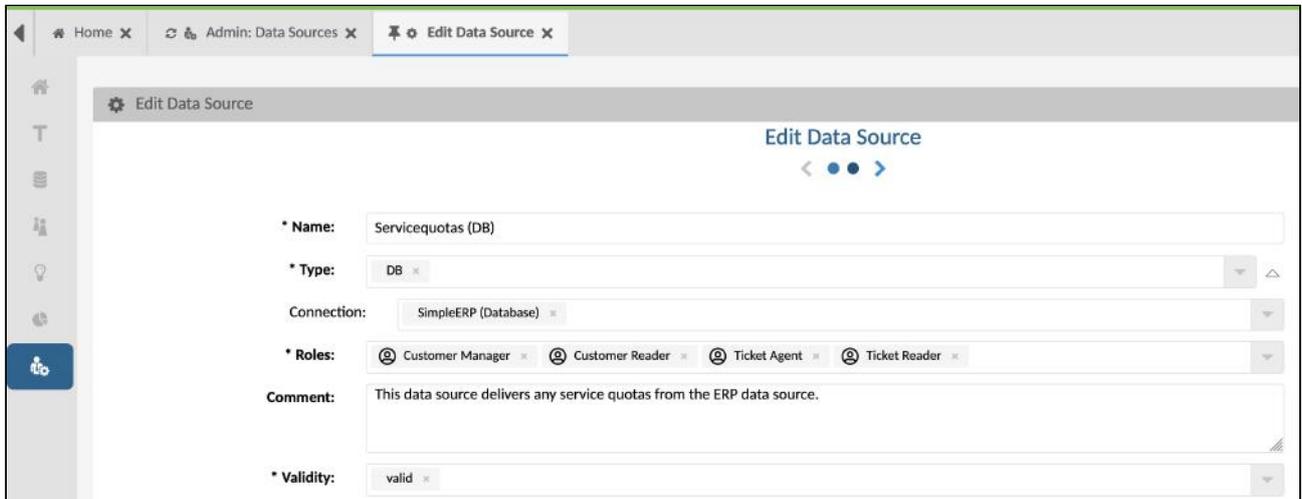
A database data source consists of general information and two parameterizable SELECT statements, which are assigned to a database connection. Both queries are not limited to a single database table/view, but can include all available database tables/views. Special restrictions, conditions or unions ("JOIN") can be defined directly in the queries.

Database data sources are set up and configured in the explorer of the admin module under *Connect > Data sources*.

For the following description it is assumed that a simple database with the "Servicequotas (DB)" table is used as the data source and that the table has the following structure:

Table/View "servicequotas"				
Column	Type	Collation	Nullable	Default
id	integer			
customer_no	character varying(32)			
customer_name	character varying(255)			
comments	character varying(512)			
quota	integer			
startdate	timestamp with time zone			
enddate	timestamp with time zone			

Step 1 - Header Data



The screenshot shows the 'Edit Data Source' configuration page. The fields are as follows:

- Name:** Servicequotas (DB)
- Type:** DB
- Connection:** SimpleERP (Database)
- Roles:** Customer Manager, Customer Reader, Ticket Agent, Ticket Reader
- Comment:** This data source delivers any service quotas from the ERP data source.
- Validity:** valid

Fig.: Set up data source - step 1

Parameter	Description	Example
Name	Name of data source	"Servicequotas (DB)" → apiResourceName: "servicequotas_DB" (s. Info)
Type	Data source type The selection list depends on the installed additional modules.	DB
Connection	Name of the data source to be used (selection list)	"Simple ERP (Database)"
Roles	Restricts the use of the data source based on KIX permission roles. Only users with the role assignments selected here can view the data from the KIX data source and use it within the scope of their authorizations.	Customer Manager, Customer Reader, Ticket Agent, Ticket Reader
Comment	Optional comment	"This data source provides any performance quotas from the ERP data source."

Parameter	Description	Example
Validity	<ul style="list-style-type: none"> • valid: The data source including the data related to it can be viewed and used in KIX. • (temporarily) invalid: The data source is inactive. No data specified in the data source is obtained or used in the KIX. 	valid temporarily invalid invalid

i The *apiResourceName* is created from the given name. This is required for addressing the data source via the KIX backend REST API or for integrating the data source into widget or sidebar tables. The *apiResourceName* is formed by:

- Replace uppercase with lowercase
- Replace spaces with underscore "_"
- Removal of invalid characters such as digits or special characters. Underscore "_" is accepted.

The data source can be reached in the REST API via `/datasources/<apiResourceName>`.

Step 2 - Database Queries

When setting up database queries/SELECT statements, a distinction is made between a **list query** ("List Select Statement") and a **detailed query** ("Item Select Statement"). Separate queries are set up for both. The system supports with default values that need to be adjusted. Both specifications are required for a database data source.

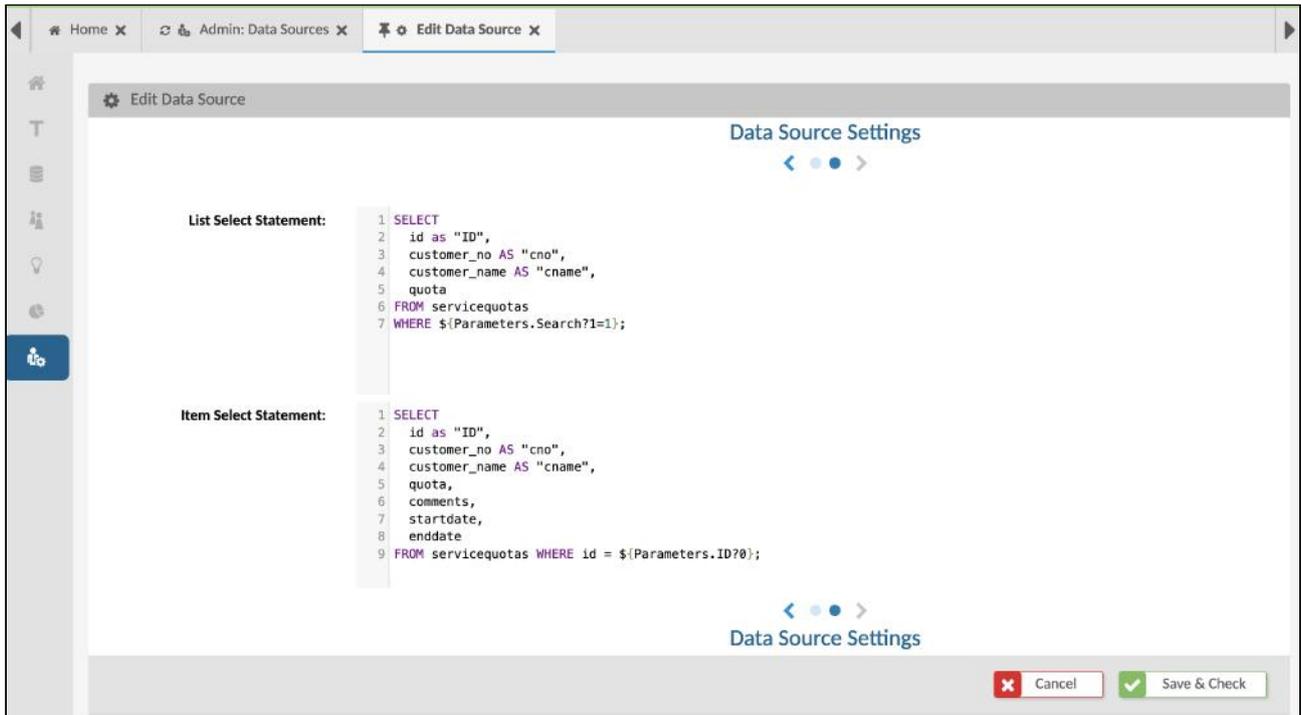


Fig.: Data source setup - step 2

Parameter	Description	Example
List Select Statement	<p>List queries are used to determine a <u>series of entries</u> that are offered for selection in a field input, for example.</p> <p>Typically, these are kept short, contain smaller result sets (e.g. only ID, name, explanation) and have broader conditions that allow user input for search terms (so-called search criteria).</p> <p>The result of a list query is used to display selection values in input fields or to display a single value.</p> <p>⚠ The alias "ID" must always be present.</p> <p>The notation "{Parameters.Search?1=1}" is the placeholder for the search criteria to be applied (e.g. when entering in an autocomplete field). The fallback condition (1=1) ensures syntactically correct SQL if no search criterion is used.</p> <p>The condition "AND startdate > '{Parameters.StartDate?2020-01-01 00:00:00}'" is a fixed condition that is applied in every query. If no date is passed, the query automatically uses "2020-01-01 00:00:00" as the start date.</p>	<pre>SELECT id as "ID", customer_no AS "cno", customer_name AS "cname", quota FROM servicequotas WHERE \${Parameters.Search?1=1}, AND startdate > '\${ Parameters.StartDate?2020-01-01 00:00:00}';</pre>

Parameter	Description	Example
Item Select Statement	<p>Detailed queries are used to display a <u>specific entry</u> from the data source.</p> <p>They are relevant if all the information for a specific entry is required. In this case, several columns are usually referred to and displayed, e.g. in a <u>sidebar</u> as table entries or at a field value as an <u>info pop-up</u>.</p> <p>In the example, the detailed request also includes the concrete value of the quota and the time period in which it applies. Accordingly, more columns are returned in the query.</p> <p>An individual entry is always identified on the basis of the ID column. Accordingly, the condition <code>"id = \${Parameters.ID?0};"</code> must always be specified. If no key value is transmitted for the query, an entry with <code>" id=0 "</code> is searched for. As a rule, this entry does not exist and therefore does not return a result. This ensures syntactically correct SQL.</p>	<pre>SELECT id as "ID", customer_no AS "cno", customer_name AS "cname", quota, comments, startdate, enddate FROM servicequotas WHERE id = \${Parameters.ID?0};</pre>

Renaming data sources

Data sources can be renamed at any time. All existing uses then require manual adjustments. This applies in particular to the configuration of the GUI (SysConfig), ticket templates and actions. The old `apiResourceName` must then be replaced with the new `apiResourceName`.

10.1.6.3 Use of Data Sources

In Dynamic Fields

Scenario:

The quota that applies should be recorded on a ticket. For this purpose, a dynamic field is set up, which is to be set by agents and viewed in detail views with the display of individual information on the quota.

To achieve this, the following must be configured in the KIX:

1. Setup database connection and data source (the database table)

2. Creation of the dynamic field "ServiceQuota"
 1. When entering the field, only entries should be offered that match the search pattern and that became valid after a specified date.
3. Setting up a ticket action "(Set) Service Quota"
4. Configuration of the GUI to display the dynamic field in the ticket details

1. Set up database connection and data source (the database table)

It is assumed that the "Servicequotas (DB)" [data source](#) (see page 267) described above has already been set up.

2. Creation of the dynamic field "ServiceQuota" (menu *System > Dynamic Fields*)

- Name: "ServiceQuota"
- Label: "Service Quota"
- Field Type: DataSource
- Object type: "Ticket"
- SSP-Visible: no
- Configuration:
 - Data Source: "Service Quotas (DB)"
 - Display Pattern: "<quota> (<cname>)"
 - Default Display Columns
 - 1: "quota" / "quota"
 - 2: "cname" / "org-name"
 - 3: "comments" / "comments"
 - 4: "start date" / "From"
 - 5: "enddate" / "until"
 - Cache TTL: 0
 - Count Mins: 1
 - Count Max: 1

3. Set up a ticket action "(set) Service Quota" (menu *Ticket > Actions*)

- Step 1 - Promotion Information
 - Reference object: Ticket
 - Name: "ServiceQuota"
 - Label: "Service Quota"
 - Context of use: agent
 - User input required: yes
- Step 2 - Filters
 - Roles: Ticket Agent
- Step 3 - Pre-Actions
 - empty
- Step 4 - Input Fields

- "Service Quota" - set in mask
- "Advanced options" are defined for limiting the offered selections by searching for the org. name:
 - (1) add information about the "filter" to the "LOADINGOPTIONS" section.
"<SEARCH_VALUE>" serves as a placeholder for the search pattern entered later by the user.
 - (2) add information about the "query" to the "LOADINGOPTIONS" section.
as a result, only those entries are offered whose start date is no earlier than April 23, 2022. Placeholders such as <KIX_TICKET_DynamicField_SomeDateTimeField> or <SEARCH_VALUE> can also be used.

```
{
  "option": "LOADINGOPTIONS",
  "value": {
    "expands": [],
    "filter": [
      {
        "filterType": "OR",
        "operator": "LIKE",
        "property": "cname",
        "type": "STRING",
        "value": "<SEARCH_VALUE>"
      }
    ],
    "query": [
      [ "StartDate" , "2022-04-23 00:00:00" ]
    ],
    "includes": [],
    "limit": null,
    "sortOrder": null
  }
},
```

- Step 5 - Post Actions
 - empty

4. Display of the dynamic field in the ticket details

- In Explorer, navigate to *System > SysConfig* and, if necessary, further to *GUI Configuration*.
- Find the "ticket-details-info-card" key and include the "ServiceQuota" dynamic field there (see also *Displaying Dynamic Field Values*). You can use the code block below.
- Then reload the frontend.

Snippet ticket-details-info-card

```
{
  "componentId": "dynamic-field-value",
  "componentData": {
    "name": "ServiceQuota"
  },
  "conditions": [
    {
      "property": "DynamicFields.ServiceQuota",
      "operator": "NE",
      "value": null
    }
  ]
},
```

The "ServiceQuota" dynamic field is now included in the ticket details so that its value can be displayed. You can also integrate the dynamic field into other views (see Configuration examples of GUI configuration).

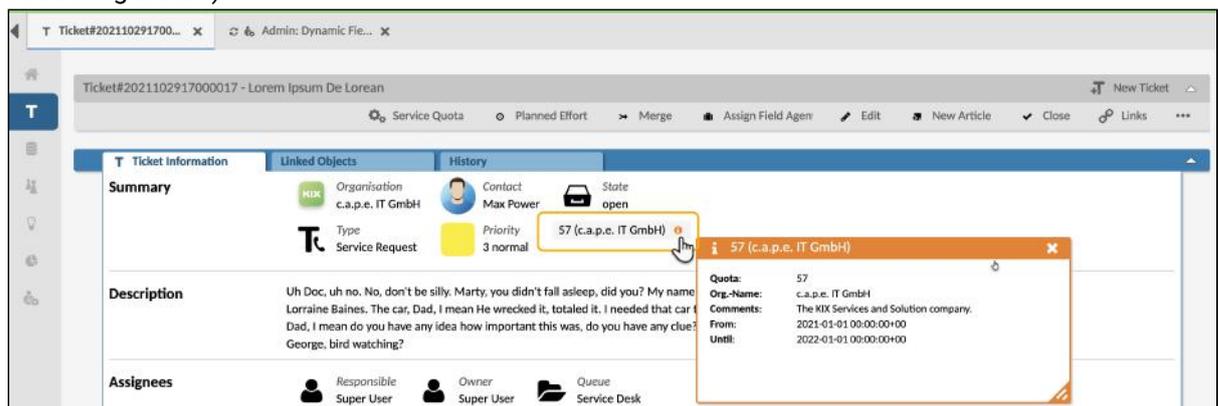
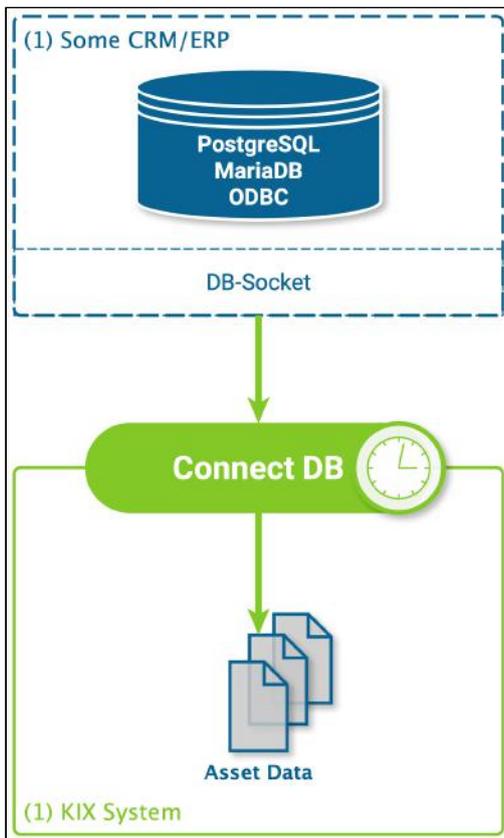


Fig.: Display of the "ServiceQuota" dynamic field in the ticket details



In sidebar tables

Scenario:

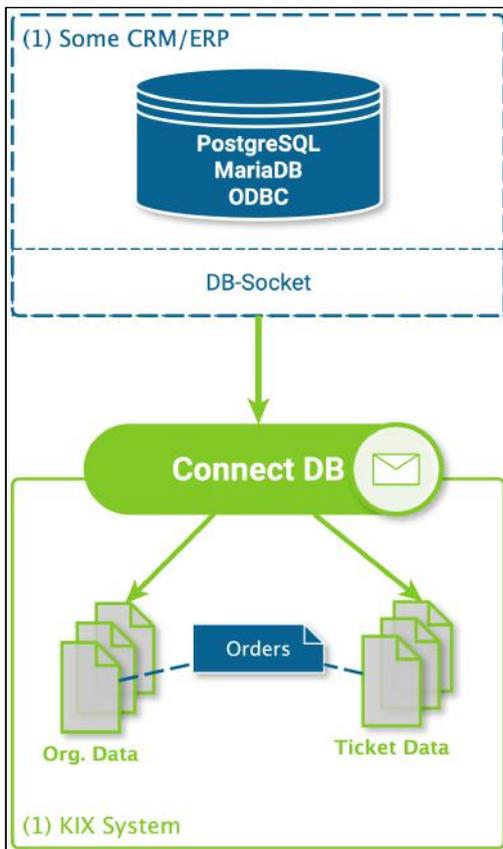
All orders for an organization should be able to be viewed on every ticket of the organization. The customer numbers that are also used in the ERP for customer organizations are known in KIX.

To achieve this, the following must be configured in the KIX.

1. Setting up the database connection and the data source (the database table)
2. Setting up the "Order List" sidebar widget in the ticket detail view

For the following description, it is assumed that a simple database with the "orders" and "customers" tables is used as the data source. It is further assumed that the tables have the following structure:

Table/View Orders				
erp=> \d orders;				
		Table "public.orders"		
Column	Type	Collation	Nullable	Default
-----+				
id	integer		not null	generated always as
identity				
customer_id	integer			
order_number	character varying(255)		not null	
description	character varying(512)		not null	
order_date	date		not null	
...				
erp=> \d customers;				
		Table "public.customers"		
Column	Type	Collation	Nullable	Default
-----+				
id	integer		not null	generated always as
identity				
customer_name	character varying(255)		not null	
customer_no	character varying(32)		not null	
addr_street	character varying(255)			
addr_city	character varying(128)			
addr_zip	character varying(16)			
addr_country	character varying(128)			
url	character varying(255)		not null	
comments	character varying(512)		not null	
...				
erp=>				



1. Database connection and data source setup

It is assumed that the "SimpleERP (Database)" [database connection \(see page 264\)](#) described above has already been set up.

Navigate to *Connect > Data Sources* and do the following configuration to set up the data source:

- Name: "OrderList (DB)"
- Type: DB
- Connection: "SimpleERP (DB)"
- Roles: "Customer Manager", "Customer Reader", "Ticket Agent", "Ticket Reader"
- Comment: -
- Validity: valid

- List Select Statement:

List Select Statement

```
SELECT
  o.id AS "ID",
  c.customer_no,
  o.order_number AS "OrderNumber",
  o.order_date AS "Date"
FROM
  orders o LEFT JOIN customers c ON (o.customer_id = c.id)
WHERE ${Parameters.Search?1=0};
```

The fallback WHERE condition "1=0" means that no entries are listed if no valid search query is submitted.

- Item Select Statement:

Item Select Statement

```
SELECT
  o.id AS "ID",
  c.customer_no,
  o.order_number AS "OrderNumber",
  o.description AS "Description",
  o.order_date AS "Date"
FROM
  orders o LEFT JOIN customers c ON (o.customer_id = c.id)
WHERE o.id = ${Parameters.ID?0};
```

2. Setting up the "Order List" sidebar widget in the ticket detail view

- Under *System > SysConfig > GUI Configuration* open the key "ticket-details"
- Add the following code block to the " `sidebars` " array:

```

CONFIGKEYHERE

{
  "instanceId": "ticket-details-datasource-orderlist",
  "configuration": {
    "id": "ticket-details-datasource-orderlist",
    "name": "Order List from DataSource",
    "type": "Widget",
    "widgetId": "table-widget",
    "title": "Translatable#Order List",
    "actions": [],
    "subConfigurationDefinition": null,
    "configuration": {
      "id": "ticket-details-datasource-orderlist-widget",
      "name": "orders from data source",
      "type": "TableWidget",
      "objectType": "DataSourceRequestItem",
      "sort": [],
      "subConfigurationDefinition": null,
      "configuration": {
        "id": "ticket-details-datasource-orderlist-table",
        "name": "orders from datasource",
        "type": "Table",
        "objectType": "DataSourceRequestItem",
        "loadingOptions": {
          "filter": [
            {
              "filterType": "OR",
              "operator": "LIKE",
              "property": "c.customer_no",
              "type": "STRING",
              "value": "<KIX_ORG_Number>"
            }
          ]
        },
        "sortOrder": null,
        "query": null,
        "limit": 35
      },
      "specificLoadingOptions": {
        "dfName": null,
        "apiResourceName": "orderlist_db"
      }
    },
    "displayLimit": 15,
    "tableColumns": [

```

```

    {
      "name": null,
      "property": "OrderNumber",
      "showText": true,
      "showIcon": true,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 250,
      "sortable": true,
      "filterable": true,
      "hasListFilter": false,
      "dataType": "STRING",
      "resizable": true,
      "defaultText": "",
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    },
    {
      "name": null,
      "property": "Date",
      "showText": true,
      "showIcon": true,
      "showColumnTitle": true,
      "showColumnIcon": false,
      "size": 100,
      "sortable": true,
      "filterable": true,
      "hasListFilter": true,
      "dataType": "STRING",
      "resizable": true,
      "defaultText": "",
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    }
  ],
  "headerHeight": 1.75,
  "rowHeight": 1.75,
  "emptyResultHint": "Translatable#0 orders found.",
  "fixedFirstColumn": false
},
"showFilter": false,
"shortTable": false,
"predefinedTableFilters": [],
"cache": false,
"resetFilterOnReload": true
},
"minimized": false,
"minimizable": true,
"icon": "fas fa-shopping-cart",
"contextDependent": false

```

```

    },
    "size": "large",
    "permissions": []
  },

```

Tip

This example shows the use of fixed parameters to restrict a list request. Variable (changeable by the user) input values in dialogs can also serve as parameters. This allows dependent entries to be mapped across multiple fields, e.g. "Product line → Product → Model" or "Order → Order item".

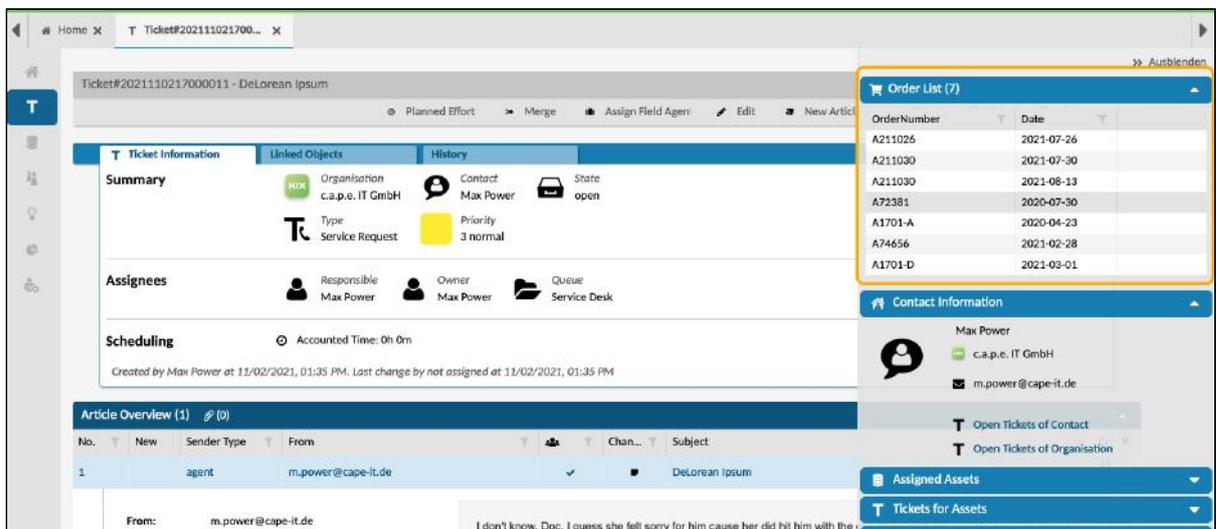


Fig.: "Order List" sidebar widget in the ticket details

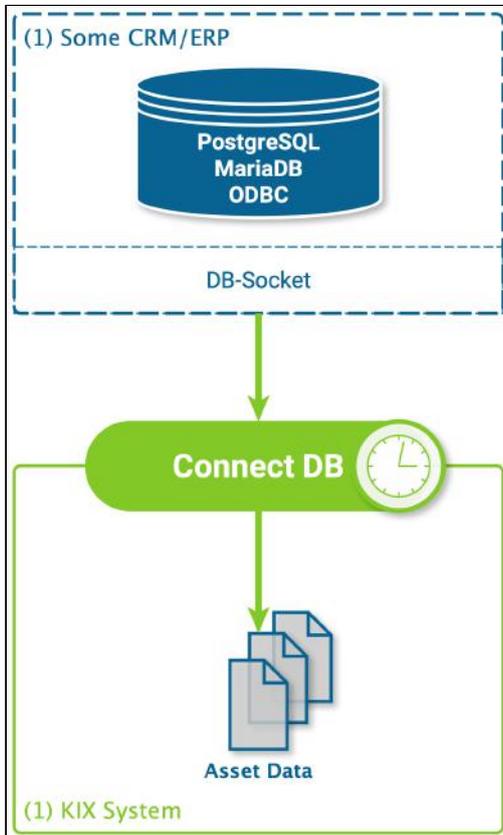
In Macro Actions

Scenario "Inventory Asset Management"

In the KIX asset database there is an asset class "Quota" in whose entries the number of service calls included per customer and service contract is described. The information on how many calls are permitted is available in the ERP in a simple database table with the columns customer number, customer name, calls, etc. This data should be imported from the ERP database into the KIX asset database and kept up to date.

To achieve this, the following must be configured in the KIX.

1. Creation of the asset class "Quota"
2. Setting up the database connection and the data source (the database table)
3. Setup of the automation job





Extract source data

```
kix=# SELECT * FROM servicequotas;
```

id	number		name		comments	quota
startdate		enddate				
1	MY_ORGA	My Organisation				100
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
3	MAMU	Mustermann GmbH				60
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
4	TWD	Torchwood Ltd.				39
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
5	FKLHZ	Fackelholz AG				36
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
6	capeIT	c.a.p.e. IT GmbH				71
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
7	HRMSHL	Harmony Shoal				45
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
8	ITKOM	IT Kommunikations GmbH				43
2021-01-01 00:00:00+00		2022-01-01 00:00:00+00				
...						

1. Creation of the "Quota" asset class

As a prerequisite, an asset class "Quota" must be set up. For this purpose, the following simple asset class definition is assumed (see also "Creating an asset class").

```

Asset class "Quota"

[
  {
    Key          => 'OrgNumber',
    Name         => 'Org. Number',
    Searchable   => 1,
    CustomerVisible => 1,
    Input        => {
      Type => 'Text',
    },
  },
  {
    Key          => 'Quota',
    Name         => 'Quota',
    Searchable   => 1,
    CustomerVisible => 1,
    Input        => {
      Type => 'Text',
    },
  }
];

```

2. Setup database connection and data source

It is also assumed that the "SimpleERP (Database)" [data connection](#) (see page 264) described above and the "Servicequotas (DB)" [data source](#) (see page 267) have already been set up.

3. Setup automation job

The task of the automation job is to query all service quotas and create or update an asset in the asset database for each entry, if it exists. In order to be able to check whether an asset already exists, the name of a service quota is assumed to be unique within the asset class. Since the information from the ERP database cannot be inserted directly into an asset structure, it must be prepared beforehand. The result is the following list of macro actions:

1. Action: Get List Of Items From Data Source
2. Action: Loop → Macro type "Synchronization"
 - a. Get Item From Data Source
 - b. XSL transformation
 - c. create or update asset

- "Synchronization" is used as the job type, all other job parameters (name, execution plan, etc.) can be defined individually.

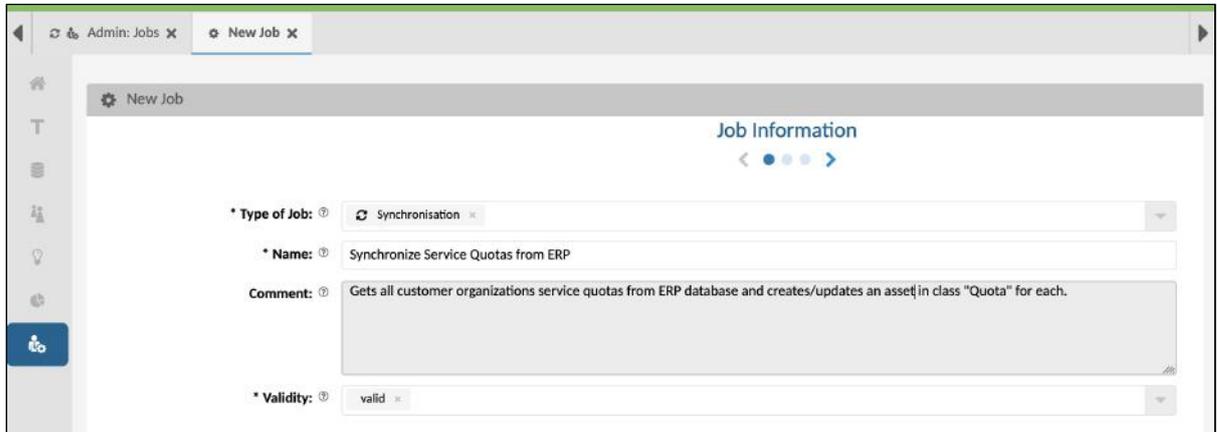


Fig.: Configuration Step 1 - Job Information

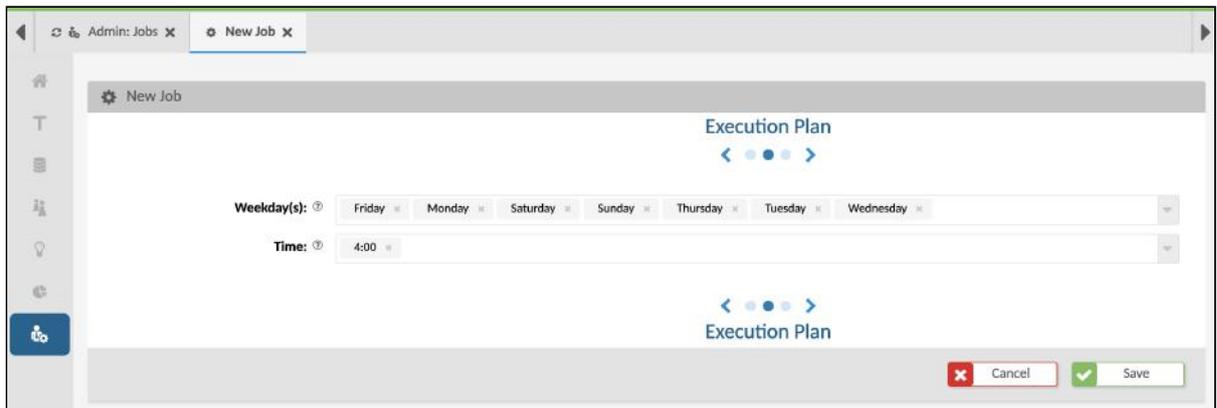


Fig.: Configuration step 2 - execution plan

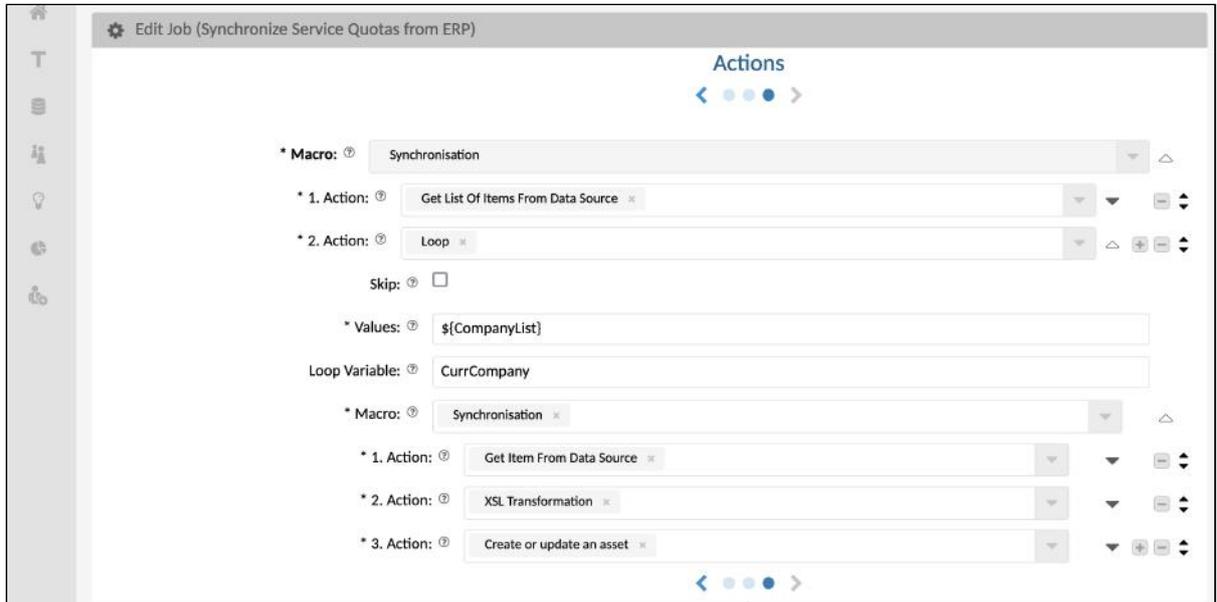


Fig.: Overview of the macro actions to be created subsequently

4. Macro Action 1 - Get List Of Items From Data Source
(Parameters of the action see under: [Connect](#) (see page 238))

⚙️ Edit Job (Synchronize Service Quotas from ERP)

Actions

⏪ ⬅️ ● ● ● ● ➡️ ⏩

* Macro: Ⓜ Synchronisation

* 1. Action: Ⓜ Get List Of Items From Data Source

Result names: Ⓜ

Itemlist: Ⓜ CompanyList

Skip: Ⓜ

* Data Source: Ⓜ Servicequotas (DB)

Search: Ⓜ powered by ace

```
1- {
2-   "Item": {
3-     "AND": [
4-       {
5-         "Field": "customer_no",
6-         "Operator": "LIKE",
7-         "Type": "STRING",
8-         "Value": "%"
9-       }
10-     ]
11-   }
12- }
```

Ln: 1 Col: 1

Parameters: Ⓜ powered by ace

```
1 { }
```

Ln: 1 Col: 1

Debug: Ⓜ

* 2. Action: Ⓜ Loop

✖ Cancel ✔ Save

- Action: "Get List Of Items From Data Source"
- ItemList: CompanyList
- Data Source: "Servicequotas (DB)"

- Search:

```

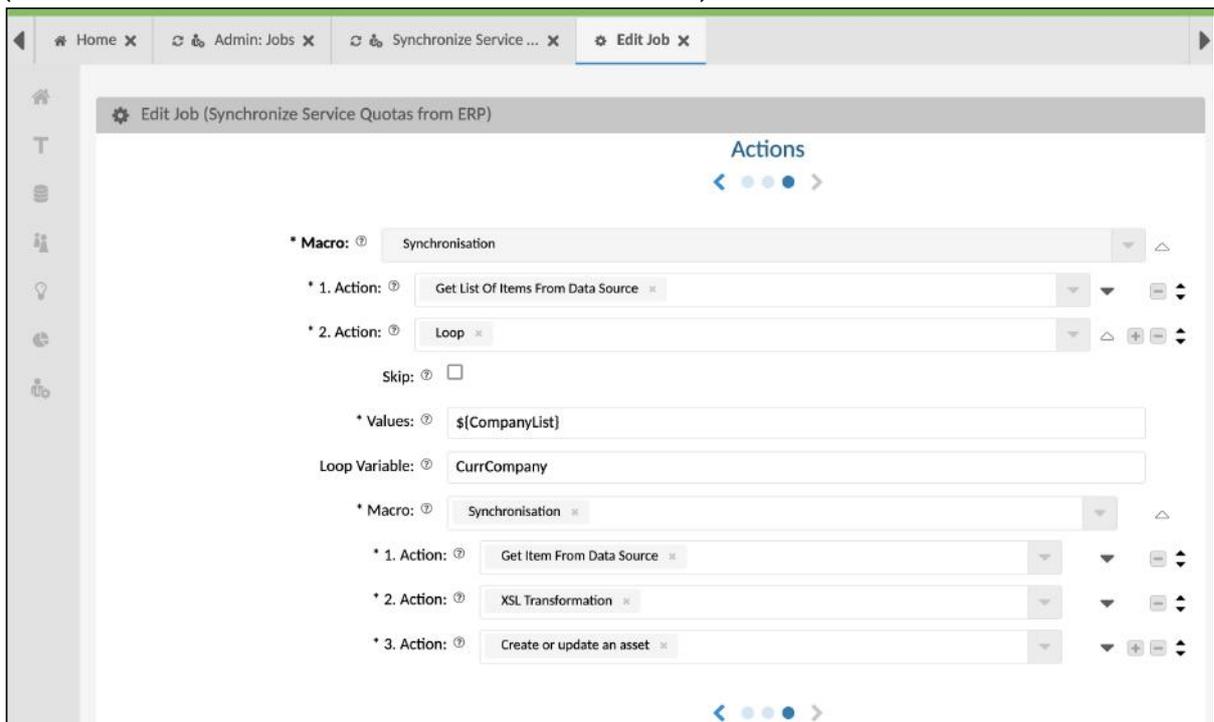
Search

{
  "Item": {
    "AND": [
      {
        "Field": "customer_no",
        "Operator": "LIKE",
        "Type": "STRING",
        "Value": "%"
      }
    ]
  }
}

```

- Parameters
 - {} (empty/no parameters)

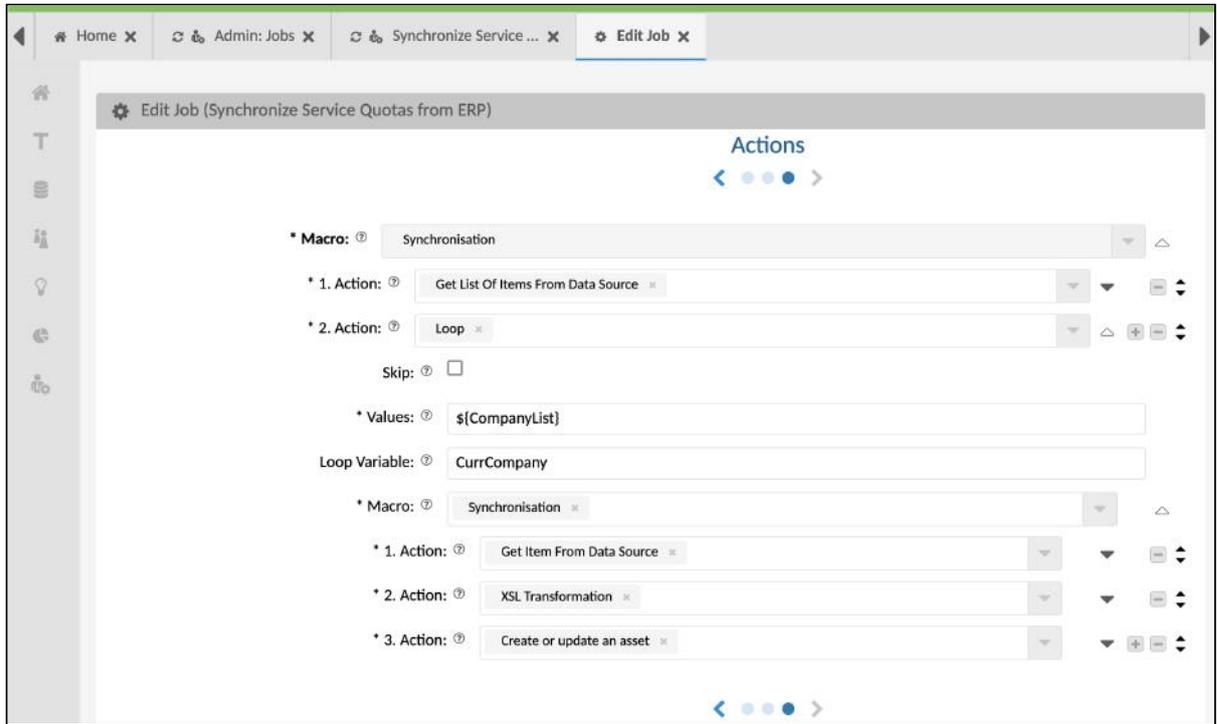
5. Macro Action 2 - Loop → Macrotyp "Synchronization"
(For information on the action see Overview Macro Actions)



- Action: Loop
- Macro: Synchronization

6. Macro Action 2.1 - Get Item From Data Source

(Parameters of the action see under: [Connect](#) (see page 238))

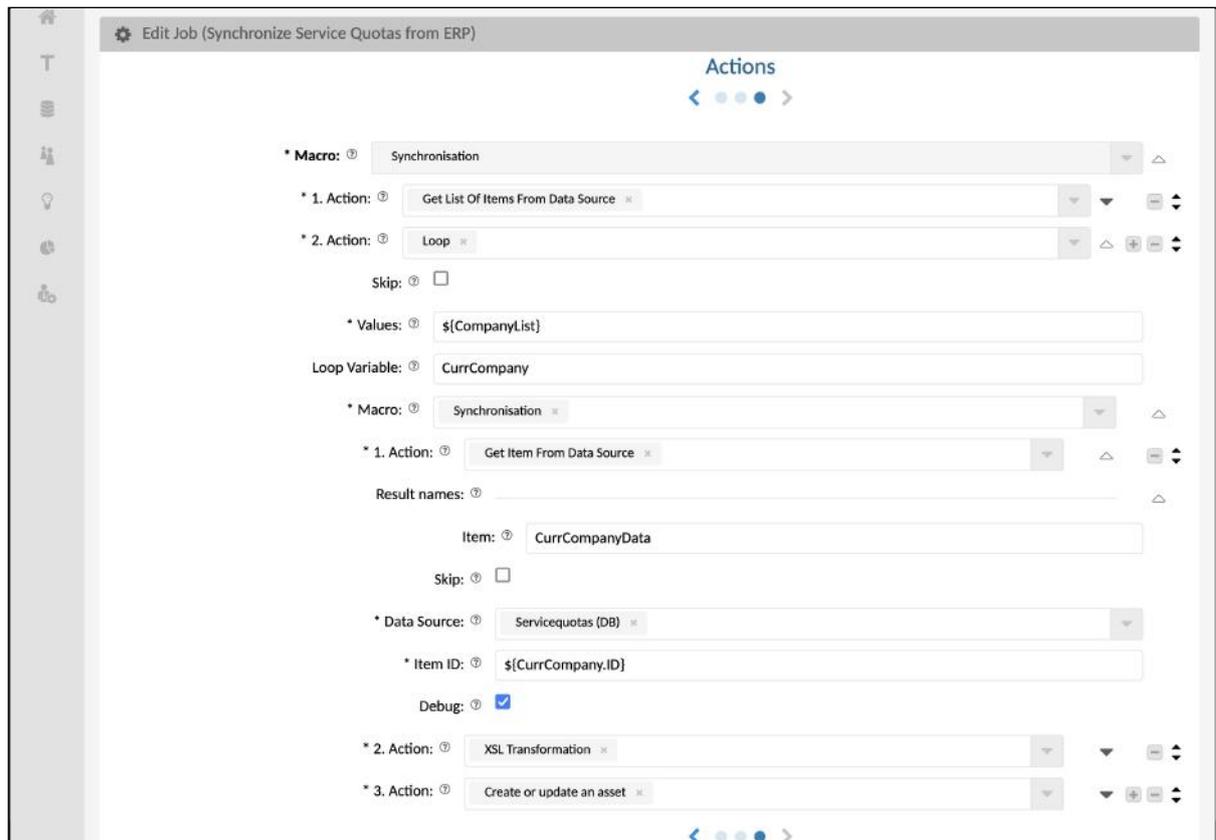


The screenshot shows the 'Edit Job' configuration page for 'Synchronize Service Quotas from ERP'. The 'Actions' section is expanded to show a macro named 'Synchronisation'. This macro contains a 'Loop' action. Inside the loop, the first action is 'Get Item From Data Source'. The 'Values' field is set to '\$(CompanyList)' and the 'Loop Variable' is 'CurrCompany'. Below the loop, there are two more actions: 'XSL Transformation' and 'Create or update an asset'.

- ⚠ Note structure: belongs to Macro under Loop
- Action: Get Item From Data Source
- Item: CurrCompanyData
- Data Source: "Servicequotas (DB)"
- Item ID: `#{CurrCompany.ID}`

7. Macro Action 2.2 - XSL Transformation

(Parameters of the action see under: [Connect](#) (see page 238))



- ⚠ Note structure: belongs to Macro under Loop
- Action: XSL Transformation
- TransformedData: CurrQuotaAssetData
- Data: \${CurrCompanyData}
- XSL Template:

XSLT mapping creation of asset data "Quota"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:KIX="urn:KIX">
<xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="RootElement">
    <root>
      <xsl:variable name="AssetName"><xsl:value-of select="cname"/></xsl:variable>
      <AssetID><xsl:value-of select="KIX:AssetGetIDByClassAndName('Quota',
$AssetName )"/></AssetID>
      <ClassID>93</ClassID> <!-- AssetClassID for "Quota" +++ check ID before
flight !!! -->
      <Version>
```

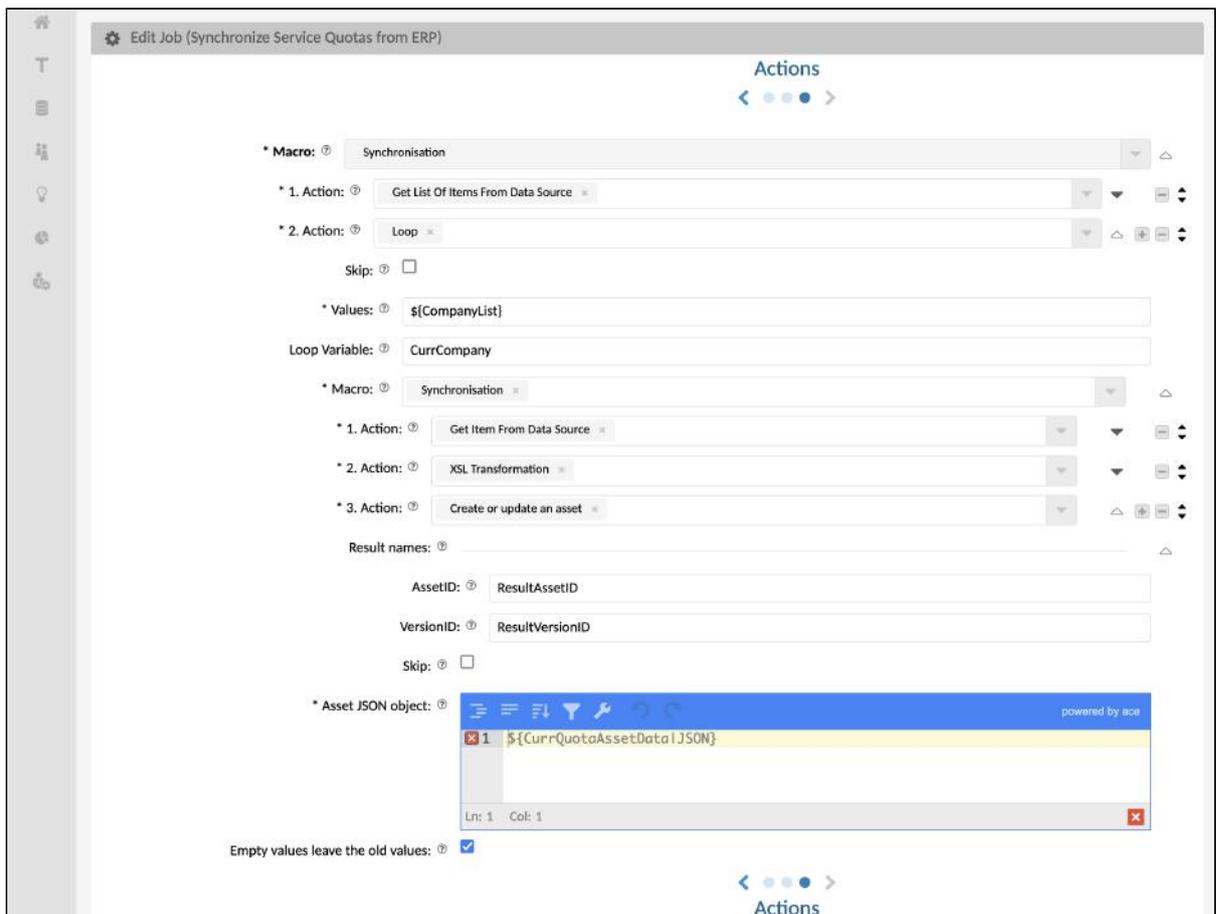
```

<DeplStateID>16</DeplStateID> <!-- Production -->
<InciStateID>1</InciStateID> <!-- Operational -->
<Name><xsl:value-of select="$AssetName"/></Name>
<Data>
  <OrgNumber><xsl:value-of select="cno"/></OrgNumber>
  <Quota><xsl:value-of select="quota"/></Quota>
</Data>
</Version>
</root>
</xsl:template>
</xsl:stylesheet>

```

- Force Array Tags: -
- Suppress Empty: Undefined
- Debug: no

8. Macro Action 2.3 - Create or update asset
(For information on the action, see Overview of macro actions)

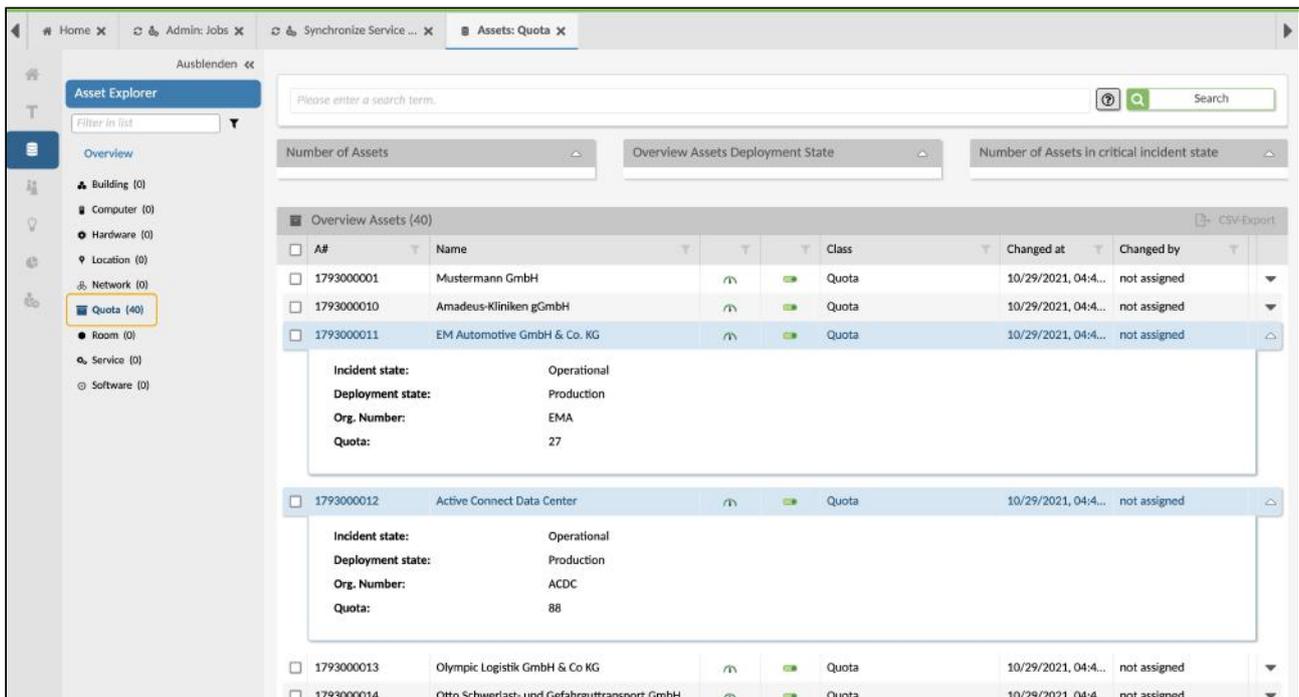


- ⚠ Note structure: belongs to Macro under Loop

- AssetID: ResultAssetID (or empty)
- VersionID: ResultVersionID (or empty)
- Asset JSON Object: `${CurrQuotaAssetData | JSON}`

Result

After a manual or automatic execution of the job, assets are created or updated:



The screenshot displays the 'Assets: Quota' overview in the KIX Admin interface. The interface includes a search bar at the top, a sidebar with a tree view, and a main table of assets. The table lists assets with columns for ID, Name, Class, Changed at, and Changed by. Below the table, detailed information for selected assets is shown, including Incident state, Deployment state, Org. Number, and Quota.

A#	Name	Class	Changed at	Changed by
1793000001	Mustermann GmbH	Quota	10/29/2021, 04:4...	not assigned
1793000010	Amadeus-Kliniken gGmbH	Quota	10/29/2021, 04:4...	not assigned
1793000011	EM Automotive GmbH & Co. KG	Quota	10/29/2021, 04:4...	not assigned
1793000012	Active Connect Data Center	Quota	10/29/2021, 04:4...	not assigned
1793000013	Olympic Logistik GmbH & Co KG	Quota	10/29/2021, 04:4...	not assigned
1793000014	Otto Schwerlast- und Gefahzutransport GmbH	Quota	10/29/2021, 04:4...	not assigned

Incident state: Operational
Deployment state: Production
Org. Number: EMA
Quota: 27

Incident state: Operational
Deployment state: Production
Org. Number: ACDC
Quota: 88

10.1.6.4 Use of the KIX Database

You can also connect KIX's own database. The access data required for this can be determined from the backend configuration - starting from the Docker host:

Determination of DB access data

```
# access KIX backend service...
user@dockerhost:/opt/kix-on-premise/deploy/linux$ docker exec -it kixpro-backend-1
bash

# check currently used database setting in KIX configuration...
I have no name!@4be1ec32c580:/opt/kix$ cat /opt/kix/config/01main | grep Database
'DatabaseHost'    => 'db',
'Database'        => 'kix',
'DatabaseUser'    => 'kix',
'DatabasePw'      => 'Password',
'Database::Type' => 'postgresql',
'DatabaseDSN'     =>
'DBI:Pg:dbname=<KIX_CONFIG_Database>;host=<KIX_CONFIG_DatabaseHost>;",
I have no name!@4be1ec32c580:/opt/kix$

# resulting DSN...
DBI:Pg:dbname=kix;host=db;
```

Hint

Note that the use of KIX's own structures can be affected by updates. If you create your own DB structures (e.g. views), give them a specific prefix, e.g. "acme_orderlists".



10.1.7 Connect Opsi

Connect Opsi is an add-on module to KIX Pro. It offers the possibility to record asset data in KIX on the basis of the inventory solution opsi (<https://www.opsi.org>). The information available in opsi can thus be used in the KIX Service Management System.

KIX provides a further, separate asset class "Opsi Device" for this purpose. The device data to be imported in it are obtained via the opsi bConnect interface. The synchronisation is done by the job "Opsi - Device Sync". Both the asset class and the job are implemented in KIX when opsi synchronisation is activated.

Content on this page:

- [Requirements](#) (see page 295)
- [Use](#) (see page 295)
 - [Activate opsi synchronisation](#) (see page 296)
 - [Reset or change opsi synchronisation](#) (see page 297)
- [Adjustments](#) (see page 298)

10.1.7.1 Requirements

To use Connect Opsi, HTTP/S access to opsi is required. For this, the FQDN and, if applicable, the port number must be specified. When using HTTPS, only correct SSL certificates may be used. The use of invalid or self-created certificates can negatively influence the function of the interface.

Furthermore, the user name and password of an opsi user authorised to log in are required.

10.1.7.2 Use

The deployment of Connect Opsi is done

- for KIX On Premise: by restarting the KIX stack using "start.sh".
- for KIX Cloud: after feedback from KIX Support.

The configuration and activation of the plug-in takes place after installation in the admin area in the Connect Opsi setup.

Welcome to Connect Opsi

The plugin "Connect Opsi" enables KIX to automatically synchronize asset data from Opsi into the CMDB. The plugin provides a pre-configured automation job for this task, as well as a new asset class "Opsi Device".

Activate the plugin

To activate the plugin, please configure the connection to the RPC interface of your Opsi server in the input form below and click the button "Activate".

Reset the plugin

When the plugin is active, the pre-configured job can be reset to the default by clicking the button "Reset to Default".

Change the RPC settings afterwards

After the plugin got activated, the RPC settings can still be changed. Just use the input form below and click the button "Save".

RPC Settings

* RPC User:

* RPC User Password:

* RPC URL:

Fig.: opsi Setup

Activate opsi synchronisation

To activate the synchronisation, enter the opsi URL (e.g. <https://YourOpsiServer.example.com:4447/rpc>) as well as the user name and password in the Admin module under Opsi > Welcome (see requirements). Then click on "Activate". The configuration is saved and creates the automation job "Opsi - Device Sync".

For a first data import from opsi we recommend to execute the job "Opsi - Device Sync" manually afterwards (Admin Module under KIX > Automation > Jobs). This may take a few minutes depending on the content of the opsi data source. If the data import is to be executed periodically in the future, assign an appropriate schedule to the job under "Execution Schedule", e.g. every day 03:00 (see Creating or Editing a Job). Do not change the rest of the job configuration! After that, the configuration is complete.

Type	Name	Event Based	Time Based	Number of Actions	Last executed at	Comment
Ticket	Anonymisation	✓		9		Please check the Admin Manual for how
Ticket	Auto Set Planned Effort (Incident)	✓		1		Sets the planned effort of a ticket on type
Ticket	Auto Set Planned Effort (Service Request)	✓		1		Sets the planned effort of a ticket on type
Ticket	Create FAQ Suggestion	✓		2		Create a new faq article based on a ticket
Ticket	Customer Response - reopen from pending	✓		1		Reopens ticket from pending by customer
Ticket	KIX Field Agent - Mobile Processing Rejected	✓		2		This job resets owner and lock state of a
Synchronisation	Opsi - Device Sync			2	2022-11-09 08:11:22	
Reporting	Periodic Reports		✓	1	2022-11-09 08:45:02	Executes all reports that should be create
Ticket	Put affected assets back into operational state	✓		1		Sets the incident state "Operational" for a
Ticket	Put affected assets into incident state	✓		1		Sets the incident state "Incident" for all a
Ticket	SLA First Response Time Fulfillment	✓		1		Sets First Response SLA fulfillment timest
Ticket	SLA Solution Time Fulfillment	✓		1		Sets Solution SLA fulfillment timestam
Ticket	SLA Time Suspension	✓		2		Suspends or resumes ticket SLA.
Ticket	TicketMerge	✓		1		Merges tickets.

Fig.: The opsi synchronisation job

Reset or change opsi synchronisation

If you have made adjustments to the configuration of the job, for example, to synchronise specific, extended information, you can reset the default configuration to the delivery state at any time using the "Reset to default" button. This removes the specific adjustments.

After activating the synchronisation, you can still change the settings for the opsi interface, for example, to store a changed URL or new login data. To do this, make the necessary changes in the Connect Ops Setup and click on "Save". The changes are active with the next job execution.

Welcome to Connect Ops

The plugin "Connect Ops" enables KIX to automatically synchronize asset data from Ops into the CMDB. The plugin provides a pre-configured automation job for this task, as well as a new asset class "Ops Device".

Activate the plugin

To activate the plugin, please configure the connection to the RPC interface of your Ops server in the input form below and click the button "Activate".

Reset the plugin

When the plugin is active, the pre-configured job can be reset to the default by clicking the button "Reset to Default".

Change the RPC settings afterwards

After the plugin got activated, the RPC settings can still be changed. Just use the input form below and click the button "Save".

RPC Settings

* RPC User:

* RPC User Password:

* RPC URL:

Fig.: Resetting the configuration



10.1.7.3 Adjustments

Please contact us if you have any requests for adjustments to the data to be used in the asset class "Ops Device" or for separate consideration in different asset classes.

10.1.8 Connect Webservice

Content on this page:

- [KIX as a provider - specific endpoints \(see page 299\)](#)
 - [Overview of the endpoints \(see page 300\)](#)
 - [Setting up an endpoint \(see page 301\)](#)
 - [1. Definition of an endpoint \(see page 301\)](#)
 - [2. Endpoint actions and responses \(see page 303\)](#)
 - [Access to received parameters \(see page 304\)](#)
 - [Use of an endpoint \(see page 306\)](#)
- [KIX as a requester - use of webhooks \(see page 308\)](#)
 - [Requirements \(see page 308\)](#)
 - [Advanced Macro Actions \(see page 308\)](#)
 - [Webhook Extended \(see page 308\)](#)
 - [Parameter \(see page 308\)](#)
 - [Example: Create a Confluence page \(see page 310\)](#)
 - [Example: Create Redmine Issue \(see page 313\)](#)
 - [Post Attachment as Form Data \(see page 317\)](#)
 - [Parameter \(see page 317\)](#)
 - [Variables and advanced filters \(see page 319\)](#)
 - [Application in macros \(see page 319\)](#)
 - [Extended operations/filters \(see page 319\)](#)

10.1.8.1 KIX as a provider - specific endpoints

Menu	Connect > Endpoints
-------------	---------------------

In addition to the universal REST API from KIX, the Connect Webservice add-on can be used to set up specific endpoints with specific functions or, in contrast to the REST API, with extended functions.

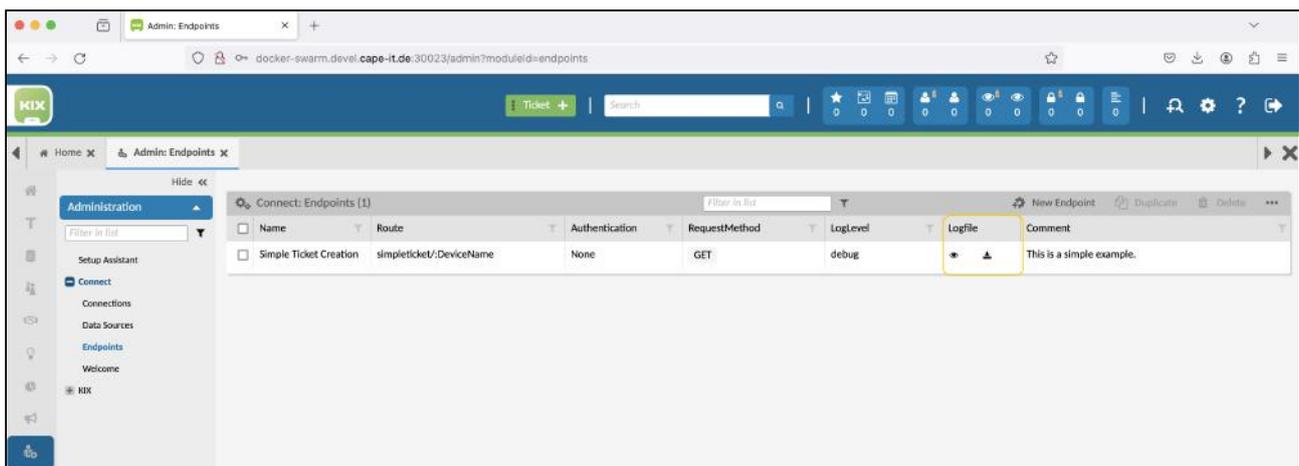
Application scenarios can be:

- Interface for updating contact/organisations
 - The triggering system has no knowledge of the KIX-internal identifiers or may only provide selected information. For example, only the email address of a contact or the number of the organisation may be known, but a changed address, telephone number or surname should be recorded. The task of the endpoint is to determine and update the appropriate entry for the transmitted identifier.

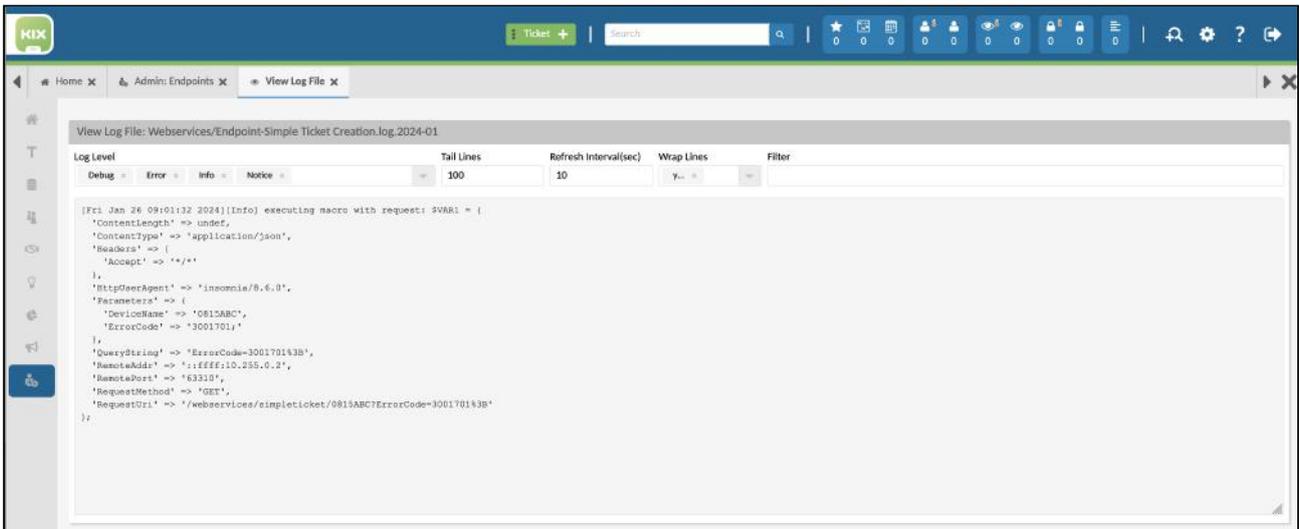
- Interface for updating asset data
 - Free memory or the last logged-in user can be reported directly to KIX without having to wait for KIX to synchronise the data. Critical information can thus be made available on the asset more quickly and easily.
- Interface for monitoring messages
 - IT monitoring services can report faults directly to KIX. The message contains, for example, a device identifier and the current error status. With the extended macro actions of Connect Webservice, not only can a ticket be created, but current information on an existing, open ticket can also be added in the case of repeated messages. A connection by e-mail is not required.
- Interface for device data or creation of tickets from incomplete entries
 - Technical monitoring tools or devices can report error statuses to a defined URL. As a rule, only the device number and an error code are transmitted. The task of the endpoint is to complete the transmitted data so that a ticket with all the necessary information can be created or an existing process for the device can be updated.
- Adding information to tickets
 - Information ("flag") can be added directly to a ticket using a simple URL. Clicking on the URL sends a response without calling up the self-service or agent portal.
 - A concrete example is low-threshold approvals or rejections. The recipient of an email with such a URL can click on it and thus directly reject or approve a request.
 - ⚠ **Important:** Sending the URL in an email harbours uncertainties. Anyone with access to the URL could trigger an approval. This procedure should therefore only be used for limited risks.

Overview of the endpoints

An overview of all configured endpoints can be found in the Admin module under *Connect > Endpoints*. The overview also contains access to the respective endpoint-specific log.



The log can be viewed in the user interface or downloaded.



Setting up an endpoint

You can set up your own endpoints in the Admin module under *Connect > Endpoints > New Endpoint*. The setup takes place in two steps:

1. Definition of the endpoint
2. Definition of the action and responses to be executed

i An endpoint is set up below using a simplified example. We use the above-mentioned application scenario "Creation of tickets from incomplete entries". The example is for illustrative purposes only and makes no claim to productive usability.

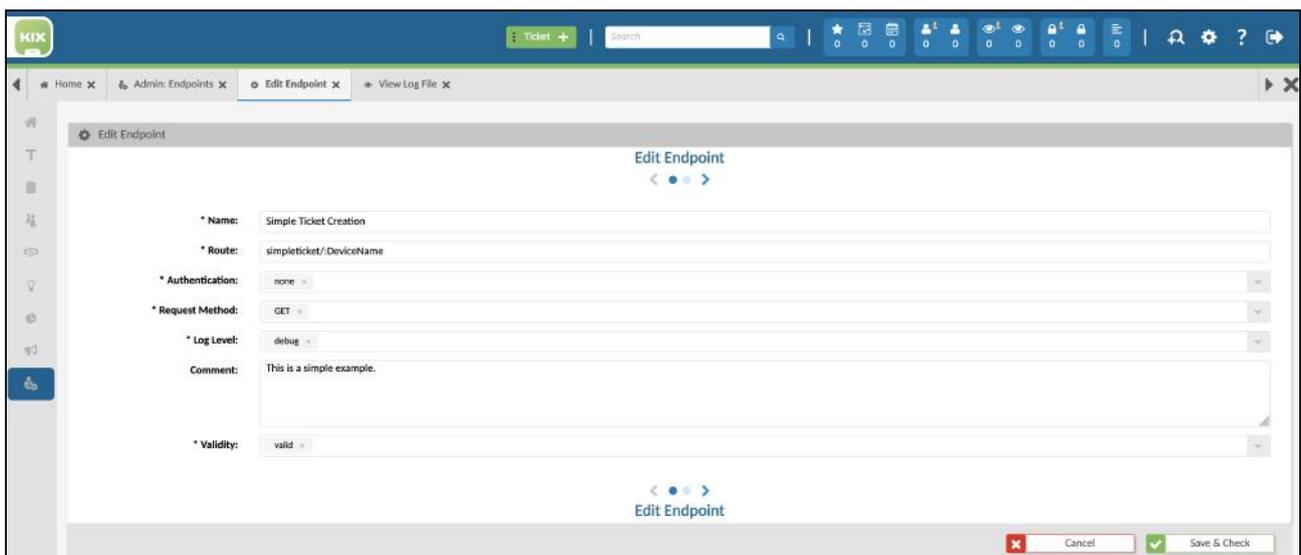
1. Definition of an endpoint

The following information is stored in the definition of the endpoint:

Parameter	Description	Example
Name	Name of the endpoint	Simple Ticket Creation
Route	The endpoint as a subpoint of the REST API /api/v1/webservices/<EndpointRoute>	simpleticket/:DeviceName

Parameter	Description	Example
Authentication	Defines whether the endpoint requires authentication (currently none available). As a workaround, this can be realised via parameter evaluation in macro processing.	none
Request Methode	Should a proxy server be addressed?	GET POST PUT PATCH DELETE OPTIONS HEAD
Log Level	Depth of detail of the log	error notice info debug
Comment	Brief description of the endpoint, e.g. intended use, such as process assignment or relevant counterparty.	This is a simple example.
Validity	Deactivating/activating the endpoint	valid (temporarily) invalid

The placeholder ":DeviceName" can be used to transmit a parameter value within the URL and without specifying a parameter name. In the example, the "DeviceName" parameter is used to create the ticket. Other parameters are separated as usual when transmitting the URL. This means with a question mark, followed by the parameter name and the value (e.g.: ...webservices/simpleticket/0815ABC?ErrorCode=3001701;)



The screenshot shows the 'Edit Endpoint' configuration page in the KIX Admin interface. The configuration is as follows:

- Name:** Simple Ticket Creation
- Route:** simpleticket/DeviceName
- Authentication:** none
- Request Method:** GET
- Log Level:** debug
- Comment:** This is a simple example.
- Validity:** valid

At the bottom of the form, there are 'Cancel' and 'Save & Check' buttons.

Fig.: Definition of the example endpoint "Simple Ticket Creation"

2. Endpoint actions and responses

Endpoint actions are basically structured like macros in jobs and offer the same options. The macro variable `${Request}` is provided to access the content of a request, such as URL parameters or the content of a POST request. This contains all the details of the request received.

A simple endpoint is an (extended) echo. The endpoint macro only uses the "Set Response" action for this. This macro action allows the definition of the HTTP status code, the HTTP headers and the content to be transmitted. The simplest form is to respond with a status code without any further information.

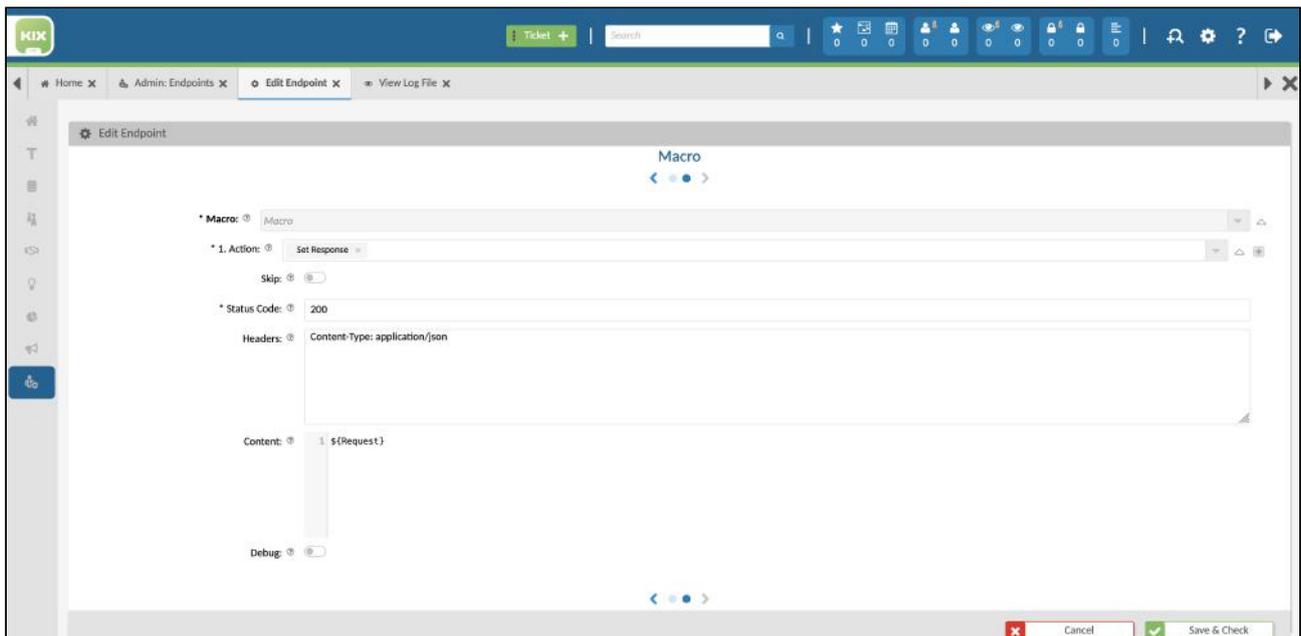


Fig.: Configuration of the "Simple Ticket Creation" endpoint as an echo

If the endpoint is used, the complete request is returned as the response:

Call an echo endpoint (use jq for formatting)

```
shell>$ curl --request GET \
--url 'http://yourkix-api.kix.cloud/api/v1/webservices/simpleticket/0815ABC?
ErrorCode=3001701' \
--header 'Content-Type: application/json' \
| jq
{
  "Parameters": {
    "ErrorCode": "3001701;",
    "DeviceName": "0815ABC"
  },
  "RequestMethod": "GET",
```

```

"ContentType": "application/json",
"HttpUserAgent": "curl/8.1.2",
"RequestUri": "/webservices/simpleticket/0815ABC?ErrorCode=3001701",
"ContentLength": null,
"Headers": {
  "Accept": "*/*"
},
"RemoteAddr": "::ffff:10.255.0.2",
"RemotePort": "63873",
"QueryString": "ErrorCode=3001701%3B"
}

```

Access to received parameters

The contents of the request can be accessed using the macro variable `${Request}`. If the request contains a parameter XYZ, this is retrieved using `${Request.Parameters.XYZ}`. The same applies to the contents of the body of a POST/PUT/PATCH request.

In the following example, a ticket is to be created using the information from the request. The macro definition is adapted for this purpose. The first action is now the call of a macro of the type Ticket, which in turn uses the TicketCreate action and returns its return value as a response to the endpoint call.

Macro configuration:

- 1. Action: Execute Macro
 - Macro: Ticket
 - 1. Action: Create Ticket
 - NewTicketID: NewSimpleTicketID
 - Title: Error Report for \${Request.Parameters.DeviceName} (\${Request.Parameters.ErrorCode})
 - Body:
 - Device Name: \${Request.Parameters.DeviceName}
 - Error Code: \${Request.Parameters.ErrorCode}
 - Contact: admin
 - Channel: note
 - Sender Type: external
 - Priority: 2 high
 - Team: Service Desk
 - Type: Incident
 - 2. Action: Set Response
 - Status Code: 200
 - Headers: Content-Type: application/json
 - Content: {"ID": "\${NewSimpleTicketID}}

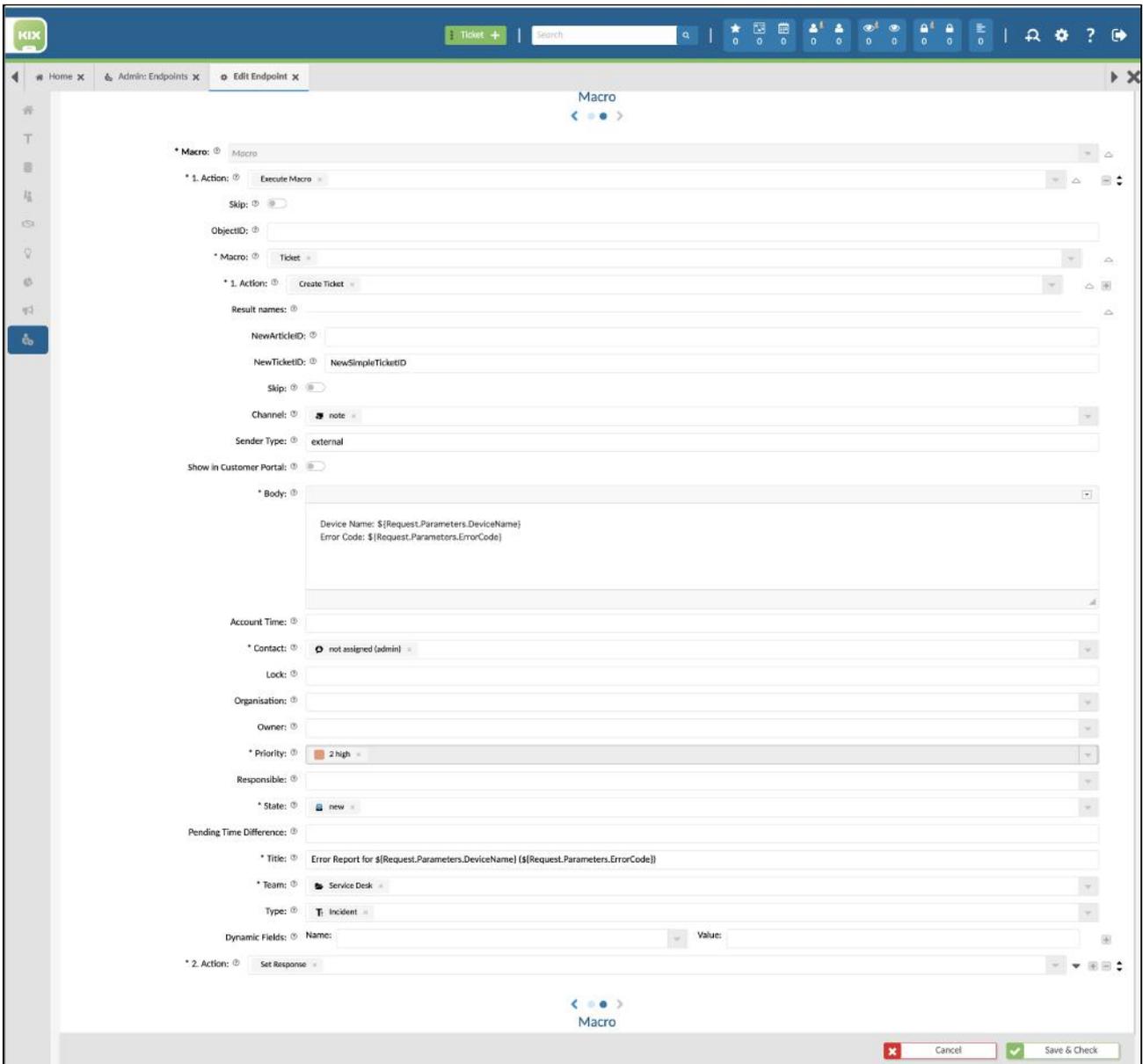


Abb.: Configuration of endpoint macro with parameter access (part 1)
(For a better overview, non-relevant parameters of the "Ticket Create" macro action have been hidden).

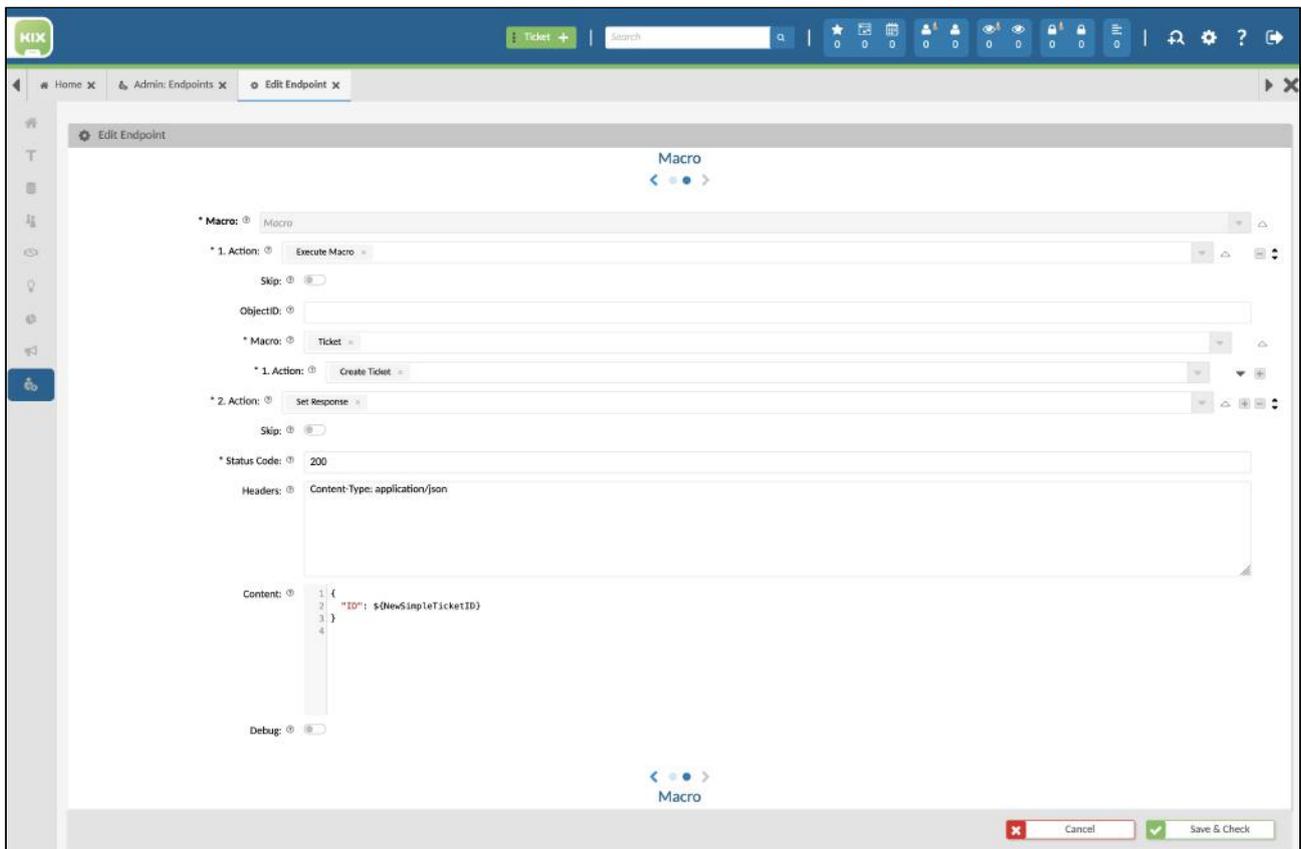


Fig.: Configuration of endpoint macro with parameter access (part 2)

Use of an endpoint

A previously set up endpoint can be called via the REST API under `"/api/v1/webservices/<EndpointRoute>"`. If the REST API can be accessed at <https://yourkix-api.kix.cloud>, for example, and an endpoint with the route `"/simpleticket"` has been set up, this can be called under the URL `"https://yourkix-api.kix.cloud/api/v1/webservices/simpleticket"`. The HTTP method that corresponds to the endpoint configuration must be used. In the example above, the call `"GET https://yourkix-api.kix.cloud/api/v1/webservices/simpleticket/0815ABC?ErrorCode=3001701"` should create a ticket for the device `"0815ABC"` with the error code `"3001701"`.

Calling the simple ticket endpoint (use jq for formatting)

```
shell> curl --request GET \
  --url 'http://yourkix-api.kix.cloud/api/v1/webservices/simpleticket/0815ABC?
  ErrorCode=3001701' \
  --header 'Content-Type: application/json' \
  | jq
{
  "ID": 3
}
```

The result of the call is a new ticket:

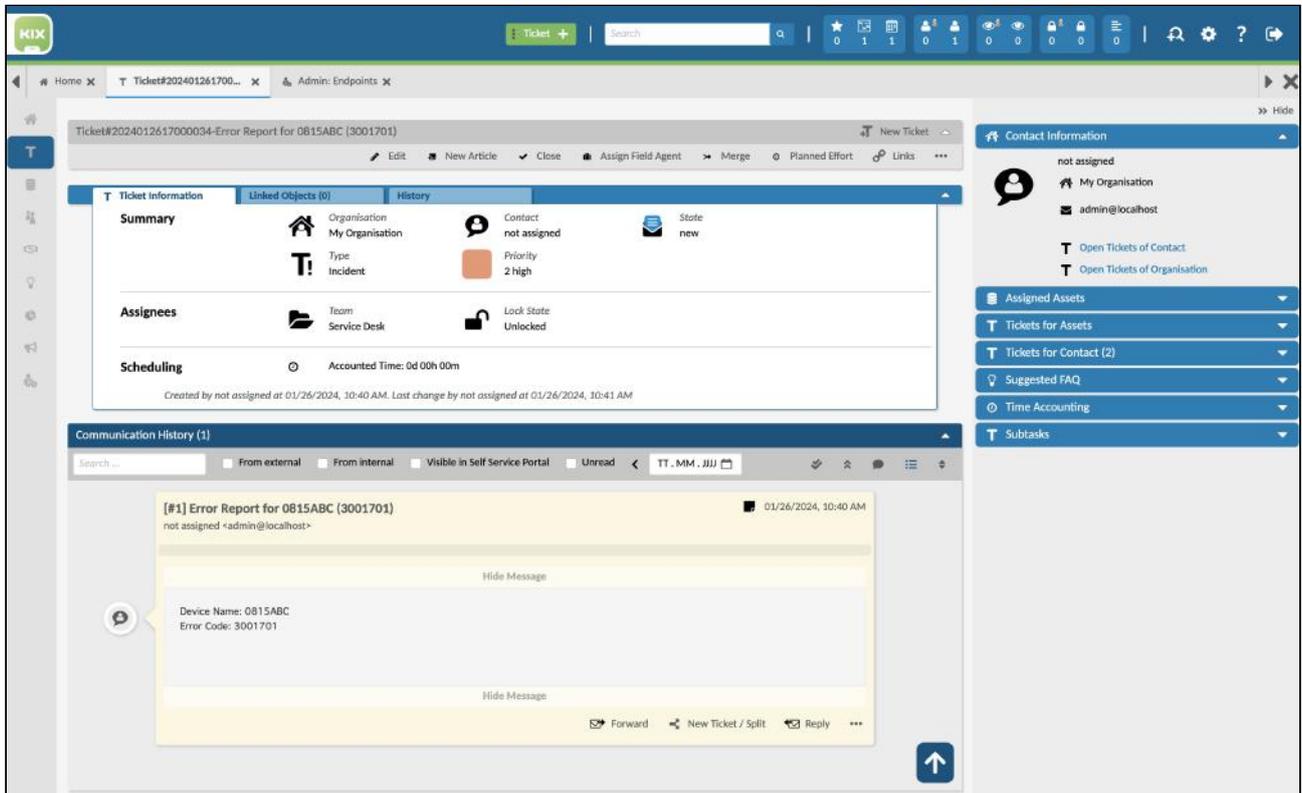


Fig.: New ticket created by the endpoint

Tip

In conjunction with the options of other macro actions such as ObjectGet, XSLTransform, Set Dynamic Field, CreateOrUpdateAsset, CreateOrUpdateContact, CreateOrUpdateOrganisation, etc., you can further specialise the functions of the interface and adapt them to the specific use case.

10.1.8.2 KIX as a requester - use of webhooks

The add-on Connect Webservice (Client) enables the integration of HTTP/s web services in KIX. The web services can be both the data source and the target for active data transmission. Common synonyms for these functions are also "outgoing webhooks" or "invoker".

In contrast to [Connect Database \(see page 263\)](#), no direct database access is used, but access to a web interface via HTTP/s. In addition, Connect Webservice can also transmit information to the connected web services instead of just consuming it. Typical scenarios are:

- Transmission of information to third-party process/ticket systems, such as Jira, Redmine, Trello, BMC, etc.
- Triggering software deliveries through endpoint management systems such as Baramundi, i-doit, opsi, etc.
- Transmission of billing data from tickets to CRP or ERP systems such as Navision, Axapta, Sugar CRM etc.

Requirements

The technical prerequisite is the existence of an HTTP/S connection from the KIX backend to the target system including the access data (if required) with corresponding authorisations.

Advanced Macro Actions

The Connect Webservice add-on contains additional macro actions for communication with other web services.

Webhook Extended

The macro action "Webhook Extended" enables active communication with another web service.

Possible triggers for the macro action can be specific changes to a ticket or asset. Lists of entries can also be retrieved and processed using periodic jobs. Concrete applications are, for example, the inventory of the KIX asset database or the proactive creation of tickets.

The Macro Action works in a similar way to the "Webhook" Macro Action contained in the basic product. It differs in that complex, multi-level data structures can be transferred as a payload in the request. In addition, the response from the called web service can be received and used for further macro actions (e.g. [XSL Transformation, Get Object Data \(see page 238\)](#)).

Parameter



Parameter	Description	Example
Response	Variable name for the result structure of the web request. A structured object is returned, which contains HTTP code, status, content, header (key-value pairs) and result with the result as a JSON object.	-
URL	The URL to call. KIX placeholders are supported.	-
Method	the invocation method (HTTP verb)	GET POST PUT PATCH DELETE OPTIONS HEAD
Use Proxy	Should a proxy server be addressed?	yes no
Proxy	Which proxy server should be addressed? If nothing is specified, the general proxy setting from the system configuration is used ("WebUserAgent::Proxy" option).	http://proxy.example.com:3128
Headers	HTTP headers to be transmitted	Content-Type: application/json
Content	The content to be transmitted ("payload"). Can be specified as a JSON string or as a macro variable. KIX placeholders are supported.	<pre>{ "issue": { "project_id": 1, "subject": "<KIX_TICKET_Title>" "description": "Lorem ipsum..." } }</pre> <p>or</p> <pre>\$(SomeMacroVariable)</pre>

Parameter	Description	Example
Debug	Activate/deactivate extended logging for the call. Tip: Debug information can also be found in the job history.	yes no

⚠ N-quotation for placeholder content

When using placeholders in the content, make sure that the content is already JSON-quoted. This is not the case, especially with multi-line content. If such a placeholder is used, it can first be assigned to a macro variable and then inserted with JSON quotes.

1. Macro Action ("Assign Variable")

```
Variable := "ArticleBody"
Value := "<KIX_FIRST_Body>"
```

2. Macro Action ("Webhook Extended")

```
{
  "issue": {
    "project_id": 1,
    "subject": "<KIX_TICKET_Title>",
    "description": "{$ArticleBody|JSON}"
  }
}
```

Example: Create a Confluence page

Scenario:

Creation of a new Confluence page as soon as a ticket of the type "Problem" with the close code "Known Error" is closed. For example, known errors can be communicated directly and documented so that everyone can see them.

The example creates the new page in Confluence as a subpage to an existing page with ID 123456 in the "BPTTEST" area. It doesn't matter which answer Confluence delivers ("fire and forget"). Basic Auth, i.e. a combination of user name and password, is used for authentication.

Preparation:

- Extend dynamic field "Close Code" with value "Known Error".
- Create a dynamic "Workaround" field and [integrate](#) (see page 145) it into the "Close" ticket action
- Obtain Confluence access data
 - in the example: Username / Password

- Determine the ID of the page in Confluence under which the page to be created should be located
 - in the example 123456
- Generation of an HTTP header for Basic Auth
 - the value is required as an HTTP header to log into Confluence
e.g. using a command line tool "base64":

```
someLinuxUnixSystem:~# echo 'Authorization: Basic '"$(echo -n
'UserName:Password' | base64)'"
Authorization: Basic VXNlck5hbWU6UGFzc3dvcnQ=
```

Configuration of the job:

1. Basic configuration:
 - Job Type: "Ticket"
 - all other job parameters (name, execution plan, filter, etc.) can be defined individually.
2. Macro Action 1 - Write article content to macro variable
 - Action: Set variable
 - Variable: ArticleBody
 - Value: <KIX_ARTICLE_Body>
3. Macro Action 2 - call of the web service
 - Action: Webhook Extended
 - Response: -
 - URL: http://confluence.intra.example.com:8092/rest/api/content/
 - Method: POST
 - Use Proxy: -
 - Proxy: -
 - Headers
Content Type: application/json
Authorization: Basic VXNlcm5hbWU6UGFzc3dvcnQ=
 - Content:

Content

```
{
  "type": "page",
  "space": {
    "key": "BPTTEST"
  },
  "ancestors": [
    {
      "type": "page",
      "id": "123456"
    }
  ],
}
```



```
"title": "Known Error <KIX_TICKET_TicketNumber> - <KIX_TICKET_Title>",  
"body": {  
  "storage": {  
    "value": "<h1>Description</h1>${ArticleBody|JSON} <br/>  
<h2>Workaround</h2> <KIX_TICKET_DynamicField_Workaround>",  
    "representation": "storage"  
  }  
}
```

References:

- <https://developer.atlassian.com/server/confluence/confluence-rest-api-examples/>
- https://en.wikipedia.org/wiki/Basic_access_authentication

Example: Create Redmine Issue

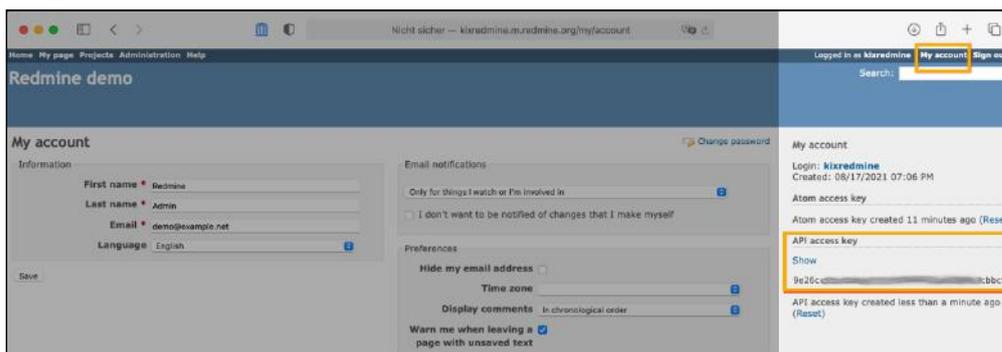
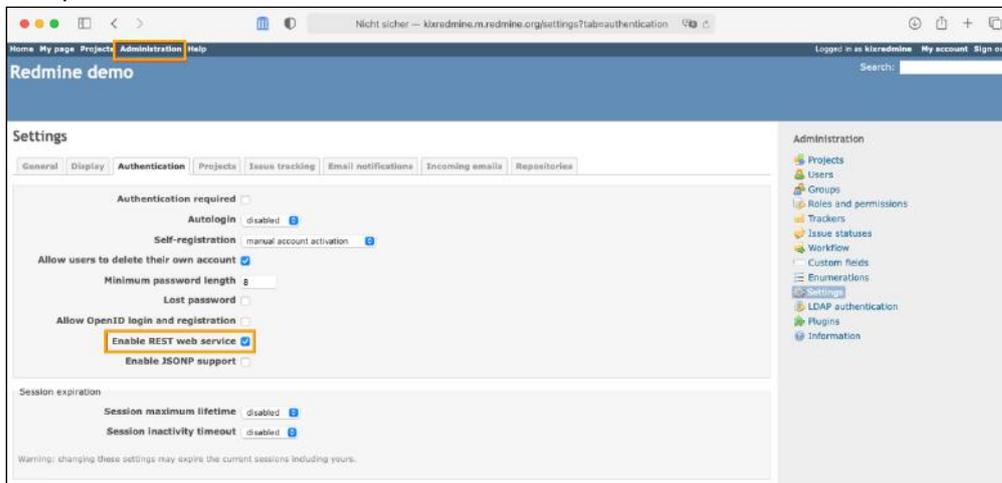
Scenario:

Creation of an issue in Redmine from a KIX ticket as soon as a ticket of the type "Incident" is provided with the solution code "Processing in Redmine". The content of the process is processed in the project management tool "Redmine". A previously determined authentication token is used for authentication.

The example creates a new issue in Redmine with the title and content of the first article of the triggering KIX ticket. The job is triggered by a change in the dynamic field "Close Code" with filtering on the value "Processing in Redmine". The IssueID created by Redmine is stored in a dynamic field in KIX. If no issue can be created, a note is stored.

Preparation:

1. Set up Redmine access and determine AuthToken



2. Extend KIX dynamic field "Close Code (see page 52) " with value "Processing in Redmine".
3. Set up the RedmineIssueID dynamic field in KIX and make it available (see page 145) for use in the Edit ticket action.
4. Set up the "Project" dynamic field in KIX and make it available for use in the "Edit" ticket action.

Configuration of the job:

In step 4 ("Actions"), the job configuration contains several macro actions that build on one another:

1. Get Object Data - Get the data for the web request
2. XSL Transformation - processing of the web request content
3. Webhook Extended - calling the web service
4. XSL Transformation - Processing of the response
5. Set Dynamic Field - Saving the Redmine IssueID

The configuration in detail:

1. Basic configuration:
 - "Ticket" is used as the job type,
 - all other job parameters (name, execution plan, filter, etc.) can be defined individually
2. Macro Action 1 - Get the data for the web request
 - Action: Get Object Data
 - ObjectData: TicketContentUnprepared
 - Object Type: Ticket
 - Object ID: <KIX_TICKET_ID>
 - Includes: Dynamic Fields, Priority, Articles
 - Expands: -/empty
3. Macro Action 2 - Preparation of the web request content
 - Action: XSL transformation
 - TransformedData: RMRequestPrepared
 - Data: `${TicketContentUnprepared}`
 - XSL templates:

XSL Template

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:KIX="urn:KIX">
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="RootElement">
    <root>

      <!-- This is the custom part... -->
      <xsl:variable name="ProjectName" select="DynamicFields[Name =
'Project']/DisplayValue"/>

      <issue>

        <!-- translate project name in KIX to Redmine project id -->
        <project_id>
```

```

<xsl:choose>
  <xsl:when test="$ProjectName='Sample Project'">1</xsl:when>
  <xsl:when test="$ProjectName='Other Project'">2</xsl:when>
  <xsl:otherwise>1</xsl:otherwise>
</xsl:choose>
</project_id>

<!-- translate KIX priority to Redmine priority -->
<priority_id>
  <xsl:choose>
    <xsl:when test="Priority='1 very high'">6</xsl:when>
    <xsl:when test="Priority='2 high'">7</xsl:when>
    <xsl:when test="Priority='3 normal'">1</xsl:when>
    <xsl:when test="Priority='4 low'">5</xsl:when>
    <xsl:when test="Priority='5 very low'">8</xsl:when>
    <xsl:otherwise>1</xsl:otherwise>
  </xsl:choose>
</priority_id>

<!-- compose subject "<TicketTitle> (T#<TicketNumber>)" -->
<subject>
  <xsl:value-of select="Title"/>
  <xsl:text disable-output-escaping="yes">(T#</xsl:text>
  <xsl:value-of select="TicketNumber"/>)
</subject>

<!-- just submit first article and a timestamp-->
<description>
  <xsl:value-of select="Articles[1]/Body"/>
  <TimeStamp>
  <xsl:value-of select="KIX:TimeStamp()"/>
</description>

</issue>
<!-- ...E0 custom part. -->

</root>
</xsl:template>
</xsl:stylesheet>

```

- Force Array Tags: -
- Suppress Empty: Empty Hash
- Debug: no

4. Macro Action 3 - Calling the web service

- Action: Webhook Extended
- Response: RMResponse
- URL: <https://kixredmine.m.redmine.org/issues.json>
- Method: POST


```

        <RMIssueID><xsl:text>Not transfered to Redmine.</xsl:text></
RMIssueID>
        </xsl:otherwise>
    </xsl:choose>

    </root>
</xsl:template>
</xsl:stylesheet>

```

- Force Array Tags: -
- Suppress Empty: Empty Hash
- Debug: no

6. Macro Action 5 - Save the Redmine IssueID

- Action: Set Dynamic Field
- Dynamic Field Name: "RedmineIssueID"
- Dynamic Field Value: `${RMResponsePrepared.RMIssueID}`

i You can activate the "Debug" macro action parameter for error analysis. The following XSL template ensures that the incoming parameters are output directly, i.e. without transformation:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:KIX="urn:KIX">
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="/">
    <xmp>
      <xsl:copy-of select="*" />
    </xmp>
  </xsl:template>
</xsl:stylesheet>

```

References:

- <https://www.redmine.org>
- https://www.redmine.org/projects/redmine/wiki/Rest_api
- https://www.redmine.org/projects/redmine/wiki/Rest_api#Authentication

Post Attachment as Form Data

The macro action "Post Attachment as Form Data" allows attachments to be sent to other web services that can receive Multipart/FormData. For example, to create a Jira issue from a ticket or when a ticket is created, which also contains the attachments that are stored on the article.

Parameter

Parameter	Description	Example
Response	Variable name for the result structure of the web request. A structured object is returned that contains HTTPCode, Status, Content, Header (key-value pairs) and Result with the result as a JSON object.	MyAttachmentResponse
URL	The URL to be called up. KIX placeholders are supported.	http://192.168.188.48:3001/rest/api/2/issue/KIX-123/attachments
Use Proxy	Should a proxy server be addressed?	yes no
Proxy	Which proxy server should be addressed? If nothing is specified, the general proxy setting from the system configuration is used (option "WebUserAgent::Proxy").	http://proxy.example.com:3128
Headers	HTTP header to be transmitted	X-Atlassian-Token: nocheck Authorization: Basic VXNlck5hbWU6UGFzc3dvcnQ=
Attachment Data	The content to be transmitted. A macro variable is usually used here, which must contain an object with the structures Content, ContentType and Filename, e.g.: <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <pre>SomeMacroVariableWithStructure = { "Content": "<base64 encoded content here>", "ContentType": "image/png", "Filename": "sample.png" }</pre> </div> Further information: see also Macro Actions "Assemble object", "Create report".	\$ {SomeMacroVariableWithStructure }

Parameter	Description	Example
Debug	Enable/disable extended logging for the call. Tip: Debug information can also be taken from the job history.	yes no

Variables and advanced filters

Use case: A specific endpoint is configured to receive an array of assets, contacts, organisations or ticket data. This data is processed in a macro action "loop" to create the relevant KIX object (asset, contact, organisation or ticket). The IDs of the objects to be created within the loop are to be collected in order to return **one** collective answer/response with all generated IDs after processing. An array is required to collect the IDs.

Application in macros

Macro variables can hold several values and thus form an array. The individual values are separated by commas (see also variables):

```
`${Variable1,Variable2}
```

If one of the transferred parameters is already an array, the two arrays are concatenated:

```
Variable1 := ['Test1.1','Test1.2']
Variable2 := ['Test2.1','Test2.2']
`${Variable1,Variable2} == ['Test1.1','Test1.2','Test2.1','Test2.2']
```

Extended operations/filters

Additional variable filters are provided with KIX Connect Webservice for extended array operations.

Filter	Description	Examples
VariableUtil.Push	Converts the variables into an array (if it is not yet an array) and adds the transferred value to the end of the array.	Variable1 := ['Test1.1','Test1.2'] Variable2 := ['Test2.1','Test2.2'] `\${Variable1} VariableUtil.Push(Variable2)} == ['Test1.1','Test1.2',['Test2.1','Test2.2']]

Filter	Description	Examples
VariableUtil.Pop	Converts the variable into an array (if it is not yet an array) and removes the value at the end of the array.	Variable1 := ['Test1.1','Test1.2'] \${Variable1 Pop} == ['Test1.1']
VariableUtil.Shift	Converts the variable into an array (if it is not yet an array) and removes the value at the beginning of the array.	Variable1 := ['Test1.1','Test1.2'] \${Variable1 Shift} == ['Test1.2']
VariableUtil.Unshift	Converts the variable into an array (if it is not yet an array) and adds the transferred value to the beginning of the array .	Variable1 := ['Test1.1','Test1.2'] Variable1 := ['Test2.1','Test2.2'] \${Variable1 VariableUtil.Unshift(Variable2)} == [['Test2.1','Test2.2'],'Test1.1','Test1.2',]

 **Note**

The functions do not change the actual variable, but only apply to the corresponding call! If the result is to be persistent, the macro action "Set variable" must be used.

10.2 Add-on "ITIL Practices"

To work effectively with tickets, you need ways to communicate about tickets and to exchange and capture information. KIX in combination with the add-on "ITIL Practices" provides you with the necessary means to automate certain processes in which tasks - based on events, time and defined processes - are processed. For each ticket type, there are suitable statuses and possible status transitions - both automatic and manual. The underlying process management guides customer users and agents through ticket creation and ensures that tickets run through defined processes at all times and are completed at the end. Thus, if a certain status is selected in a ticket, further process steps take place automatically.



KIX Pro 18 has been awarded SERVIEW CERTIFIED TOOL for 15 out of 19 possible ITIL® 4 practices. To make the use of ITIL® 4 as easy as possible for all users, the add-on "**ITIL® Practices**" was developed.

"ITIL Practices" is an add-on for KIX Pro 18 and is supported from version 26. It includes a number of pre-configured reports, asset classes, FAQ categories, ticket templates and actions to handle problem cases, services and management tasks according to ITIL®4. The add-on ensures

that all necessary information is available in the right place and that contact persons are always informed about their tasks and the next steps in the process.

The use of processes makes workflows comprehensible and reduces sources of error. In addition, agents and customers have an overview of all information throughout the entire process at all times. The resulting shortening of processing and throughput times ensures an increase in productivity while reducing costs. The planning and control tailored to you shortens downtimes, reduces inventories and thus saves further costs.

The add-on "ITIL Practices" comes with pre-configured ticket templates and actions for the following processes:

- Service Request
- Incident
- Problem
- Change
- Contingency Plans (Business Continuity Preparation)

10.2.1 Video

You can also find a detailed showcase of the add-on "ITIL Practices" in our Youtube channel: <https://www.youtube.com/watch?v=M8Ds6TGIEWo> 



10.2.2 Activate/Deactivate "ITIL Practices"

10.2.2.1 Prerequisite

As a basis for using the add-on "ITIL Practices", you need KIX 18 Pro, which runs at least on version 26.

10.2.2.2 Installation/System Update

After the assignment, we will provide you with an updated image of your system - usually on the next working day.

If you are running KIX Pro on-premises, please perform a system update afterwards:

```
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./stop.sh  
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./update.sh
```

Afterwards, the add-on is integrated into your system and can be activated by you.

Our support team will provide the add-on in the KIX Cloud.

10.2.2.3 Activate the "ITIL Practices" add-on

After the system update or after deployment in the cloud, you will find the add-on "ITIL Practices" in the explorer of the admin module. Click on "Activate" on the welcome page. You will be asked whether you want to activate the configuration of the add-on. Confirm this.

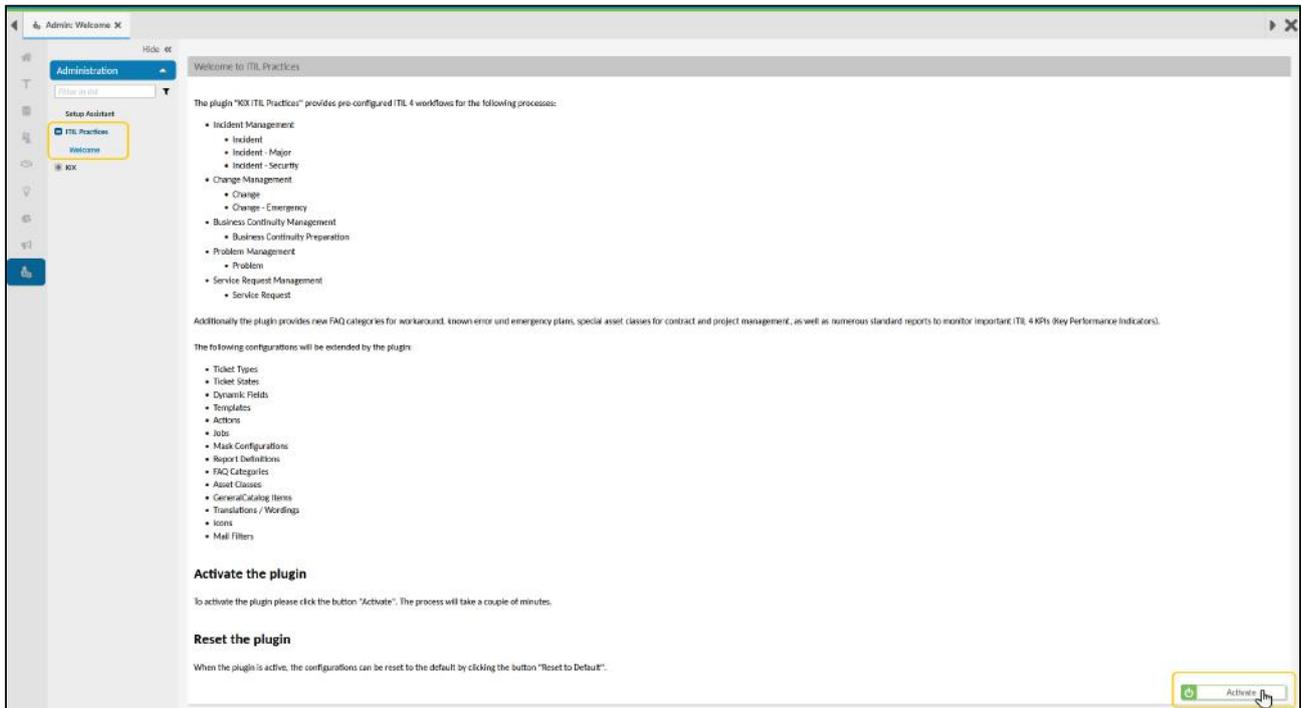


Fig.: Activate add-on "ITIL Practices"

KIX then starts the import. This process can take a few minutes. The installation routine shows you which packages are being imported. Finally, a success message is briefly displayed.

After that, the add-on "ITIL Practices" is integrated into KIX. No further general steps are required. You can use the components delivered with the add-on directly or adapt them to your specific use case with only a few steps.

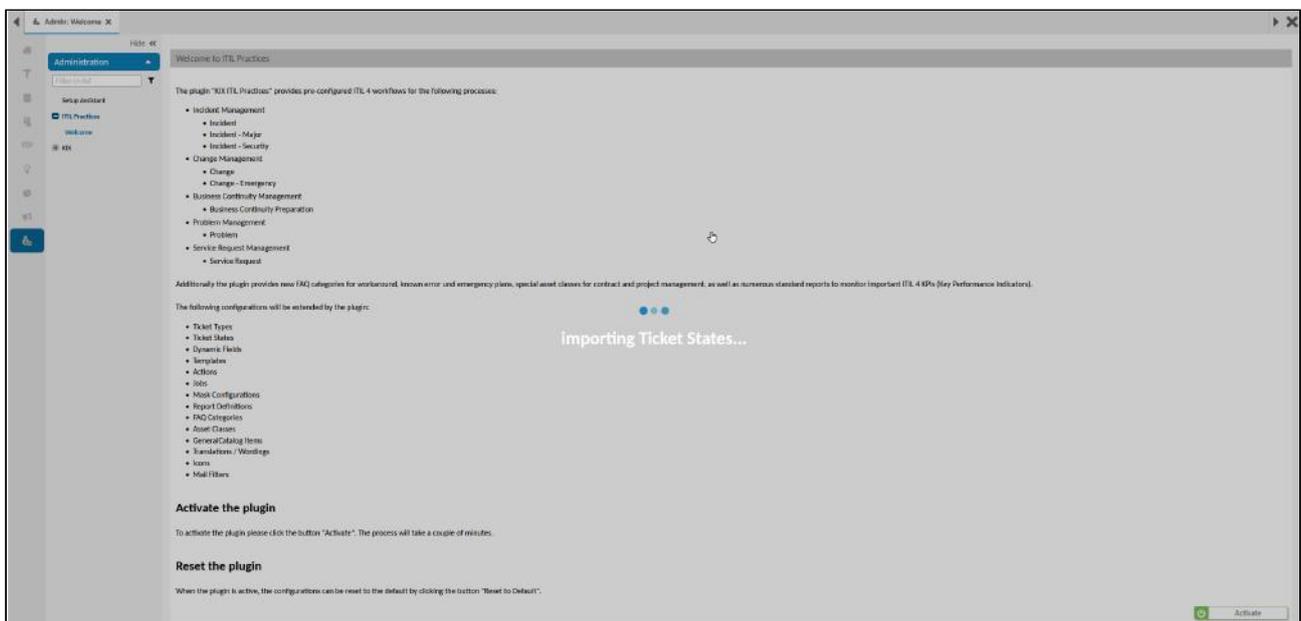


Fig.: Setting up the add-on "ITIL Practices"

With the activation of the add-on, KIX is supplemented with further configurations, objects and reference data that are required for the processes. In addition, existing configurations are adapted.

 **Important!**

During activation, various standard configurations are overwritten. If you have changed standard configurations, **please save them beforehand!**

The following master data and configurations are supplemented or extended by the add-on:

- Ticket types
- Ticket status
- Dynamic fields
- Templates
- Actions
- Jobs
- Mask configurations
- Report definitions
- FAQ categories
- Asset classes
- GeneralCatalog Entries
- Translations / Terminology
- Icons
- Mail Filter

10.2.2.4 Reset the ITIL Practices add-on

You can reset the add-on "ITIL Practices" to the delivery state. To do this, click on "Reset to default". Resetting the system takes a few minutes.

All configurations that are delivered by the add-on and have an unchanged identifier/name are reset to the default values of the add-on. This does not apply to translations and role assignments (affects reports, ticket templates and actions). Created tickets, articles and reports etc. remain unchanged.

10.2.3 Implementation of ITIL practices by Means of Processes

The add-on "ITIL Practices" provides a number of already configured reports, ticket templates and actions with which problem cases, service notifications and change requests can be processed according to ITIL®4. An overview of this can be found in the chapter "Advanced Configuration".

The following prepared ITIL®4 workflows (processes) are delivered with the add-on:

- Service Request Management
 - Service Request

- Incident Management
 - Incident
 - Incident - Major
 - Incident - Security

- Problem Management
 - Problem

- Change Management
 - Change
 - Change - Emergency

- Business Continuity Management
 - Business Continuity Preparation

10.2.3.1 Ticket Templates

The add-on contains a separately configured ticket template for each of the above-mentioned processes. This can be found when creating a new ticket under the template group "IT Requests". The selected template determines the further course of the process.

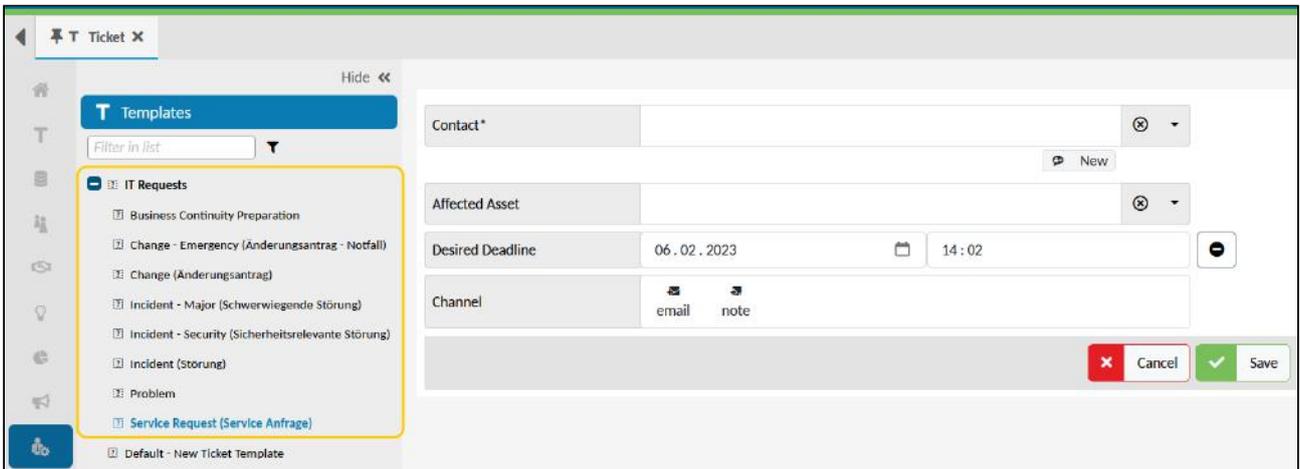


Fig.: Ticket templates for ITIL practices in the agent portal

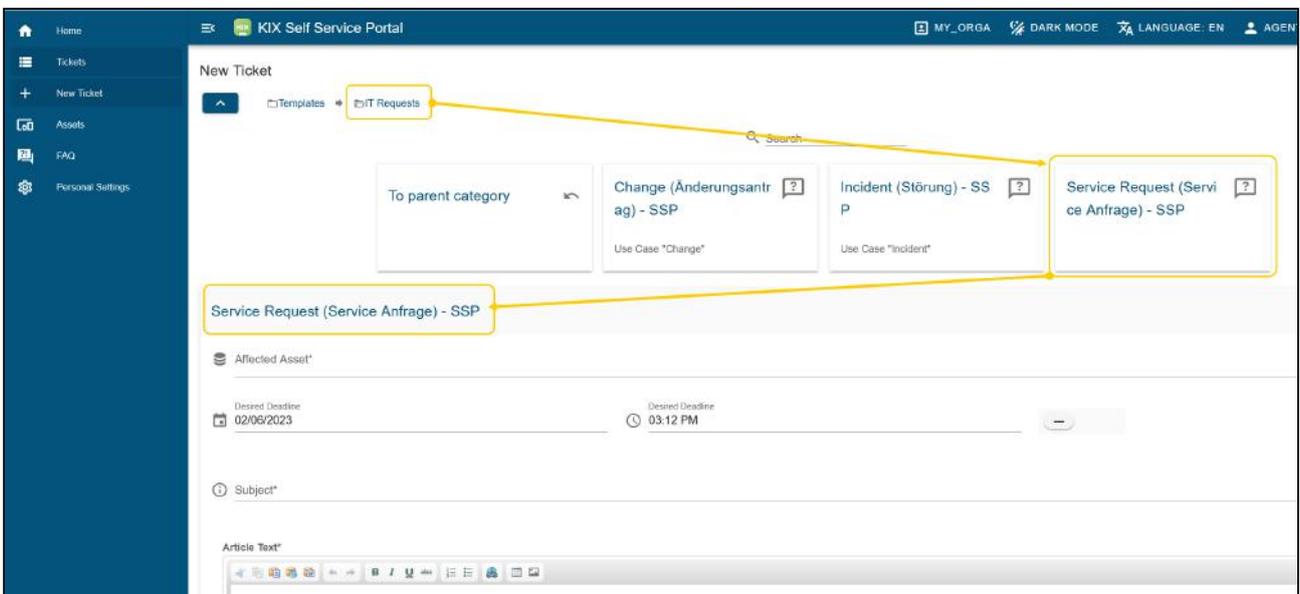


Fig.: Ticket templates that the add-on "ITIL Practices" brings with it in the SSP

An overview of the ticket templates included in the add-on as well as further notes can be found under: [Templates for "ITIL Practices"](#) (see page 356)

Tip

If you want to translate the names of templates or dynamic fields, for example, you can store the corresponding patterns in the Admin Module under *KIX > Internationalisation > Translations*. However, checklist items are not translated. Title and description are displayed as they are created. If necessary, you can change them yourself in the respective checklists.

10.2.3.2 Ticket Actions

ITIL Practices changes the configuration of several standard ticket actions. For example, the actions "Edit ticket" or "Close" are hidden to ensure a correct process flow.

In addition, ITL Practices implements a number of ticket actions that are required for processing. When and under which conditions an action is available on the ticket depends on the selected process and the process flow. The process flow is also based on the status of the ticket. ITIL Practices provides several new ticket statuses for this purpose.

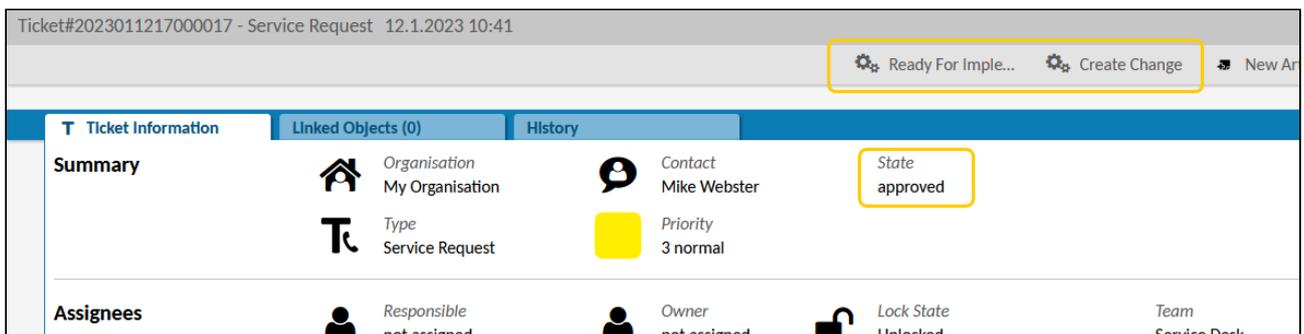


Fig.: Ticket actions depending on the ticket status.

The configuration of the actions only takes standard users into account in the delivery state. However, you can adapt the actions individually in order to restrict or extend authorisations or to integrate further dynamic fields in dialogues.

An overview of the ticket actions included in the add-on can be found under: [Actions in "ITIL Practices"](#) (see page 348)

Process State

The dialogue-based ticket actions delivered with the add-on all contain the field "Process State". The process state is a dynamic field of the type Checklist. Each item on the checklist corresponds to a process step that can be answered by specifying a state. Depending on the selected state, the process state (recognisable by the progress bar) and the ticket status will change.

Status	Meaning	Process state
OK	Okay / Done / Approved	Is incremented because valid answer.
NOK	Not OK / Not executed / Not approved	Is incremented because valid answer.
PENDING	Pending status / Waiting for feedback / On hold	Is not incremented, as floating state

Status	Meaning	Process state
n.a.	No information / Information not required / No statement made	Is incremented because e.g. "no statement made" is also a valid answer.
-	Unanswered / No selection made at the checklist item	Is not incremented.

In the example of a service request, the facts are first checked. In order to confirm this, the checklist item "Review" receives the status OK. The process step of the check is thus completed and the ticket switches to the next process step "Approval". If the approval is granted, this checklist item also receives the status OK. The same applies to all further process steps (checklist items), so that the ticket is moved forward in the process. Once all process steps have been completed, the ticket can be closed. If a process step is answered with NOK, -, n.a. or with PENDING, the process is no longer advanced.

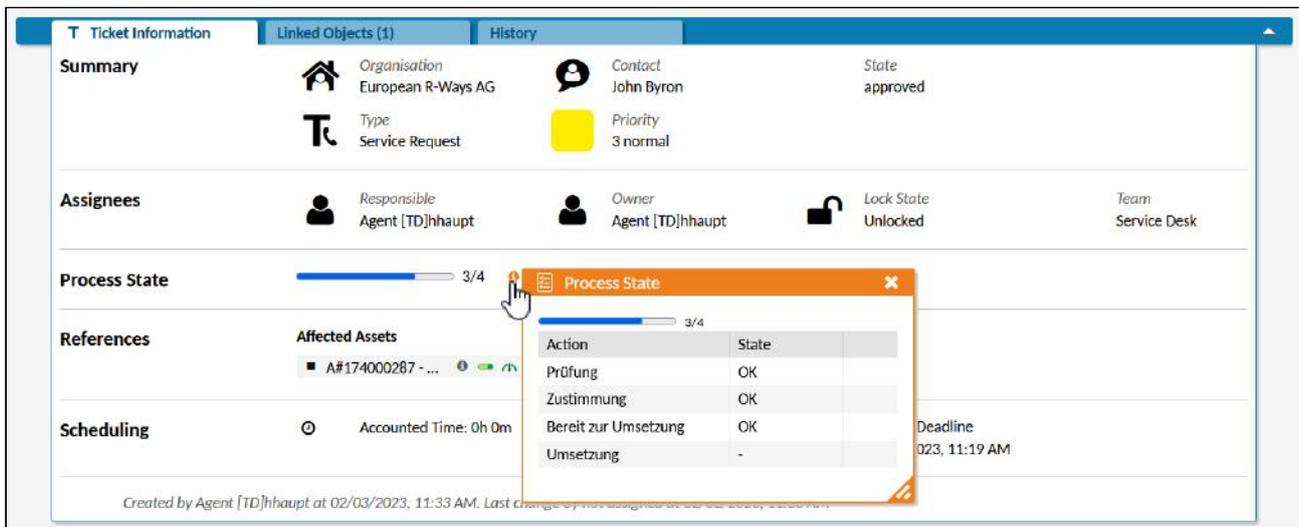
The actions available on the ticket depend on the respective process state. This means that the actions available on the ticket are always those that are relevant for the respective process step (see above Ticket actions).

Note: The process flow cannot be undone. A change to the status of the ticket is not possible retroactively; the saved process state remains, even if the individual statuses are changed. This ensures a correct process flow. Exception: Rejection of a solution leads back to the process step "Approval | Zustimmung".

 **Attention!**

For each process, the "ITIL Practices" add-on brings a separate "Process state" checklist. The configurations of these checklists **must not be changed**. Otherwise, the jobs linked to them will no longer function correctly, so that the entire process flow will be destroyed.

The process state of a ticket can also be seen in the progress bar in the ticket detail view. A click on the info symbol shows the details.



Action	State
Prüfung	OK
Zustimmung	OK
Bereit zur Umsetzung	OK
Umsetzung	-

Fig.: Process state of a ticket in the ticket detail view

✓ Tipp

The title and description of the checklist are not initially translated. They are displayed as they are created. If necessary, you can change them yourself in the respective checklists.

10.2.3.3 Permissions

The configurations for jobs, actions, report definitions etc. provided with the add-on always use the initial default values (roles, users, teams etc.). If necessary, you must adjust values in the configurations, e.g. to grant additional authorisations.

Further information can also be found in the chapter "Advanced configuration".



10.2.3.4 Process 1: Service Request

A service request is usually an interruption of a service availability. A service request can be, for example:

- The completion of tasks by other departments
 - Provision of consumables
 - Procurement of hardware
- Customer enquiries
 - Resetting a password
 - Access to applications.

Flowchart of a Service Request

KIX handles a service request according to the following scheme and provides the required ticket templates, ticket statuses, actions etc. with the add-on "ITIL Practices".

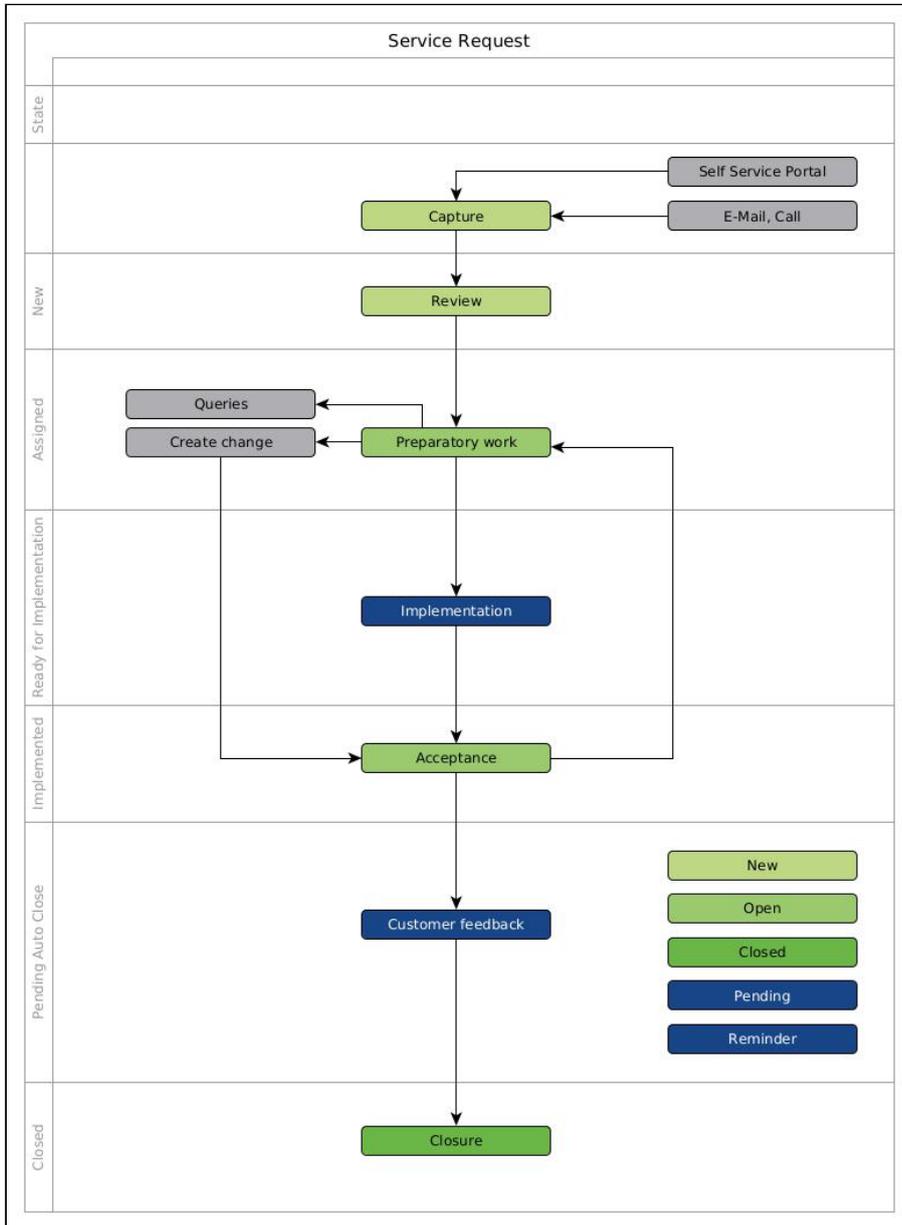


Fig.: Flowchart of the "Service Request" process

1. Capture

First, the service request (e.g. provision of consumables) is captured and a new ticket is created for it. The ticket template "Service Request" is used to record the service request.

With saving, the ticket changes to the next process step "Review" and receives the default values for priority (normal) and status (new).

2. Review

During the review, the facts are checked and assigned to the responsible agent. The "Review" action on the ticket is available for this purpose. It opens a dialog in which, among other things, the responsible agent can be selected and the priority can be reset.

If the "Review" process step receives the status OK, the ticket is forwarded to the next process step "Ready for Implementation" when the ticket is saved.

- The ticket is transferred to the next process step "Ready for Implementation",
- Assigned to the team of the selected agent,
- The ticket status is set to "assigned".

3. Ready for Implementation

Once the service request is assigned, its implementation can be prepared. A procurement and thus another ticket (child) may be necessary (see "Branch to the Change Process" below).

Once the procurement has taken place, the implementation can also take place. For this purpose, the action "Ready for Implementation" is available on the ticket (one-click action without dialogue). The use of the action gives the OK for the implementation, so that the ticket is transferred to the next process step. The ticket status remains "ready for Implementation".

4. Implementation

Once the necessary preparatory work (e.g. ordering the consumables) has been completed, the service request can be implemented. The action "Implementation" is available on the ticket for this purpose. It opens a dialogue to store the result and/or notes on the implementation on the ticket. For example, to note the reasons for a failed implementation.

If the implementation is answered with OK, the ticket receives the status "implemented". In the next process step, the solution can be accepted or rejected.

5. Acceptance

If the conversion has taken place (e.g. the consumable has been delivered), the solution can be accepted (e.g. correct consumable) or rejected (e.g. wrong consumable). The ticket actions of the same name are available on the ticket for this purpose.

If the solution is rejected, the ticket is reset to the 3rd process step (Ready for Implementation) so that the process can start again.

If the solution is accepted, the process is considered completed. The ticket is set to the status "pending auto close" assigned.



Branch to the Change process

If the action "Create Change" is provided on the ticket, this enables a process branch. For example, if procurement measures (e.g. ordering consumables) are required for a service request. The action creates another ticket (child) and links it to the source ticket (parent).



10.2.3.5 Process 2: Incident

The process of an incident usually involves the interruption of a service or an unplanned event that leads to the failure of a service. The aim is to restore normal service operation as quickly as possible.

An incident can be, for example:

- functional impairment of a web server
- an interrupted power or data connection
- an error message reported by a device
- failure of a business application
- a cyber attack
- defective hardware or equipment
- and many more.

Flowchart of an Incident

If an incident is reported, a ticket can be created in KIX according to the process "Incident". The process of an incident runs according to the following scheme:

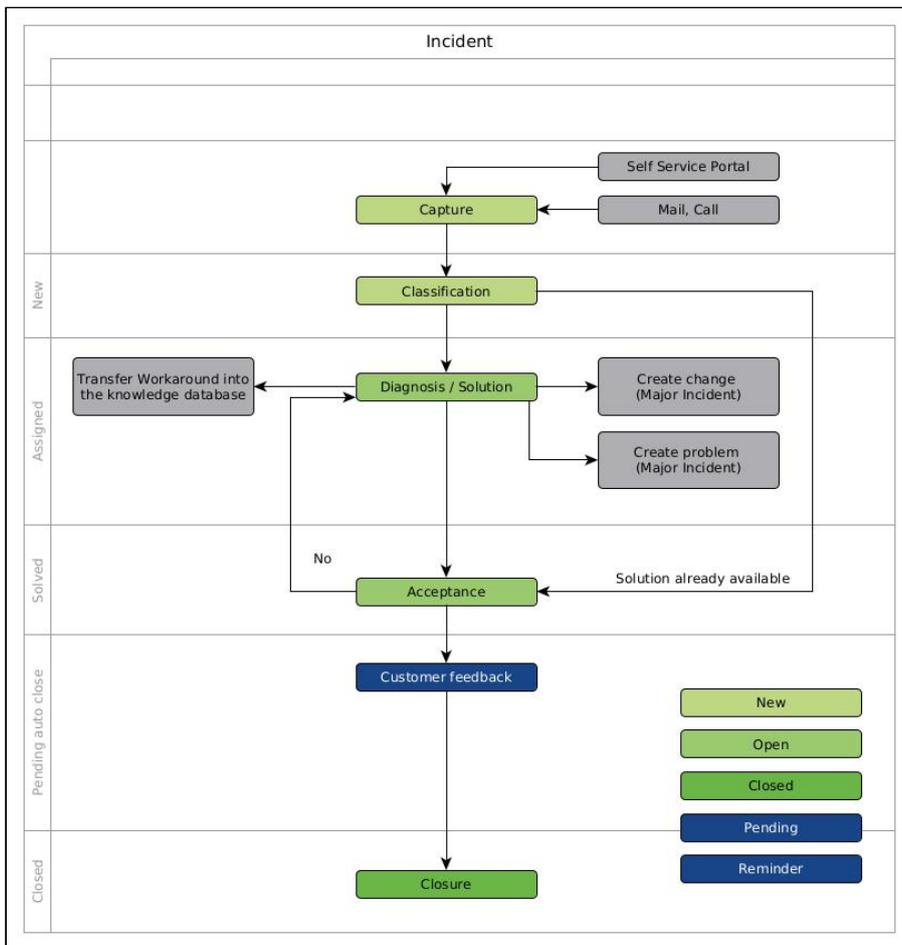


Fig.: Flowchart of the "Incident" process

1. Capture

The add-on "ITIL Practices" distinguishes according to the type of incident:

- Incident
- Incident - Major
- Incident - Security

The add-on provides a ticket template with the same name for each of these incident types.

When saved, the ticket receives the default values for priority (normal) and status (new). In the next process step - when classifying the ticket - the ticket can be evaluated differently.

2. Classification

By classifying, the severity of the disruption is indicated and the urgency of the ticket is classified. This is based on the effects of the impact. The action "Classification" is available on the ticket. In the dialogue that opens, the impact can be classified, the priority can be changed, the symptoms can be described and the ticket can be assigned to the responsible agent. The priority of the ticket cannot be changed after it has been classified.

After saving, the ticket receives the status "assigned" and switches to the next process step "Diagnosis" or "Solution".

3. Diagnosis

Diagnosis identifies the cause of the incident. The action "Diagnosis" is available on the ticket for this purpose. If the diagnosis is confirmed with OK, the ticket receives the status "in process". The action "Diagnosis" is still available on the ticket, so that changes can be made, e.g. to the effect or additions to the symptom/cause.

The process step "Diagnosis" can be skipped if a solution is available, e.g. if the problem has already been solved. Then use the ticket action "Solution".

4. Solution

At the end of the incident process is the solution. A ticket action with the same name is available on the ticket. In the dialogue that opens, the following can be specified, among other things:

- Solution description: e.g. work carried out, information on non-feasibility, etc.
- WorkAround: Description of an alternative procedure or a workaround. If a workaround is specified, an FAQ entry is automatically created.
- Close code: Specification of the solution (completed, not completed, etc.).

If a close code is selected and the processing status of the solution is answered with OK, the ticket receives the status "solved" and switches to the next process step "Acceptance".



5. Acceptance

If the fault is solved (positively or negatively), the solution can be rejected or accepted. The ticket actions of the same name are available on the ticket for this purpose.

- If the solution is accepted, the process is considered successfully completed. The ticket is set to the status "pending auto close".
- If the solution is rejected, the ticket is reset to the 3rd process step (Diagnosis/Solution). The ticket receives the status "assigned" so that the process can start again.

Branch to the Change process

If the action "Create change" is provided on the ticket, this enables a process branch to the change management, e.g. if procurement measures are required (e.g. ordering new server technology). The action creates another ticket (child) and links it to the source ticket (parent).

Branch to the Problem process

The ticket action "Create problem" is available for "Incident-major" faults. It enables a process branch to the problem management, e.g. if a frequently recurring defect is the cause of the incident. The action opens a dialogue for creating a new item and creates another ticket of the type "Problem" in parallel.



10.2.3.6 Process 3: Problem

A problem is usually the (possible) cause of (recurring) malfunctions (incidents). Problem management is concerned with identifying and managing these causes. The goals are to solve problems permanently, to avoid costly disruptions and to secure knowledge about (known) problems.

A problem can be, for example

- a changed configuration file
- causes of a damaged database entry
- software error
- and many more.

Flowchart of the Problem process

If there is a problem based on the guideline values of ITIL®4, a corresponding process ticket "Problem" can be created in KIX. The problem management in KIX runs according to the following scheme:

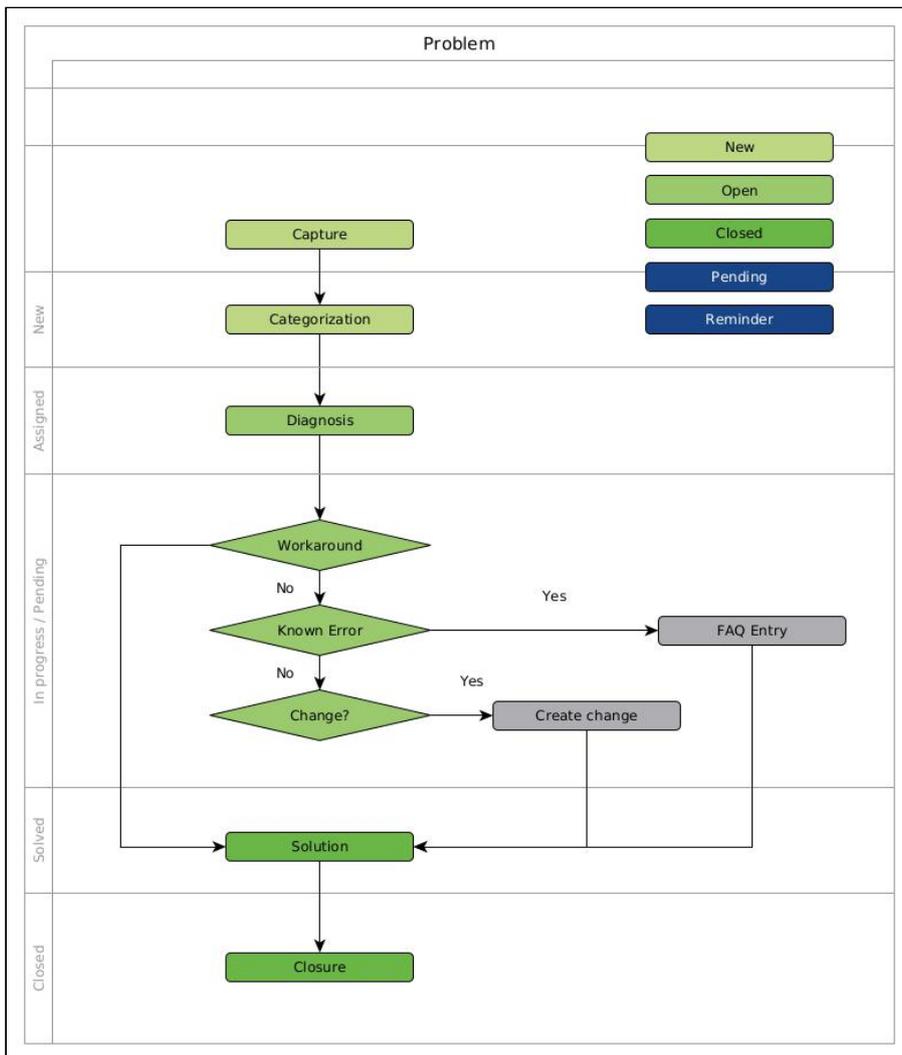


Fig.: Flowchart of the "Problem" process

1. Capture

The recording of a problem can be done in different ways:

- manually: using the ticket template "Problem".
- automated: using the ticket action "Problem" (process: Serious Incident) .

The created ticket receives the default values for priority (normal) and status (new) and switches to the next process step "Categorization".

2. Categorization

Categorising the ticket helps to evaluate the problem and give it a control variable. For this purpose, the ticket has the action "Categorization". It opens a dialogue in which, among other things, priority, impacts and

error category can be determined. The impacts depend on the type of impairment of the service. The error category is used to classify the problem in subject areas.

If the categorisation is answered with OK, the ticket receives the status "assigned" and switches to the next process step "Diagnosis".

3. Diagnosis

The diagnosis serves to identify the problem and to confirm or reject it. The ticket contains the action "Diagnosis" for this purpose. The result of the diagnosis is indicated in the "Problem State" field.

Once this has been done and the diagnosis has been confirmed with OK, the ticket receives the status "in progress" when it is saved.

The action "Diagnosis" is still available on the ticket, so that changes can be made, for example, to the problem status or additions to the symptom/cause.

The process step "Diagnosis | Diagnose" can be skipped if a solution is available. Then use the ticket action "Solution".

4. Known Error

A Known Error is an error with a known cause. A FAQ entry can be created for such errors. The action "Known Error" is available on the ticket for this purpose. It automatically creates a FAQ entry based on the information stored in the ticket - without dialogue - and links it to the source ticket.

5. Solution

If the problem could be solved or if a workaround to the problem exists, the ticket can be closed. This is done using the ticket action "Solution". In the dialogue that opens, the following can be specified, among other things:

- Solution description: e.g. work carried out, notes on non-feasibility, etc.
- Workaround: Description of an alternative procedure or a workaround. If a workaround is specified, an FAQ entry is automatically created.
- Problem state: The problem status must be specified if the process step "Diagnosis" was skipped.

If the problem status is set and the processing status of the solution is answered with OK, the ticket is closed directly. The process step of acceptance is omitted.

Branch to the Change process

If the action "Create change" is provided on the ticket, this enables a process branch to Change Management. The action opens a dialogue for creating a new item and creates another ticket of the type "Change" in parallel. Both tickets are linked to each other as a parent-child relationship.



10.2.3.7 Process 4: Change

Organisational or infrastructural changes that can have direct or indirect effects on services are assigned to change management.

A change can be, for example

- the replacement of a device
- introduction of new services
- etc.

Flowchart of a Change Request

KIX handles a change process according to the following scheme and provides the required ticket templates, ticket statuses, actions etc. with the add-on "ITIL Practices".

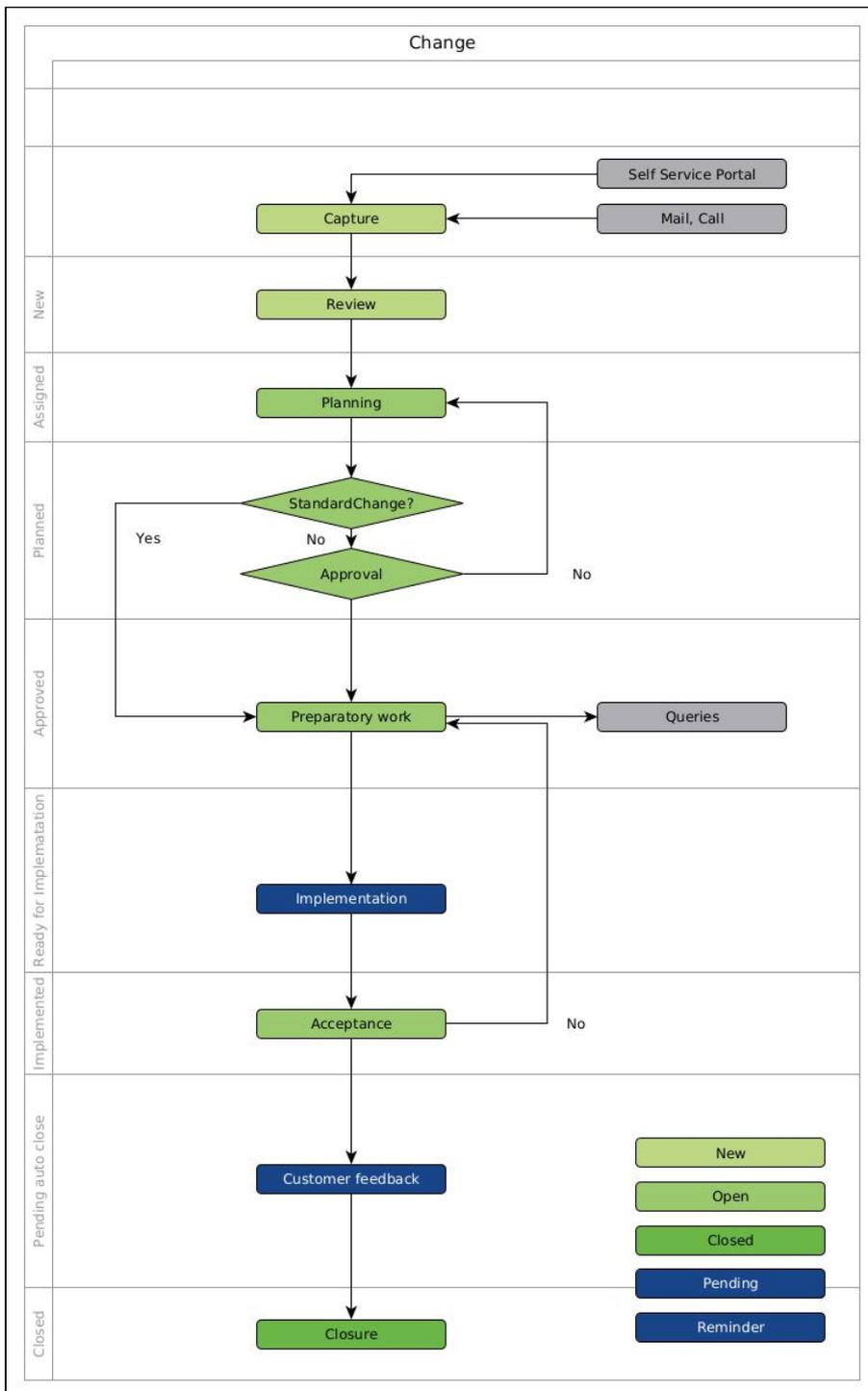


Fig.: Flowchart of a "Change" process in KIX

1. Capture

The add-on "ITIL Practices" distinguishes according to the type of change request:

- Change (Change request | Änderungsantrag): common changes with low or increased risk.

- Change - Emergency (Change request - Emergency | Änderungsantrag - Notfall): Changes due to an unexpected fault that needs to be fixed immediately.

The add-on provides a ticket template of the same name for each of these incident types. The procedure is the same for both change processes.

In the Business Case area, the

- the *reasons* for the request
- the *benefits* that the change will bring
- likely *implementation costs*
- and possible *alternatives*

can be noted.

The entry of a change request can be done in different ways:

- manually: using the corresponding ticket template
- automatically: by using the ticket action "Create change".

The created ticket receives the default values for priority (normal) and status (new) and switches to the next process step "Review".

2. Review

During the check, the facts are checked and possible risks are assessed. For this purpose, the action "Review" is available on the ticket. Several risks can be noted in the dialogue that opens.

The selection in the field "Standard Change" classifies the change:

- Standard Change - yes:
 - Usual changes, low-risk, frequently recurring, pre-approved (e.g. replacement of a headset).
 - Skips the "Approval" process step
- Standard Change - no:
 - Changes with increased risk or high financial expenditure (e.g. replacement of the backup server).
 - Requires approval

If the process step is completed with OK, the ticket receives the status "assigned" and switches to the next process step "Planning".

3. Planning

During planning, an estimate of the costs and consideration of any possibilities to reverse the changes is made. This can be specified using the ticket action "Planning".

If the process step is completed with OK, the ticket receives the status "scheduled" and switches to the next process step "Approval".

4. Approval

For change requests with increased risk or financial effort, approval is required before implementation. The ticket action "Approval" is therefore available for tickets that are not a standard change (see check). The action opens a dialogue in which the approval can be given in the processing status.

In case of approval (OK), the ticket changes to the status "approved" and to the next process step "Ready for Implementation".

In case of rejection (NOK), the ticket remains in the process step "Approval" until it has been granted and receives the status "planned".

5. Ready for Implementation

Once the change request has been approved, its implementation can be prepared. The ticket is in a pending status until, for example, all queries have been clarified and changes to the next process step "Implementation" after clicking on the ticket action "Ready for Implementation".

6. Implementation

After the hardware has been completely replaced, for example, the conversion is considered completed. The ticket action "Implementation" exists on the ticket. It opens a dialogue for documenting the implementation.

If the implementation is answered with OK, the ticket receives the status "implemented". In the next process step, the solution can be rejected or accepted.

7. Acceptance

Finally, the result of the conversion can be rejected or accepted. The ticket actions of the same name are available on the ticket for this purpose.

- If the solution is rejected, the ticket is reset to the 3rd process step (planning) so that the process can start again.
- If the solution is accepted, the process is considered completed. The ticket is set to the status "pending auto close".

10.2.3.8 Process 5: Business Continuity Preparation

Contingency plans are used to keep business operations running in the event of interruptions to business processes or services. The contingency plans available in the form of FAQs can be used to remedy the situation quickly.

Flowchart for contingency plans

KIX handles contingency plans according to the following scheme and provides the required ticket templates, ticket statuses, actions, etc. with the add-on "ITIL Practices".

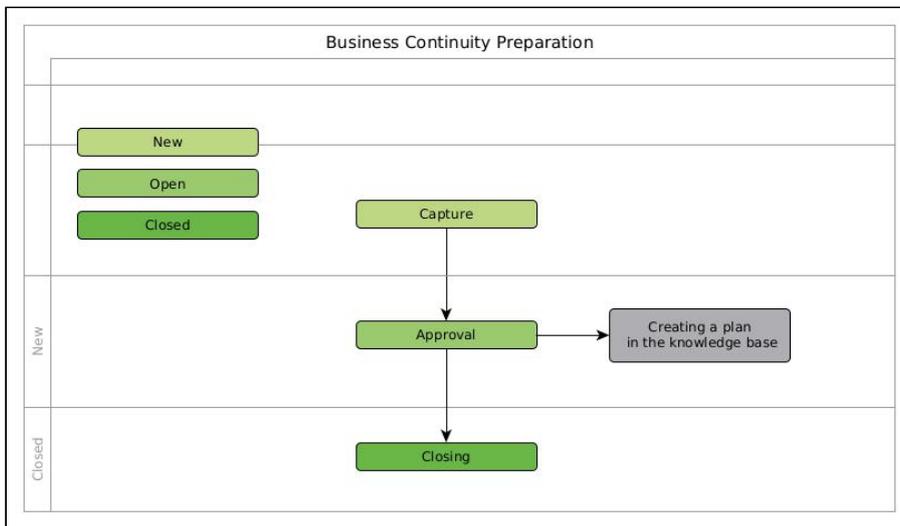


Fig.: Flowchart "Contingency plan" process

1. Capture

The ticket template "Business Continuity Preparation" is used to create an contingency plan. In the dialogue that opens, the task steps to be carried out in an emergency can be entered, among other things.

After saving, the ticket receives the default values for priority (normal) and status (new). In the next process step, the approval can be given.

2. Approval

In order for the prepared contingency plan to be stored in the FAQ, it must be approved. For this purpose, the ticket has the action "Approval". It opens a dialogue in which the approval can be given or rejected.

If the ticket is approved, a new FAQ entry is created.

In case of rejection, the ticket remains in the process step "Approval", so that the contingency plan can be changed and accepted at a later point in time.



3. Closing

Approval of a contingency plan gives it the status "created". Depending on the customer-specific definition, measures and work steps must then be completed in order to resolve the emergency. This concludes the process.



10.2.4 Advanced Configuration

KIX Pro already contains numerous basic functions for your IT service management. Building on this, the add-on "ITIL Practices" provides you with many useful configurations to get a quick start in ITIL® 4. You can quickly and easily integrate these configurations into your processes and flexibly adapt them to your individual needs.

The advanced configurations include elements from the following areas:

- Actions
- Assets
- Report definitions
- Dynamic fields
- FAQs
- Jobs
- Reference data
- Templates

10.2.4.1 Actions for "ITIL Practices"

Actions are individually configured functions (macro actions) that are available on the ticket or the article under certain conditions. The add-on "ITIL Practices" brings - related to the respective process - the following ticket actions:

- **General:**
 - Solution Rejected
 - Solution Accepted
 - Solution Accepted / Rejected (SSP)

- **Incident:**
 - Classification
 - Solution
 - Diagnosis
 - Create Change
 - Create Problem

- **Problem:**
 - Categorization
 - Diagnosis
 - KnownError
 - Solution
 - Create Change

- **Change**
 - Review
 - Approval
 - Planning
 - Ready For Implementation
 - Implementation

- **ServiceRequest**
 - Review
 - Ready For Implementation
 - Implementation
 - Create Change

- **Business Continuity Preparation**
 - Approval

In the menu *KIX > Workflow > Actions* of the Admin Module, you can reconfigure these actions and create new ticket- or item-based actions. By creating your own actions and defining which action is available to



whom and when, you can adapt the system to the workflow of your company and thus provide exactly those functions on the ticket that are needed for the daily service provision. An overview of which actions are initially delivered in KIX Pro can be found in the Admin Manual in the chapter "Actions".

Info: Actions are always applied to existing tickets and their articles. In contrast, templates are only applied to new tickets.

10.2.4.2 Assets for "ITIL Practices"

Asset Classes

All resources and resource information in the company, such as buildings, machines, equipment or even contracts, are referred to as assets and are created as such in the system. In order to map these in a structured way in KIX, they are assigned to asset classes. For example, all buildings in the company are grouped under the asset class "buildings", all computers in the company are grouped under the asset class "computers", etc. The administration of the asset classes can be found in the menu *KIX > Assets > Asset classes*.

The add-on "ITIL Practices" brings further asset classes for contract and project management:

- Agreement
- Data Processing Agreement
- Project

General Catalog

The General Catalogue is a catalogue in which a wide variety of values can be stored in order to reuse them, for example, in the class definition of an asset class. General Catalogue entries can be, for example: Attributes of asset classes, event states of assets, assembly series, model types, building types, etc. The General Catalog can be found in the menu *KIX > Assets > General Catalog*.

The add-on "ITIL Practices" provides the following additional General Catalog entries:

- `ITSM::ConfigItem::AffectedData`
- `ITSM::ConfigItem::AffectedPersons`
- `ITSM::ConfigItem::PlaceOfDelivery`

10.2.4.3 Dynamic Fields for "ITIL Practices"

Dynamic fields are individual input and output fields that can be created as needed and integrated into the interfaces. They enable the storage of additional information on the ticket as well as on organisations, contacts or FAQs.

The add-on "ITIL Practices" provides the following dynamic fields, which are used by the functions and ticket templates delivered with the add-on.

- Approval
- BackoutPlan
- BusinessCase
- Causes
- ChangeProcessState
- ChangeReference
- CostEstimation
- CreateChangeTicket
- CreateProblemTicket
- DesiredDeadline
- DisasterRecoveryPlan
- ErrorCategory
- Impact
- IncidentProcessState
- ProblemProcessState
- ProblemState
- Risks
- ServiceRequestProcessState
- Solution
- SolutionAccepted
- SolutionAcceptedComment
- StandardChange
- Summary
- Symptoms
- WorkAround

The administration of Dynamic Fields can be found in the menu *KIX > System > Dynamic Fields*.

By configuring the corresponding SysConfig keys (menu *KIX > System > SysConfig* or *KIX > System > GUI Configuration > Agent Portal*), the dynamic fields are integrated into the user interface at the desired position.

Please refer to the KIX Start Admin manual for further information on dynamic fields and their integration.



10.2.4.4 FAQ for "ITIL Practices"

The knowledge base contains a compilation of frequently needed information as well as answers to frequently asked questions (FAQ). Contingency plans, workarounds and known errors can also be stored here so that they are quickly available when needed.

The add-on "ITIL Practices" makes it possible to store information as FAQ entries directly from the processes." ITIL Practices" provides the following FAQ categories:

- Known Error
- Emergency plan
- Workaround

The administration of FAQ categories can be found in the Admin Module in the menu *KIX > Knowledge Base > FAQ Categories*. Administrators can create further FAQ categories there if required.

10.2.4.5 Jobs for "ITIL Practices"

The add-on "ITIL Practices" provides a series of jobs which, depending on the processing status, drive the progress of the processes and create FAQ entries. Jobs are created and configured in the menu *KIX > Automation > Jobs*.

 **Important!**

Only change the job configurations in exceptional cases. Otherwise there may be disturbances in the process.

With the add-on "ITIL Practices" you receive the following new jobs, related to the respective process:

- Change: 1 Set State on Review
- Change: 2 Set State on Planning
- Change: 3 Set State on Approval
- Change: 4 Set State on Ready For Implementation
- Change: 5 Set State on Implementation
- Incident: Set State on Classification
- Incident: Set State on Diagnosis
- Incident: Set State on Resolution
- Problem: Set State on Categorization
- Problem: Set State on Diagnosis
- Problem: Set State on Resolution
- ServiceRequest: 1 Set State on Review
- ServiceRequest: 3 Set State on Implementation
- Set State on Solution Accepted
- Create Change
- Create Problem
- Create FAQ Suggestion (Workaround)

10.2.4.6 Reference Data for "ITIL Practices"

In computer science and business administration, reference data refers to data that contains basic information about operationally relevant objects that are required for ongoing processing in business processes.

Both KIX Start and KIX Pro come with a range of reference data, such as a selection of ticket types or ticket statuses. Based on this, the add-on "ITIL Practices" supplements this reference data with the following values:

- **Ticket Types:**
 - Problem
 - Change
 - Change (Emergency)
 - Incident (Major)
 - Incident (Security)
 - Business Continuity Preparation

- **Ticket Status:**
 - approved (open)
 - assigned (open)
 - implemented (open)
 - ready for implementation (pending)
 - in process (pending)
 - planned (open)
 - solved (open)
 - known error (open)
 - in validation

10.2.4.7 Report Definitions for "ITIL Practices"

The "Reports" module enables the creation of individual reports for the evaluation of key figures.

The report definitions can be created on the basis of SQL Select queries. Agents (with the role "Report User") can then access these report definitions and independently generate reports in the desired output format.

Basic information on this topic can be found in the Admin Manual KIX Start in the chapter "Reporting".

In addition to the initially delivered report definitions, the add-on "ITIL Practices" contains further report definitions for the evaluation of important ITIL®4 key indicators:

- Number of Accounted Time: Created In Date Range for User and Team per Ticket
- Number of Tickets: Created Per Type and Priority
- List of Assets: Changed in Date Range per Class and IncidentState
- List of Tickets: SLA Violation In Date Range per Organisation and State
- Number of Assets: Per Class and IncidentState
- Number of Tickets: Closed In Date Range of Type "Incident" per CloseCode
- Number of Tickets: Closed In Date Range per SatisfactionPoints
- Number of Tickets: Created In Date Range of Type "Problem" per ProblemState
- Number of Tickets: Created In Date Range per Service and Impact
- Number of Tickets: SLA Violation In Date Range per Organisation and State

10.2.4.8 Templates for "ITIL Practices"

The add-on "ITIL Practices" brings along an extra configured ticket template for each process type (service request, malfunction, problem, change request, emergency plan):

- Service Request (Service Anfrage) - **Agent**
- Service Request (Service Anfrage) - **Customer/SSP**
- Problem
- Incident (Störung) - **Agent**
- Incident (Störung) - **Customer/SSP**
- Incident - Security (Sicherheitsrelevante Störung)
- Incident - Major (Schwerwiegende Störung)
- Change (Änderungsantrag) - **Agent**
- Change (Änderungsantrag) - **Customer/SSP**
- Change - Emergency (Änderungsantrag - Notfall)
- Business Continuity Preparation

These templates are all assigned to the template group "IT orders".

You can change the configuration of the ticket templates, e.g. to change permissions or to integrate further dynamic fields. The administration of ticket templates can be found under *KIX > Workflow > Templates*.

Attention!

The "**Process state**" is a dynamic field of the checklist type. Many jobs, actions, etc. are linked to it. Therefore, **do not remove** the processing status from the ticket templates and **do not change** the configuration of the dynamic field! Otherwise, serious errors may occur in the process flows!

Info

When saving, the status, team, priority, etc. are automatically set on the ticket. These values result from the default values stored in the system. If required, you can change the default values in the SysConfig keys *Ticket::Queue::Default*, *Ticket::State::Default*, *Ticket::Priority::Default*.

10.3 Maintenance Plan

The add-on "KIX Maintenance Plan" supports you with recurring maintenance tasks. It enables you to manage recurring tasks for the devices, contracts or operating equipment documented in the asset management.

Main use scenarios are:

- Planning, execution and documentation of maintenance
- Reminder/renewal of contracts, certificates, approvals and statutory inspection periods, e.g.:
 - regular inspections according to BGV/GUV-V A 3
 - regular inspections of machines, vehicles, production facilities
- Representation of non-availabilities of operating resources
- Automatic creation of task descriptions and maintenance orders
- Digital equipment logbook/object life history

Based on your freely defined maintenance plans, the add-on generates the upcoming maintenance tasks and creates the corresponding tickets when they are due. Clear lists and calendars inform you about upcoming, planned and completed maintenance tasks. In this way, you keep track of all maintenance tasks and can perform them according to the due date and fully documented.

10.3.1 Prerequisite

As a basis for using the add-on "KIX Maintenance Plan", you need KIX 18 Pro, which runs at least on version 29.

Content on this page:

- [Prerequisite](#) (see page 357)
- [Fundamentals](#) (see page 358)
 - [Maintenance service](#) (see page 359)
 - [Maintenance asset](#) (see page 359)
- [Maintenance plan](#) (see page 359)
 - [Mapping](#) (see page 359)
 - [Scheduling](#) (see page 359)
 - [Ticket templates](#) (see page 360)
- [Maintenance tasks](#) (see page 360)
- [Maintenance ticket](#) (see page 361)
- [Administration](#) (see page 361)
 - [Installation/system update](#) (see page 362)
 - [Authorisation roles](#) (see page 362)
 - [Ticket templates](#) (see page 362)
 - [Update maintenance tasks](#) (see page 364)
 - [Functional enhancements](#) (see page 364)

10.3.2 Fundamentals

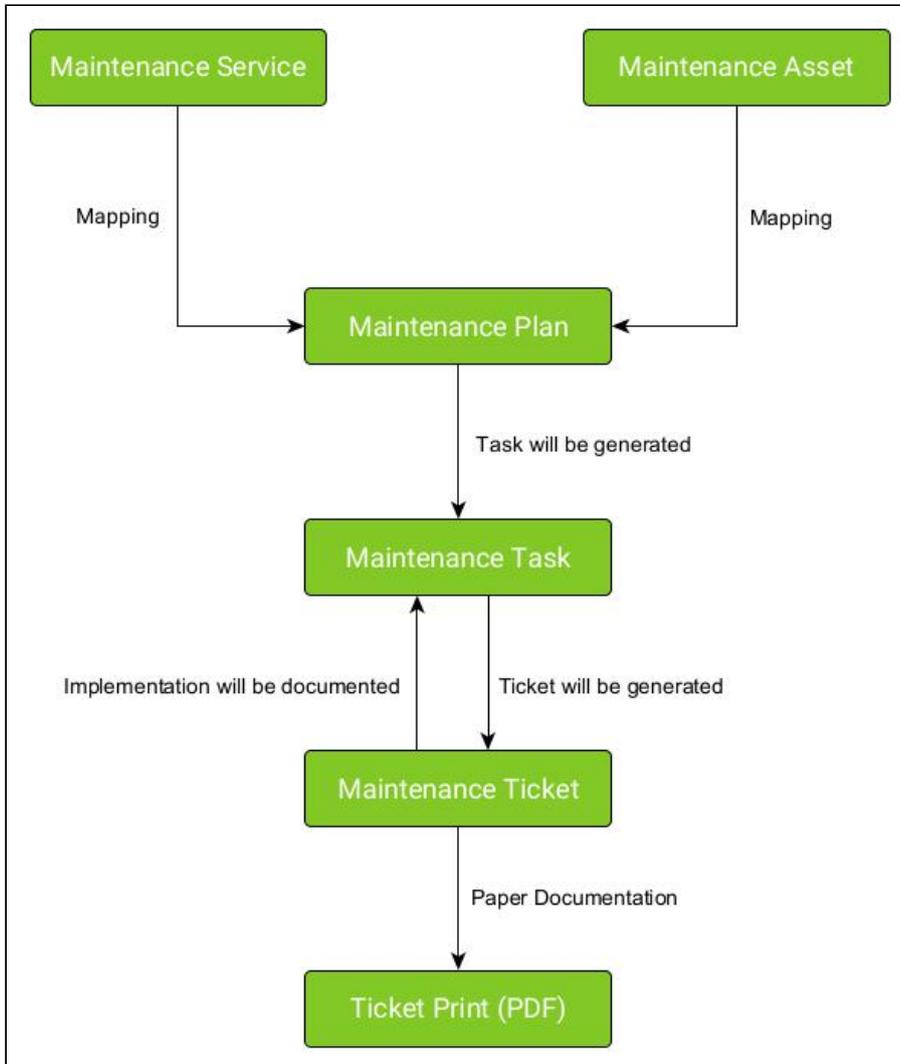


Fig.: Flow chart add-on "KIX Maintenance Plan"

Each maintenance is a service in the service catalogue. Therefore, the maintenance services are created in the asset class "Service" so that they are subsequently available in the explorer of the "Maintenance Plan" module.

In the "Maintenance Plan" module, the maintenance plans are created and the maintenance services are linked to the corresponding assets. KIX creates the resulting maintenance tasks once a day based on the defined maintenance plans.

The maintenance plan defines, among other things, at which point in time the tickets for the maintenance to be carried out are to be created. At this time, KIX generates the tickets to perform the maintenance. Administrators can freely configure the ticket templates to be used by the system for this purpose.

The execution of the maintenance is documented at the maintenance task by storing, among other things, the respective status and the data on due date and completion.

10.3.2.1 Maintenance service

A maintenance service defines *WHAT* the maintenance includes. It describes the service to be performed on the asset (e.g. grease all rotating gear parts).

Each maintenance service is an asset of the asset class "Service" and therefore part of the service tree. Therefore, maintenance services are created as assets of the asset class "Service".

After creating a maintenance service, you may have to execute the following console commands (menu "System > Console"), so that the maintenance service is displayed in the service tree of the module "Maintenance Plan":

- `Console::Command::Maint::Service::UpdateServiceCatalog`
- `Console::Command::Maint::Cache::Delete`

10.3.2.2 Maintenance asset

A maintenance asset defines the *WHERE* of maintenance. Maintenance assets are all assets that are subject to (regular) maintenance. This can be any asset in the system, but also a service (e.g. life cycle check).

10.3.3 Maintenance plan

The maintenance plan is the heart of the planning. It defines the *HOW* of maintenance. Here the mapping (linking) of maintenance service and maintenance asset takes place and it is determined *WHAT WHERE* and *WHEN* is planned. Maintenance plans are created and managed in the "Maintenance plan" module.

10.3.3.1 Mapping

In the simplest case, a maintenance plan is characterised by the fact that a maintenance service and a maintenance asset exist and are assigned to each other. The assignment can take place through

- explicit assignment of a maintenance service to a concrete asset
- Assignment of a maintenance service to various assets that fulfil certain conditions (e.g. all assets of the class Computer, of the manufacturer XYZ, in operation since 01.01.2023).

Due to this flexibility, future assets that fulfil the criteria are also taken into account in the maintenance plan without having to adjust the maintenance plan.

10.3.3.2 Scheduling

In the maintenance plan, the schedule can also be defined:

- how often the maintenance should be carried out (periodicity)
- the duration of the advance planning (Projection Days)
- the time of ticket creation (Planning Days)
- the validity period of the maintenance plan
- the planning type
 - Shedule Based
 - Completion Based

10.3.3.3 Ticket templates

When setting up the maintenance plan, the ticket template to be used can be selected. The ticket template is used by the system to generate the maintenance tickets for scheduling the maintenance task at the given time. According to the template configuration, the assigned team, the contents of the task description, FAQ references, etc. are thus defined for the implementation of the maintenance (see also section below).

10.3.4 Maintenance tasks

KIX checks the list of maintenance plans once a day. It checks which tasks exist for which maintenance plans within the next n days. As a result of this check, the maintenance tasks are determined and created. The maintenance tasks are displayed in the "Maintenance plan" module (list and calendar) as well as on the respective assets and services. This check can also be triggered manually at any time, e.g. after changes to a maintenance plan.

Maintenance Tasks (5)									
<input type="checkbox"/>	Service	State	Plan Due Date	Completion Due D...	Ticket Number	Ticket State	Ticket Title	Completion Date	
<input type="checkbox"/>	Service 1	canceled	06/23/2023	07/07/2023	2023063017000028	closed	Maintenance Service ...		
<input type="checkbox"/>	Service 1	done	06/16/2023	06/30/2023	2023063017000019	closed	Maintenance Service ...	06/30/2023, 12:58 PM	
<input type="checkbox"/>	Service 1	planned	06/30/2023	07/14/2023	2023063017000037	new	Maintenance Service ...		
<input type="checkbox"/>	Service 1	projected	07/07/2023	07/21/2023					
<input type="checkbox"/>	Service 1	projected	07/14/2023	07/28/2023					

Fig.: List of maintenance tasks on the asset

Note

Please note: Maintenance tasks are not tickets! Maintenance tasks only inform about planned maintenance and serve as maintenance documentation. The tickets for performing the maintenance will be created at a later time (as defined in the maintenance plan).

The maintenance tasks are the core element of the maintenance documentation (maintenance history). All information is brought together here:

- the concrete asset that is/was maintained
- the concrete maintenance service that is/was carried out
- the date of the planning due date
 - This is the date on which the ticket should be created.

- the planned execution date
 - This is the date on which the maintenance should be carried out.
- the actual execution date
 - The time when the maintenance was actually performed.
- the status of the maintenance task
 - projected: Created maintenance task that is expected to result in a maintenance ticket.
 - planned: Task for which a maintenance ticket has already been created.
 - done: Maintenance ticket has been closed (not cancelled).
 - cancelled : Maintenance ticket has been cancelled.

10.3.5 Maintenance ticket

Maintenance tickets are tickets that the system generates using the information stored in the maintenance plan. They are used to schedule and then perform the maintenance task.

The mechanism for checking and creating maintenance tasks also checks the due date of the maintenance tasks and creates the associated maintenance tickets a few days before the due date (as defined in the maintenance plan).

Maintenance tickets are displayed

- at the respective asset
- at the respective service
- in the Maintenance Plan module (calendar and list)
- in the ticket dashboard

After creating the maintenance ticket, the status of the maintenance task changes from "projected" to "planned".

After the maintenance has been performed, the ticket can be closed. When the ticket is closed, the selected completion code and - if "completed" - the completion time are written back to the maintenance task.

For the documentary documentation of the maintenance, the maintenance ticket can be downloaded as a PDF and printed and filed if required.

10.3.6 Administration

The administrative effort to set up and maintain the add-on "KIX Maintenance Plan" is low. The only tasks to be carried out by the administrator are

- Installation (if KIX is used as an on-premises solution)
- Assigning the authorisation roles
- Set up one or more ticket templates on the basis of which the maintenance tickets are created.

The creation of assets and services as well as the setup of maintenance plans is done by the user in the modules Assets or Maintenance Plan. A detailed description can be found in the [user manual \[LINK\]](#).

10.3.6.1 Installation/system update

After your order, we will provide you with an updated image of your system - usually on the next working day.

If you are running KIX Pro on-premises, please perform a system update afterwards:

```
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./stop.sh  
user@DockerHost:/opt/kix-on-premise/deploy/linux# ./update.sh
```

After this, the add-on is integrated into your system and is available as an additional module in the left

module menu .

The deployment of the add-on in the KIX Cloud is done by our support team.

10.3.6.2 Authorisation roles

Additional, supplementary authorisation roles are delivered with the "KIX Maintenance Plan" add-on. These are additionally granted to the users concerned.

- Maintenance Reader:
 - can view but not edit maintenance plans
 - additionally requires the roles Agent User, Ticket Reader and Asset Reader
- Maintenance Manager:
 - can view, edit and deactivate maintenance plans
 - additionally requires the roles Agent User, Ticket Agent, Asset Maintainer

10.3.6.3 Ticket templates

The ticket template is used by the system to generate the maintenance tickets at the time of the planning due date so that the execution of the maintenance task can be scheduled. The time of the planning due date is stored in the maintenance planner.

Depending on the configuration, the assigned team, the contents of the task description, FAQ references and other information for the implementation of the maintenance are defined in the template.

In order for the template to be used by the add-on "KIX Maintenance plan", it must be assigned to the **Usage context "System"** as well as the **Behaviour "Maintenance"**.

A separate ticket template can be configured for each maintenance plan. Several maintenance plans can also use a common ticket template.

The configuration of the ticket template is done exactly as for all other templates in the Admin module in the menu "*Workflow > Templates*". Which input fields are to be included in the future tickets and with which data the ticket is initialised can be freely configured in the template. The use of KIX placeholders in the template is possible (e.g.: <KIX_TICKET_DynamicField_AffectedAsset_Object_0_Attribute_0_Value>).

Assets are often also linked to FAQ entries. So that these FAQ entries are also available on the maintenance ticket, the dynamic fields have been extended by the field type "FAQ Reference". This allows you to create templates that contain references to FAQ entries. This also serves to make FAQ entries available in the Field Agent App.

If no or an incomplete ticket template is specified in the maintenance plan, the following attribute values are set on the ticket:

Attribute	Value
Title	"Maintenance <name of maintenance service> - <name of maintenance facility>"
Status	according to SysConfig key " <i>Ticket::State::Default</i> "
Team	according to SysConfig key " <i>Ticket::Queue::Default</i> ".
Priority	according to SysConfig key " <i>Ticket::Priority::Default</i> "
Type	Maintenance
Article/Contant	Without template use, this information remains blank

Regardless of the specifications in the ticket template, the following are always set on the ticket:

Attribute	Value
Affected Service	the corresponding maintenance service
Planned Effort	Effort from service
Plan Begin	Start date according to maintenance plan
Plan End	End date according to maintenance plan
Type	Maintenance

10.3.6.4 Update maintenance tasks

Configuration key	Daemon::SchedulerCronTaskManager::Task###UpdateTaskList
--------------------------	---

The add-on "KIX Maintenance Plan" contains an automatism that performs a daily update of all maintenance services (Daemon-Cron-Task + Console Command). This update usually takes place at 04:00. If necessary, you can deactivate this update in the SysConfig key

"Daemon::SchedulerCronTaskManager::Task###UpdateTaskList".

10.3.6.5 Functional enhancements

The add-on "KIX Maintenance Plan" adds the following entries to KIX:

- Addition of the value "Maintenance (periodic)" to the General Catalog class "ITSM::ConfigItem::Service::Type".
- Additional authorisation roles "Maintenance Reader" and "Maintenance Manager" (in the context "Agent")
- Additional ticket type "Maintenance".



11 Practice

The following chapters contain a collection of practical examples, possible uses and exemplary use cases. With this service, we want to provide you with a range of information to help you understand KIX a little better, to identify possible applications and to whet your appetite for trying it out.

- [Periodic job "Licence renewal" \(see page 366\)](#)
- [Use of checklists \(see page 370\)](#)
 - [Configuring and providing checklists \(see page 376\)](#)
 - [Data structure of checklists \(see page 397\)](#)

11.1 Periodic job "Licence renewal"

In this use case, a job is created that generates a ticket as soon as the licence of a software asset expires within the next 1 to 3 weeks.

To do this, the job checks all assets of the "Software" class and checks the specified licence expiry date. If this date is within the next 1 to 3 weeks, a ticket is generated and assigned to the responsible team. The ticket contains the affected asset as well as its name and ID so that the agent can recognise at a glance which asset it is.

Note: This use case can only be mapped with KIX Pro, as no jobs can be executed on assets in KIX Start.

11.1.1 Preparation

1. The licence expiration date must be stored for the relevant assets in the "Software" class, as this date is checked in the job.

Therefore, if necessary, add the licence expiry date for one or more software assets.



2. The attribute "Expiration Date" is initially not searchable and is therefore not available for selection in the job filters. The asset class definition of the "Software" class must therefore be adjusted:
 1. In the Admin module, navigate to *KIX > Assets > Asset Classes*
 2. Click on the "Software" class to open the detailed view and then click on "Edit".
The "Edit asset class" dialogue opens.
 3. Enlarge the "Class Definition" text field to make it easier to edit the code.
Optionally, you can copy the code into a (text) editor of your choice.
 4. Use the search function of your browser or editor and search for the "Expiration Date" attribute.

5. Place a comma after the last line and add the parameter `'Searchable' => 1`.
This sets the "Expiration Date" attribute to searchable.

```
* Class Definition: ⓘ
'CountDefault' => 0,
'CountMax' => 1,
'CountMin' => 0,
'CustomerVisible' => 0,
'Input' => {
  'Required' => 1,
  'Type' => 'Date',
  'YearPeriodFuture' => 10,
  'YearPeriodPast' => 20
},
'Key' => 'ExpirationDate',
'Name' => 'Expiration Date',
'Searchable' => 1
}
```

6. Save the change.

11.1.2 Configure job

1. Job Information

1. Job type: Asset

i A ticket is created when the job is executed, but as the job is applied to assets, "Asset" must be specified as the job type.

2. Name, comment: Fields can be filled in individually.
3. Validity: valid

2. Execution schedule

Configure a time of your choice. Time-controlled execution is recommended, e.g.

1. Weekday(s): Monday
2. Time: 10:00 (clock)

3. Filters

Set the filters to determine the software assets whose licence expires within the next 1 to 3 weeks.

1. Class - contained in - Software
2. Expiry date - within - the next - 1 - weeks - until - the next - 3 - weeks
(If the "Expiry date" attribute cannot be selected, it is not searchable (see Preparation))

4. Actions

1. 1st Action: Set variable

The asset number and name of the triggering asset are determined using a placeholder and saved together with text as a string in a variable.

i Variables can hold several attributes and form arrays. The combination of text and placeholders is also possible.

a. Variable: `varCurrentAsset`

b. Value: `A#: <KIX_ASSET_Number> (<KIX_ASSET_Name>)`

2. 2nd Action: Execute macro

a. Macro: Ticket

i. Action: Create ticket

i You can change the following parameters as required.

A. Channel: Note

B. Item text:

*The software licence will expire within the next 14 days.
Please renew the licence.*

*Asset: `${varCurrentAsset}`
(Asset ID: `${RootObjectID}`)*

C. Contact: Martin Mustermann

D. Organisation: My Organisation

E. Priority: high

F. Status: New

G. Title: `Software licence renewal required: $
{varCurrentAsset}`

H. Team: Service Desk

I. Dynamic Fields:

The variable `${RootObjectID}` determines the asset that triggers the job and sets this as the "Affected Asset" on the ticket.

I. Name: `Affected asset`

Value: `${RootObjectID}`

11.1.3 Execute job

The job is executed at the configured time and creates a ticket to extend the licence. The object triggering the job is set as the "Affected asset" (`{RootObjectID}`). This corresponds to the asset whose software licence needs to be extended.

In addition, the article text and the ticket subject contain the asset number and the name of the asset in question so that the agent can recognise at a glance which asset is involved. This information was saved in the variable " `varCurrentAsset` " and output in the appropriate place when the ticket is generated.

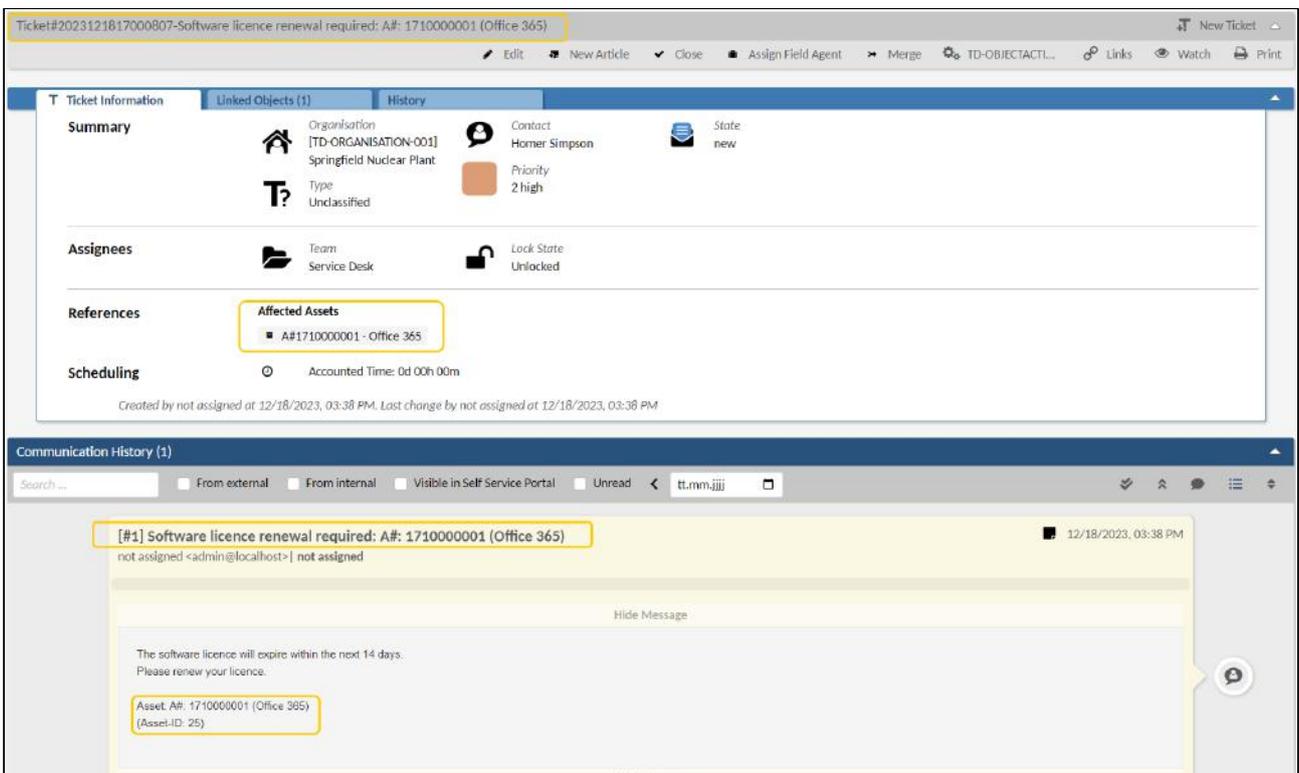


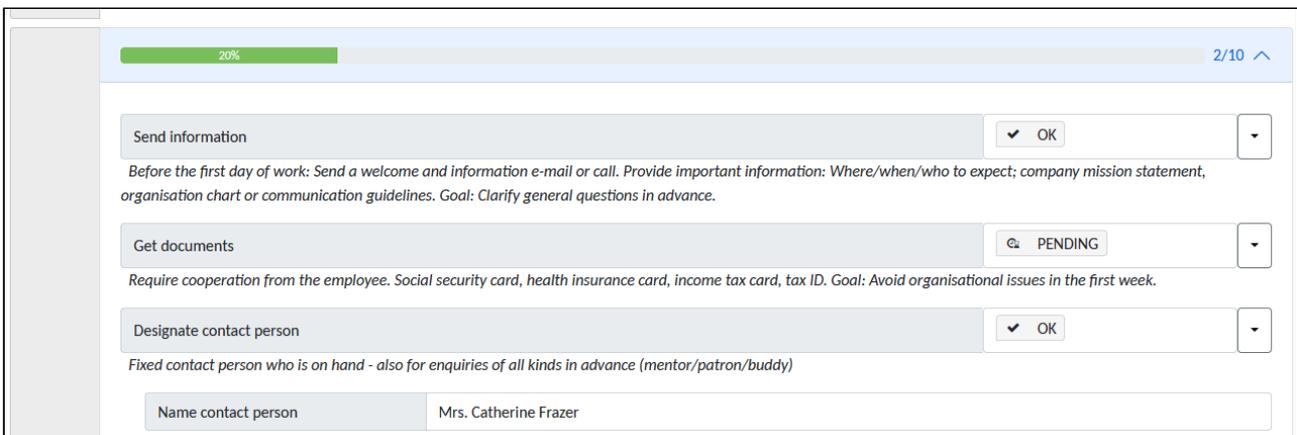
Fig.: A ticket created by the job

11.2 Use of checklists

Checklists are one of the most important tools in daily work and their uses are extremely diverse: They are used as shopping or to-do lists, but also for questionnaires or checklists. With KIX 18, you have free choice in creating the checklists you need, so that the right checklist can be configured for every application and purpose. The processing status of a checklist can be seen at any time by means of a progress bar.

Content on this page:

- [Checklists in practice](#) (see page 372)
- [Checklists in KIX Pro](#) (see page 373)
- [Configuration of checklists](#) (see page 373)
- [References](#) (see page 374)



The screenshot shows a checklist interface with a progress bar at 20% and a '2/10' indicator. It contains three checklist items:

- Send information:** Status 'OK'. Description: "Before the first day of work: Send a welcome and information e-mail or call. Provide important information: Where/when/who to expect; company mission statement, organisation chart or communication guidelines. Goal: Clarify general questions in advance."
- Get documents:** Status 'PENDING'. Description: "Require cooperation from the employee. Social security card, health insurance card, income tax card, tax ID. Goal: Avoid organisational issues in the first week."
- Designate contact person:** Status 'OK'. Description: "Fixed contact person who is on hand - also for enquiries of all kinds in advance (mentor/patron/buddy)". Below this is a text input field with the value "Mrs. Catherine Frazer".

Fig.: Part of a checklist in the ticket form

In KIX 18, any number of checklists can be created by the administrator and made available on the ticket. In the course of ticket processing, agents can then work through these checklists partially or completely. If the ticket is forwarded to another agent or team, the processing of the checklist can be continued there by these agents.

Each checklist consists of any number of freely defined list items. Each list item can in turn contain further list items subordinate to it. The individual list items can be processed in any order and at any time, as they are stored directly on the ticket. Each list item consists of a short title for naming ¹ and an optional note text for further explanation of the list item ². This avoids ambiguities and questions when working through the checklist.



This close-up shows the 'Designate contact person' checklist item. Callouts highlight the following elements:

- 1:** The title 'Designate contact person'.
- 2:** The descriptive note text: "Fixed contact person who is on hand - also for enquiries of all kinds in advance (mentor/patron/buddy)".
- 3:** The text input field containing 'Mrs. Catherine Frazer'.
- 4:** The dropdown menu showing 'OK'.

Fig.: List items of a checklist

A list item can be either a text field ³ or a selection field (dropdown) ⁴.

Text field: Text fields are used to record notes, answers, reasons or hints. Depending on the requirements, single-line or multi-line text fields can be used. They are used, for example, to

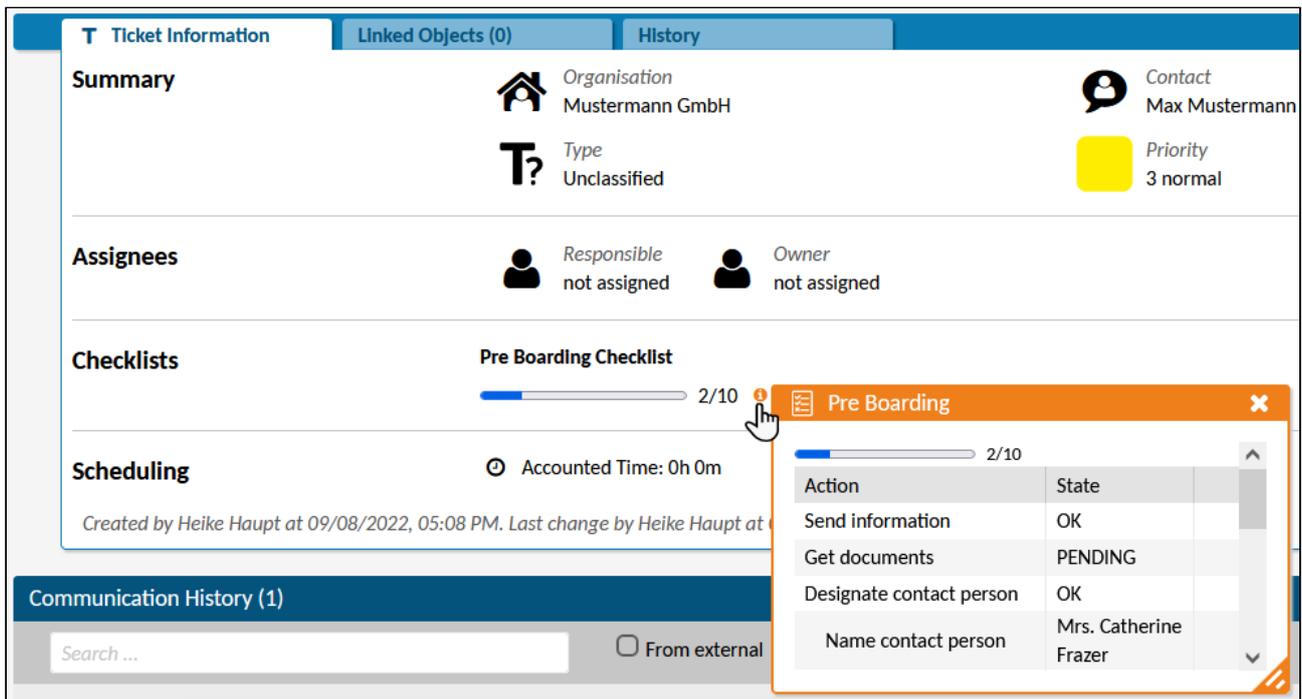
- the respondent's answers during a telephone survey
- the reasons for a refusal in the approval process
- notes for the service technician on the repair or installation order
- names or telephone numbers
- and other notes

in writing.

Selection field: Selection fields (dropdown) are used to select and indicate the current state. The status indicates the processing status or the result of the respective list item. Different states can be selected, which in their entirety influence the processing progress of the entire checklist.

State	Meaning	Progress
OK	Okay / Done / Approved	is counted up, since valid answer
NOK	Not OK / Not executed / Not approved	is counted up, since valid answer
PENDING	Pending status / Waiting for feedback / On hold	will not be counted up, as floating state
n.a.	no information / information not required / no statement made	is counted up, since e.g. "no statement made" is also a valid answer
-	unanswered / no selection made at the checklist item	will not be incremented

The progress of a checklist can be visualised by means of a progress bar and/or a numerical value (2/10) in the ticket detail view as well as in the dashboard tables. In addition, the ticket detail view displays the info symbol for the progress bar, which calls up the details on the current processing status when clicked.



The screenshot shows the 'Ticket Information' tab with the following details:

- Summary:** Organisation: Mustermann GmbH; Contact: Max Mustermann; Type: Unclassified; Priority: 3 normal.
- Assignees:** Responsible: not assigned; Owner: not assigned.
- Checklists:** Pre Boarding Checklist (2/10).
- Scheduling:** Accounted Time: 0h 0m.

The 'Pre Boarding' checklist pop-up window contains the following table:

Action	State
Send information	OK
Get documents	PENDING
Designate contact person	OK
Name contact person	Mrs. Catherine Frazer

Fig.: Checklist progress in the ticket detail view

11.2.1 Checklists in practice

In KIX, any number of individually configured checklists can be created and made available in the ticket interfaces. This allows a wide variety of scenarios to be mapped, for example:

1. The checklist for service technicians as a guideline for complete service provision to guarantee compliance with quality standards.
The service technician can work through the checklist point by point, answer it and, if necessary, note down a comment on individual points.
2. The checklist for mapping the tasks to be completed in an onboarding process.
If the ticket passes through several departments, each department can document and tick off its work steps. The current status of the process progress is thus immediately visible to every ticket processor.
You can find a configuration example of such a checklist here: [Configuration and provision of checklists](#) (see page 376)
3. The checklist as a discussion guide or questionnaire for support or sales.
These targeted questions enable agents to localise and specify a reported damage or malfunction.
4. The checklist for listing all required documents and verifications that are delivered together with a machine or technical system produced by you.
This ensures that all required documents and evidence such as product data sheets, safety

instructions, supplier documentation, etc. are enclosed with the product and that the legal requirements are met.

5. The checklist for recording the process steps of approval procedures or in order management. By ticking off the individual steps and making notes, it is ensured that nothing is overlooked and why which decisions were made.
An application example for order management can be found in the KIX Start - User Manual under: Practice > The order management.

11.2.2 Checklists in KIX Pro

In combination with the extended functions of KIX Pro, checklists are also used to provide individually configured ticket [templates](#) (see page 191) for both the Agent Portal and the Self Service Portal. This allows you to provide a variety of application-specific ticket forms: e.g. a ticket form for orders, another ticket form for fault reports, another ticket form for onboarding etc.... You can configure each one of these ticket forms to contain both the input fields corresponding to the use case and one or more application-specific configured checklist(s). This is possible with KIX Pro 18.

Likewise, a variety of ticket [actions](#) (see page 137) can be created in KIX Pro and - taking permissions into account - made available at the ticket. These can also contain checklists whose configuration is specifically adapted to the function of the respective ticket action. Ticket actions are always called up when the situation requires it. For example, an additional questionnaire can be added to an incident ticket if necessary, if it is a case of average.

Several ticket actions and checklists can be applied to each ticket. For example, an order ticket can be used by the

- the "Purchasing" team can use its purchasing checklist with the products to be ordered on the ticket and then
- the "Sales" team fills in its checklist with the products and documents to be delivered and finally
- the "After-Sales Management" team use a questionnaire on customer satisfaction.

The combined use of ticket templates and ticket actions is also possible in KIX Pro, whereby both can contain differently configured checklists.

11.2.3 Configuration of checklists

Checklists in KIX are dynamic fields of the type "Checklist". As with all dynamic fields, the data filled out in a checklist is always bound to the respective ticket in which the data was entered.

The configuration of checklists is done by the administrator in the module "Admin" in the menu *System > Dynamic Fields*. The provision of a checklist in the interfaces is done - as with any other dynamic field - by adjusting the corresponding configuration keys. KIX Start users use the configuration keys in the *System > SysConfig menu*. KIX Pro users can use the integrated JSON editor in the *System > GUI Configuration > Agent Portal* menu. This allows a more comfortable editing of configuration keys. A list of the most important

configuration keys including a complete application example for creating and providing a checklist can be found under: [Configuration and Provision of Checklists](#). (see page 376)

After deployment, the appearance and behaviour of the checklist depends on the context in which it was included:

- In the "new ticket" and "edit ticket" dialogues, it is displayed as an editable list.
- In the ticket details and dashboard tables, it is displayed - depending on the selected component - as a progress bar or as a numerical value (2/10).

In addition, agents can individually switch the display of the checklist status in a dashboard table on or off via the customisation of their Home Dashboard. A description of how to personalise the Home Dashboard can be found in the KIX Start User Manual (Start) under: Home Dashboard > Personalise your Home Dashboard.

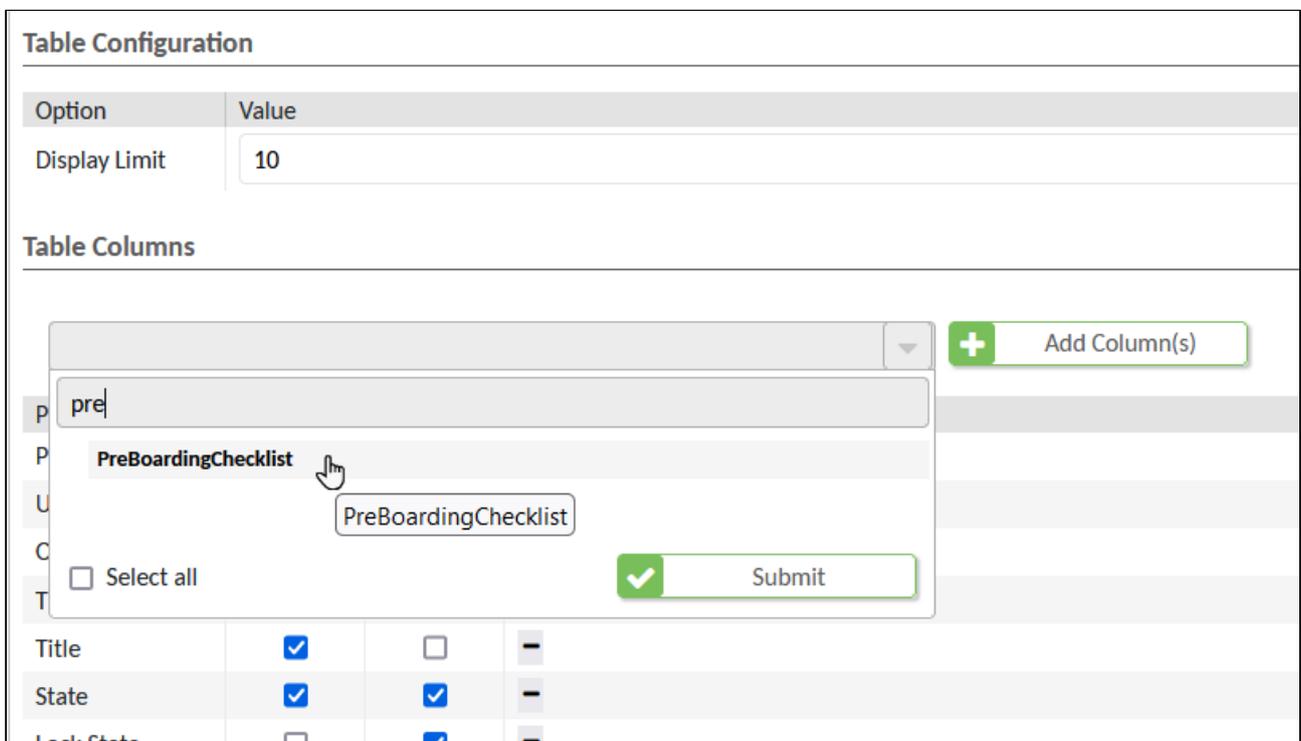


Fig.: Adding checklists to the Home Dashboard

11.2.4 References

In the admin manuals of KIX Start and KIX Pro you will find further information on dynamic fields, the field type "checklist" as well as on ticket templates and ticket actions:

- Dynamic Fields
- Object and field types of dynamic fields
- Integrating a Dynamic Field



- Displaying Values of Dynamic Fields
- Configuration of dashboard tables
- [Ticket actions \(see page 137\)](#)
- [Ticket Templates \(see page 191\)](#)
- Application example: [Configuring and providing checklists \(see page 376\)](#)

11.2.5 Configuring and providing checklists

Checklists in KIX are lists with predefined tasks, which can be processed by the agents at the time of their choice, either completely or partially. Administrators can create any number of differently configured checklists in the Admin module under *System > Dynamic Fields* and then make them available in ticket dialogues, in the ticket zoom view and/or in dashboard tables via the configuration of the user interface.

The following complete example shows how a checklist is configured and integrated into the different interfaces. The application scenario is a pre-boarding process that maps the tasks to be completed when hiring new employees (exemplary short form).

The following steps are carried out in the example:

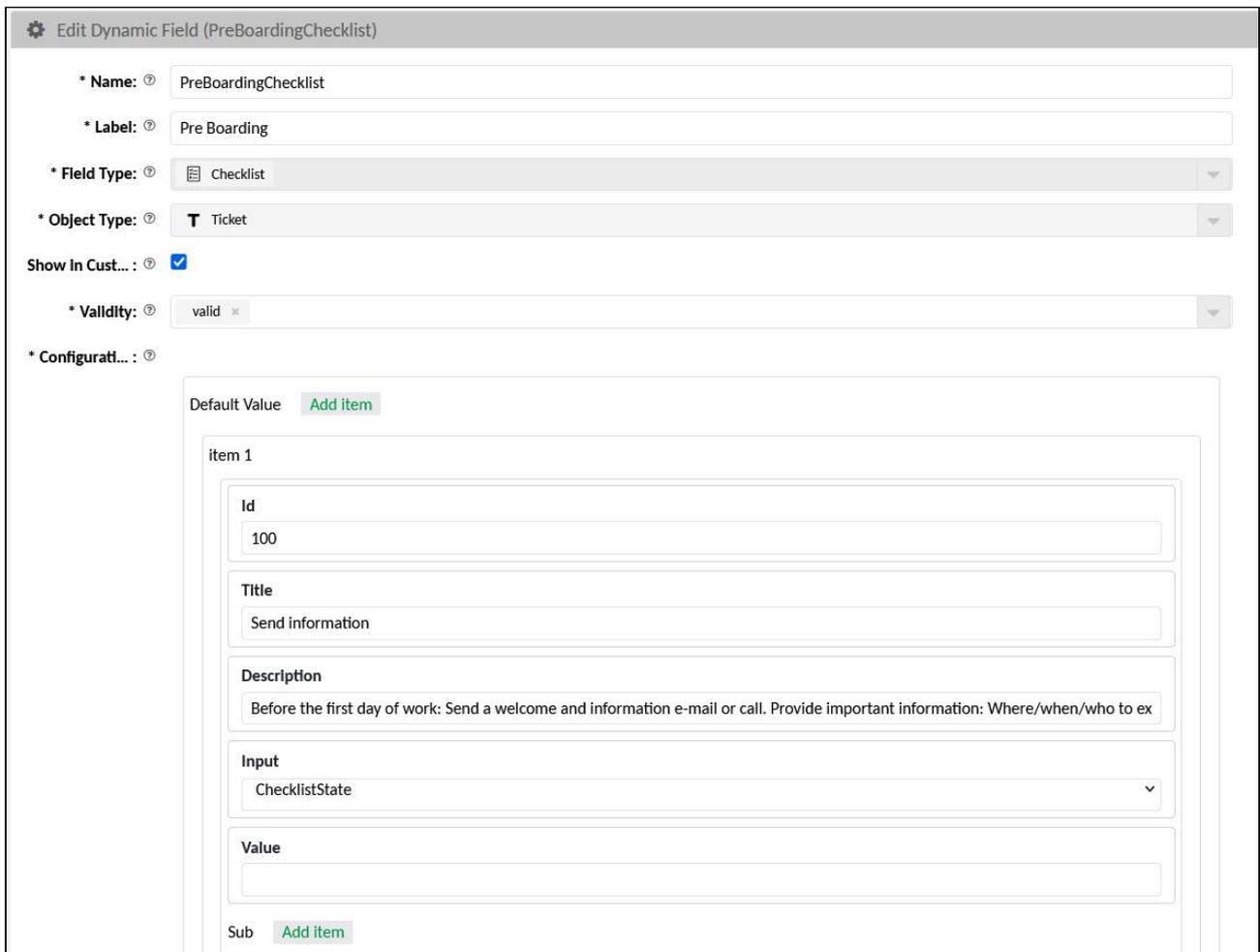
1. **Create checklist:** Create and configure a dynamic field of the type "Checklist".
2. **Provide checklist:** Integration of the checklist into the dialogues
 1. "New ticket"
 2. "Ticket edit"
3. **Display** of the checklist status
 1. in the ticket zoom view
 2. in the Home Dashboard

Info: Each checklist created in the system is an independent dynamic field. You are therefore free to choose whether and where you want to provide or display the checklist and therefore also to use only parts of this example.

11.2.5.1 1. Create a checklist

A checklist in KIX is a dynamic field of the type "Checklist". It is created and configured like any dynamic field in the menu *System > Dynamic Fields*. The configuration of a checklist consists of the basic data such as name, label, field and object type and the configuration of the individual list items. Each list item can contain further list items subordinate to it.

All list items must have a unique (alphanumeric) ID. It can be freely assigned, but may only occur once within a checklist. Each list item consists of a short title for naming and an optional description as a reference text to the list item. A list item can be either a single-line or multi-line text field or a selection list. Text fields are used to record notes, answers, justifications or hints. Selection fields are used to document the current state of a list item (OK, NOK, PENDING, n.a.). This specification is defined under Input. Furthermore, each list item can optionally be preset with a value, which can be changed by the agent at the time of use if necessary.



Edit Dynamic Field (PreBoardingChecklist)

* Name: PreBoardingChecklist

* Label: Pre Boarding

* Field Type: Checklist

* Object Type: Ticket

Show in Cust...:

* Validity: valid

* Configurati...:

Default Value [Add item](#)

item 1

Id: 100

Title: Send information

Description: Before the first day of work: Send a welcome and information e-mail or call. Provide important information: Where/when/who to ex

Input: ChecklistState

Value:

Sub [Add item](#)

Fig.: Checklist configuration

Configuration

For the example of the pre-boarding process, create the checklist as follows:

1. Navigate to the *System > Dynamic Fields* menu.
2. Click on "New Field".
3. In the dialogue that opens, create the checklist with the following basic data:
 1. **Name:** PreBoardingChecklist.
 ⓘ Make a note of the name. You will need it for the integration into the interfaces.
 2. **Label:** Pre Boarding Checklist
 3. **Field type:** Checklist
 4. **Object type:** Ticket
 5. **Show in Customer Portal:** no/disabled
 6. **Validity:** valid
4. Create an item for each list item.
 1. To do this, click on "Add Item" several times to add further list items to the checklist.

2. Configure the individual list items as specified under Item Configuration.
5. Finally, save your configuration. The checklist is now created and can be made available in the interfaces.

Item Configuration

The values given are based on the application example of the pre-boarding process. You can adapt the values to your specific needs.

Item	Sub-Item	ID	Title	Description	Input	Value
1		100	Send information	Before the first day of work: Send a welcome and information e-mail or call. Provide important information: Where/when/who to expect; company mission statement, organisation chart or communication guidelines. Goal: Clarify general questions in advance.	ChecklistState	---
2		200	Get documents	Require cooperation from the employee. Social security card, health insurance card, income tax card, tax ID. Goal: Avoid organisational issues in the first week.	ChecklistState	---
3		300	Designate contact person	Fixed contact person who is on hand - also for enquiries of all kinds in advance (mentor/patron/buddy)	ChecklistState	---
	1	310	Name contact person	---	Text	---

Item	Sub-Item	ID	Title	Description	Input	Value
4		400	Set up workplace	Workplace must be set up on the 1st day of work. This includes: Provision of passwords, printer access, telephone, office chair/desk, other aids. Goal: Avoid technical blockers in the first week.	ChecklistState	---
	1	410	Passwords provided	---	ChecklistState	---
	2	420	Printer access provided	---	ChecklistState	---
	3	430	Telephone provided	---	ChecklistState	---
	4	440	Office chair and desk provided	---	ChecklistState	---
	5	450	Other office supplies	---	TextArea	---
5		500	Create a familiarisation plan	Must be in place early, before work starts. Content: detailed work plan for first week; tasks; future projects. Goal: prevent chaos and idle time.	ChecklistState	---
	1	510	URL familiarisation plan	---	Text	---

Item	Sub-Item	ID	Title	Description	Input	Value
6		600	Organise flat search	Support for foreigners in finding accommodation, moving house or dealing with authorities. Goal: Avoid distractions, enable concentration on training.	ChecklistState	---

11.2.5.2 2. Provide checklist

After the dynamic field of the type "Checklist" has been created and configured, it can be made available in the desired interfaces.

Provision in KIX Start

Provisioning in KIX Start is done by configuring the corresponding SysConfig keys. Each context in which the field is included has its own configuration key. The most important ones are:

Configuration key	Configuration object	Presentation of the checklist
ticket-new-form-group-data	Dialogue "New ticket"	List items as input and selection fields
ticket-edit-form-group-data	Dialogue "Ticket edit"	List items as input and selection fields
ticket-details-info-card	Ticket zoom view	Progress bar/numerical value incl. details
home	Home Dashboard tables	Progress bar/numerical value without details

For easier editing, we recommend using an external JSON editor (e.g. www.jsonformatter.io⁹). 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.

⁹ <http://www.jsonformatter.io>

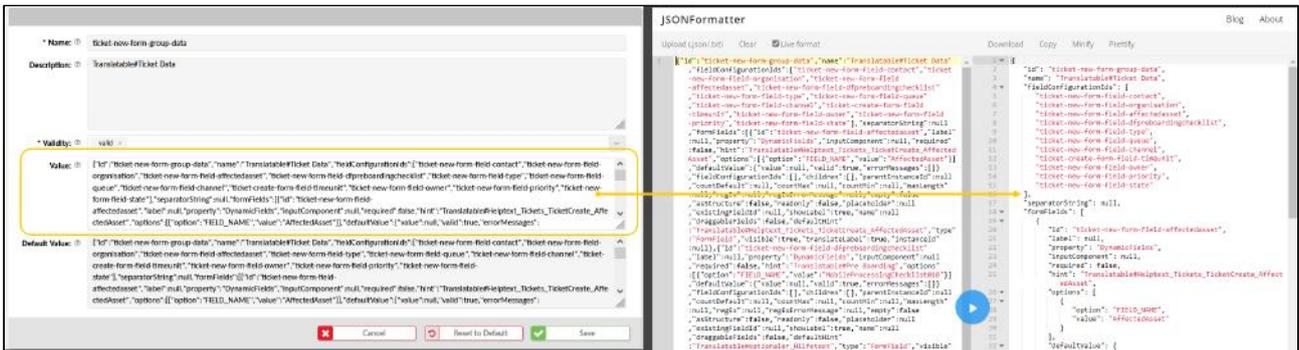


Fig.: Copy SysConfig key to external JSON editor

Provision in KIX Pro

In KIX Pro, dynamic fields are conveniently provided via [ticket actions](#) (see page 137) and [templates](#) (see page 191). The dynamic field only has to be selected in a dropdown in the respective template/action.

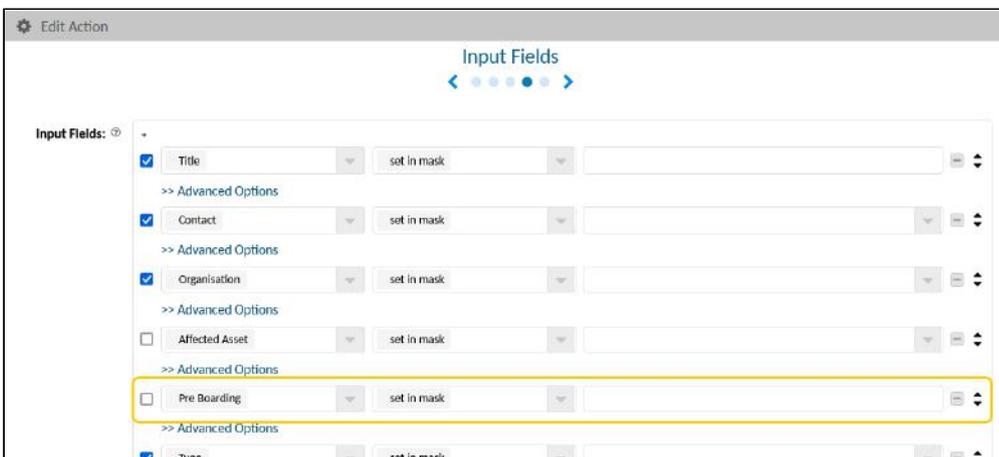


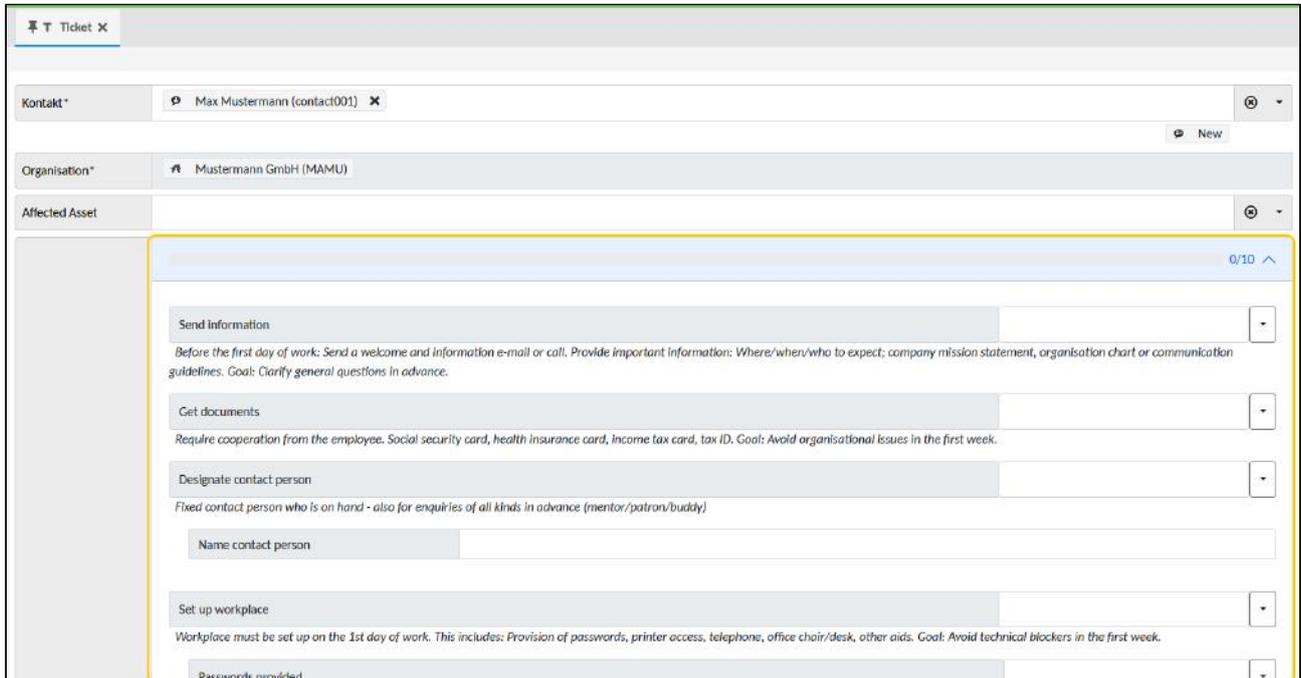
Fig.: Add and place dynamic field (checklist) of the ticket action "Edit ticket"

The configuration stored in the SysConfig key serves as a fallback. To edit a SysConfig key, KIX Pro users can use the integrated JSON editor in the *System > GUI Configuration > Agent Portal* menu. This eliminates the need to use the external JSON editor, including the steps to copy the value back and forth.

Procedure for KIX Start:

Provision in the "new ticket" dialogue

Configuration key:	ticket-new-form-group-data
---------------------------	----------------------------



The screenshot shows a 'Ticket' dialog box with the following fields and checklist items:

- Kontakt***: Max Mustermann (contact001)
- Organisation***: Mustermann GmbH (MAMU)
- Affected Asset**: (empty)
- Checklist items**:
 - Send information**: Before the first day of work: Send a welcome and information e-mail or call. Provide important information: Where/when/who to expect; company mission statement, organisation chart or communication guidelines. Goal: Clarify general questions in advance.
 - Get documents**: Require cooperation from the employee. Social security card, health insurance card, income tax card, tax ID. Goal: Avoid organisational issues in the first week.
 - Designate contact person**: Fixed contact person who is on hand - also for enquiries of all kinds in advance (mentor/patron/buddy). Includes a sub-field for 'Name contact person'.
 - Set up workplace**: Workplace must be set up on the 1st day of work. This includes: Provision of passwords, printer access, telephone, office chair/desk, other aids. Goal: Avoid technical blockers in the first week.
 - Passwords provided**

Fig.: The integrated checklist in the "new ticket" dialogue

Procedure for KIX Pro:

In KIX Pro, the dialogue "new ticket" is formed via the standard [template](#) (see page 191) "Default - New Ticket Template". Insert the checklist there as follows:

- Navigate to *Workflow > Templates* and open the template "Default - New Ticket Template" for editing.
- Go to step 2 "Input fields". At the bottom of the dialogue you will find a free selection field. Select the dynamic field "Pre Boarding".
- Optionally move the field via drag & drop to the desired position in the form and finally click on "Save". Afterwards, the checklist is included in the "New Ticket" dialogue.

Procedure for KIX Start:

To provide the checklist in the "new ticket" dialogue, proceed as described below. The checklist is then included in every new ticket.

1. Navigate to *System > SysConfig*. Open the key "ticket-new-form-group-data". This key defines the structure of the ticket creation mask.
2. Copy the source code from the field "Value" into a JSON editor. In the maximised view, you can edit the source code more easily.

3. Paste the following source code (without comments!) into the code block under " `formFields` :
[...]" .

Place it between 2 code sections. Each code section always begins and ends with a curly bracket.
Two consecutive code sections are separated by commas.

You can replace the values given under " `id` " and " `value` " with the designations you use.

```

1      {
2          "id": "ticket-new-form-field-dfpreboardingchecklist",    //
Unique ID of the field in the key
3          "label": null,
4          "property": "DynamicFields",
5          "inputComponent": null,
6          "required": false,           //Require field: yes/no
7          "hint": "Translatable#Pre Boarding",
8          "options": [
9              {
10             "option": "FIELD_NAME",
11             "value": "PreBoardingChecklist" //Name of the dynamic
field as created above
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#optional_help_text",
37     "type": "FormField",
38     "visible": true,
39     "translateLabel": true,
40     "instanceId": null
41 },

```

```

1  {
2    "id": "ticket-new-form-group-data",
3    "name": "Translatable#Ticket Data",
4    "fieldConfigurationIds": [{}],
16   "separatorString": null,
17   "formFields": [
18     {},
60     {},
86     {
87       "id": "ticket-new-form-field-dfpreboardingchecklist",
88       "label": null,
89       "property": "DynamicFields",
90       "inputComponent": null,
91       "required": false,
92       "hint": "Translatable#Pre Boarding",
93       "options": [
94         {
95           "option": "FIELD_NAME",
96           "value": "PreBoardingChecklist"
97         }
98       ],
99       "defaultValue": {
100        "value": null,
101        "valid": true,
102        "errorMessages": []
103      },
104       "fieldConfigurationIds": [],
105       "children": [],
106       "parentInstanceId": null,
107       "countDefault": null,
108       "countMax": null,
109       "countMin": null,
110       "maxLength": null,
111       "regex": null,
112       "regexErrorMessage": null,
113       "empty": false,
114       "asStructure": false,
115       "readonly": false,
116       "placeholder": null,
117       "existingFieldId": null,
118       "showLabel": true,
119       "name": null,
120       "draggableFields": false,
121       "defaultHint": "Translatable#optional_help_text",
122       "type": "FormField",
123       "visible": true,
124       "translateLabel": true,
125       "instanceId": null
126     }
127   ]
128 }

```

4. Add the ID of the dynamic field to the code block " `fieldConfigurationIds` " (see code block above). The order of the field IDs determines the order of the form fields in the ticket creation screen.

```
{
  "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-affectedasset",
    "ticket-new-form-field-dfpreboardingchecklist",
    "ticket-new-form-field-type",
    "ticket-new-form-field-queue",
    "ticket-new-form-field-channel",
    "ticket-create-form-field-timeunit",
    "ticket-new-form-field-owner",
    "ticket-new-form-field-priority",
    "ticket-new-form-field-state"
  ],
  "separatorString": null,
  "formFields": [
```

5. In the editor, minimise 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 KIX into the field "Value" and apply the changes with "Save".
7. Click on "Reload Frontend Configuration" so that the checklist is displayed in the "New Ticket" dialogue.



Provision in the "Edit ticket" dialogue

Configuration key	ticket-edit-form-group-data
--------------------------	-----------------------------

In order for the individual list items to be processed when editing the ticket, the checklist must also be made available in the "Edit ticket" dialogue.

Procedure for KIX Start: Open the SysConfig key "ticket-edit-form-group-data" and proceed as described above under "Provision in the 'new ticket' dialogue".

Procedure for KIX Pro: Navigate to *Ticket > Actions* and open the action "Ticket Edit" for editing. At the bottom of the opening dialogue you will find a free selection field. Select the dynamic field "Pre Boarding". Optionally, move the field to the desired position in the form via drag & drop and finally click on "Save". Afterwards, the checklist is included in the "Edit ticket" dialogue.

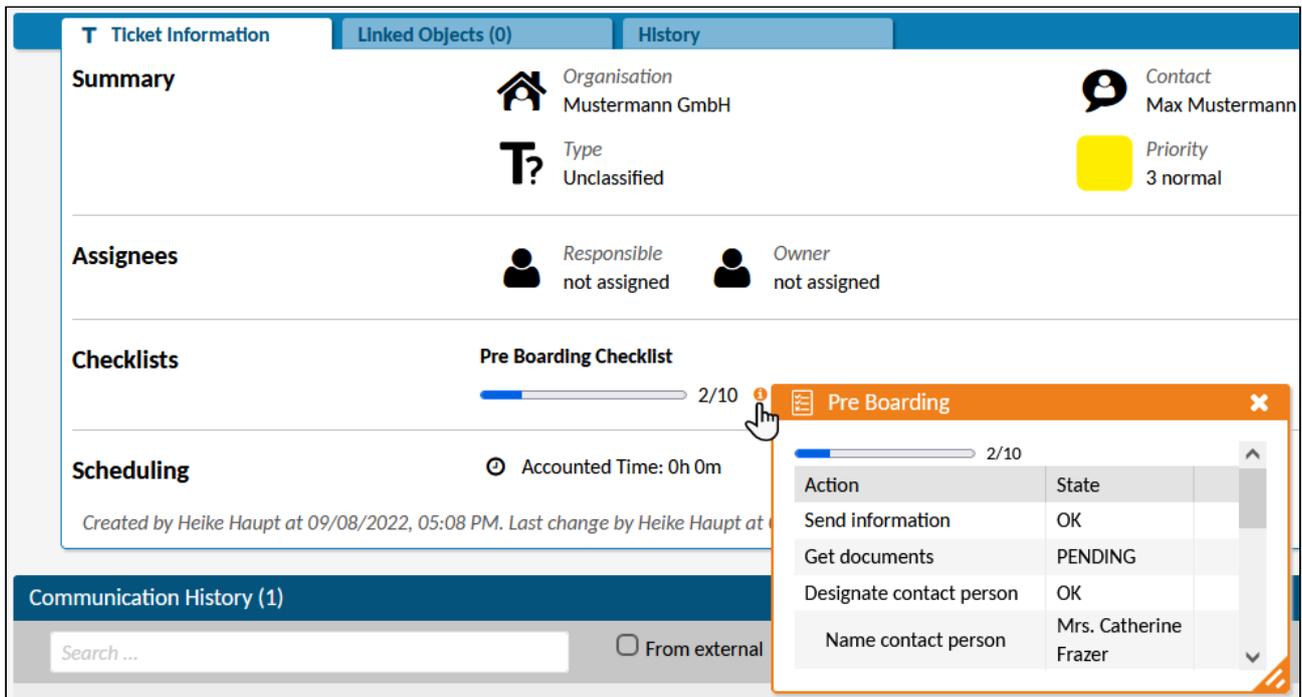
11.2.5.3 3. Show checklist status

The status of a checklist changes as it is processed. It can be visualised in the form of a progress bar. The progress bar can be provided according to the need in the different interface structures of KIX. The provision takes place in KIX Start and in KIX Pro via the configuration of the respective SysConfig keys. Each context has its own configuration key (see above).

For editing the configuration keys, we recommend that users of KIX Start use an external JSON editor. Users of KIX Pro can use the JSON editor in the menu *System > GUI Configuration > Agent Portal* (see above).

Display in the ticket zoom view

Configuration key:	ticket-details-info-card
---------------------------	--------------------------



Action	State
Send information	OK
Get documents	PENDING
Designate contact person	OK
Name contact person	Mrs. Catherine Frazer

Fig.: The checklist status in the ticket zoom view

Procedure:

1. KIX Start: Navigate to *System > SysConfig* and open the key "ticket-details-info-card". In this key, the display of the lane "ticket information" is configured.
KIX Pro: Navigate to *System > GUI Configuration > Agent Portal* and locate the key "ticket-details-info-card".

2. **KIX Start:** Copy the source code from the "Value" field into a JSON editor.
In maximised view, you can edit the source code more easily.

3. **KIX Start + KIX Pro:** Copy the following code block into a `values` block at the desired position. With its placement, you determine where the progress bar is displayed.
If necessary, replace the name of the dynamic field (PreBoardingChecklist) with the name you use.
Note: The specification of the `conditions` is optional. These define the conditions for displaying the checklist in the ticket zoom view. In the example, the checklist will not be displayed as long as no value is set for it.

```
[
  {
    "text": "Pre Boarding Checklist",
    "textStyle": "font-weight:bold;margin-bottom:0.5rem",
    "icon": "kix-icon-ci",
    "componentId": "dynamic-field-value",
    "componentData": {
      "name": "PreBoardingChecklist"
    },
    "conditions": [
      {
        "property": "DynamicFields.PreBoardingChecklist",
        "operator": "NE",
        "value": null,
        "useObjectService": false,
        "useDisplayValue": false
      }
    ]
  }
]
```

```

{
  "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": "1644499906890",
    "name": "1644499906890",
    "type": null,
    "avatar": [],
    "rows": [
      {
        "title": "Translatable#Summary",
        "style": "",
        "separator": true,
        "values": [
          [
            {icon: "ticket"},
            {icon: "ticket"}
          ],
          [
            {
              "componentId": "dynamic-field-value",
              "componentData": {
                "name": "PreBoardingChecklist"
              },
              "conditions": [
                {
                  "property": "DynamicFields.PreBoardingChecklist",
                  "operator": "NE",
                  "value": null
                }
              ]
            }
          ]
        ]
      }
    ]
  }
}

```

Variant:

You can create a separate section in the ticket zoom view that contains one or more checklists. To do so, use the following code block and insert it into the rows block at the desired position.

Display numerical value (2/10 - 2 of 10))

```

{
  "title": "Translatable#Checklists",
  "style": "",
  "separator": true,
  "values": [
    [
      {
        "text": "Pre Boarding Checklist",
        "textStyle": "font-weight:bold;margin-bottom:0.5rem",
        "icon": "kix-icon-ci",
        "componentId": "dynamic-field-value",
        "componentData": {

```

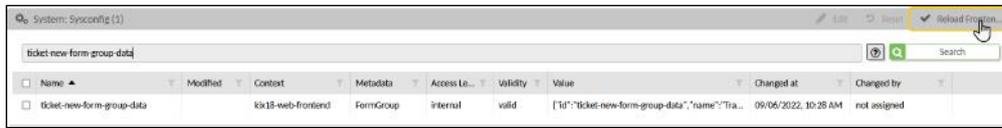
```
        "name": "PreBoardingChecklist"
    },
    "conditions": [
        {
            "property": "DynamicFields.PreBoardingChecklist",
            "operator": "NE",
            "value": null,
            "useObjectService": false,
            "useDisplayValue": false
        }
    ]
},
[
    {
        "text": "Optionale 2. Checkliste",
        "textStyle": "font-weight:bold;margin-bottom:0.5rem",
        "icon": "kix-icon-ci",
        "componentId": "dynamic-field-value",
        "componentData": {
            "name": "NameofChecklist"
        },
        "conditions": [
            {
                "property": "DynamicFields.NameofChecklist",
                "operator": "NE",
                "value": null,
                "useObjectService": false,
                "useDisplayValue": false
            }
        ]
    }
]
],
},
```

```

{
  "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": "1644499906890",
    "name": "1644499906890",
    "type": null,
    "avatar": [],
    "rows": [
      {
        "text": "Pre Boarding Checklist",
        "textStyle": "font-weight:bold;margin-bottom:0.5rem",
        "icon": "kix-icon-ci",
        "componentId": "dynamic-field-value",
        "componentData": {
          "name": "PreBoardingChecklist"
        }
      },
      {
        "text": "Optionale 2. Checkliste",
        "textStyle": "font-weight:bold;margin-bottom:0.5rem",
        "icon": "kix-icon-ci",
        "componentId": "dynamic-field-value",
        "componentData": {
          "name": "NameDerCheckliste"
        }
      }
    ]
  }
}

```

4. KIX Start: In the editor, minimise the source code to remove unnecessary spaces and line breaks and copy the source code to the clipboard.
5. KIX Start: Paste the source code from the clipboard back into the "Value" field.
6. KIX Start + KIX Pro: Apply the changes by clicking on "Save".
7. KIX Start + KIX Pro: Click on "Load Frontend Configuration" to update the display in the ticket zoom view.



Name	Modified	Context	Metadata	Access Le...	Validity	Value	Changed at	Changed by
ticket-new-form-group-data		kix18-web-frontend	FormGroup	Internal	valid	["id":"ticket-new-form-group-data","name":"Tra...	09/06/2022, 10:28 AM	not assigned

Display in the Home Dashboard

In KIX Start and KIX Pro, agents can customise the tables in their home dashboard themselves and also show or hide the dynamic fields created in the system as additional column(s). In this way, each agent can decide for himself whether and in which table he would like to have the checklist status displayed. This does not change the basic configuration of the table(s), only the user-defined view in the Home Dashboard.

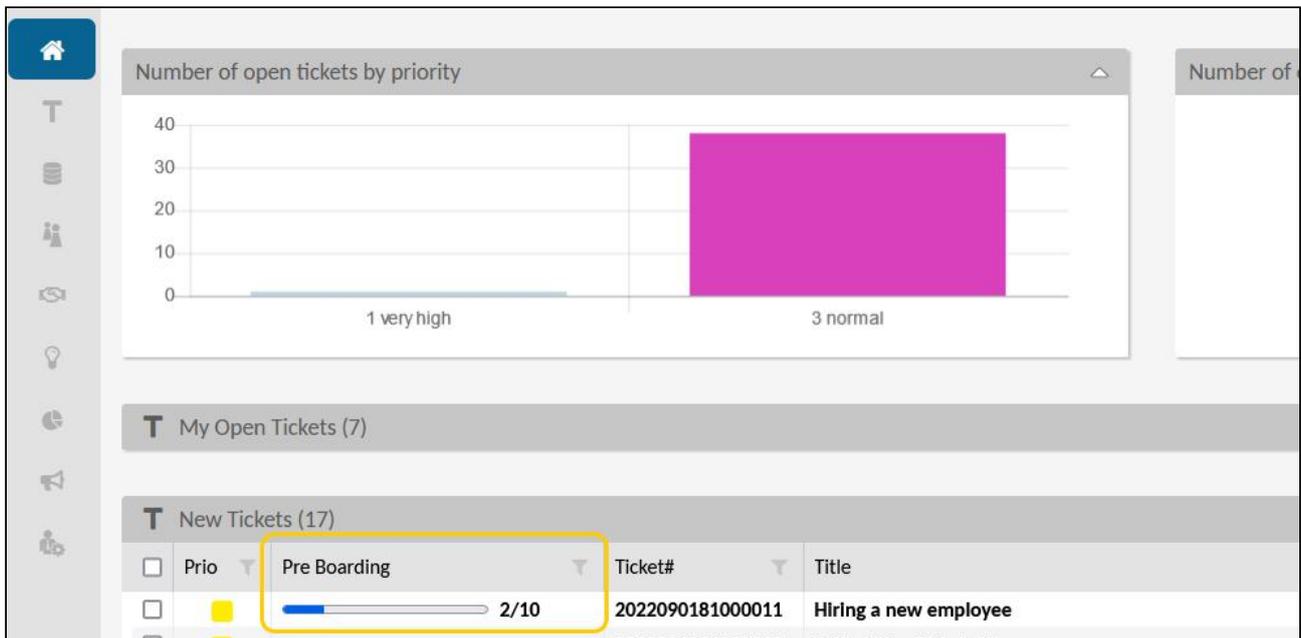


Fig.: The checklist status in the Home Dashboard

To customise, click Customise Dashboard at the bottom of the Home Dashboard screen.

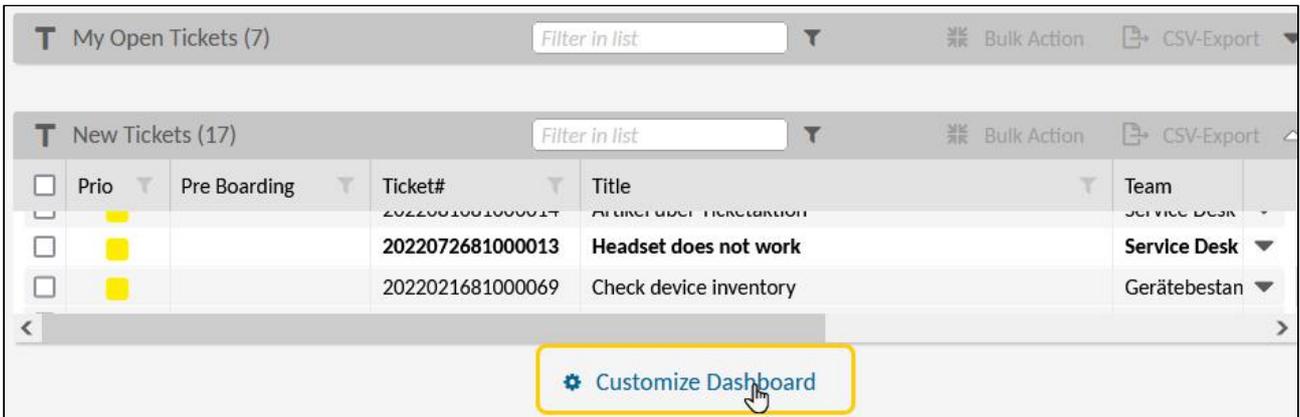


Fig.: "Customize Dashboard" button

In the view that opens, the checklist and thus the progress bar can be integrated into the desired table by adding a table column. The column can be moved into the desired order by dragging and dropping.

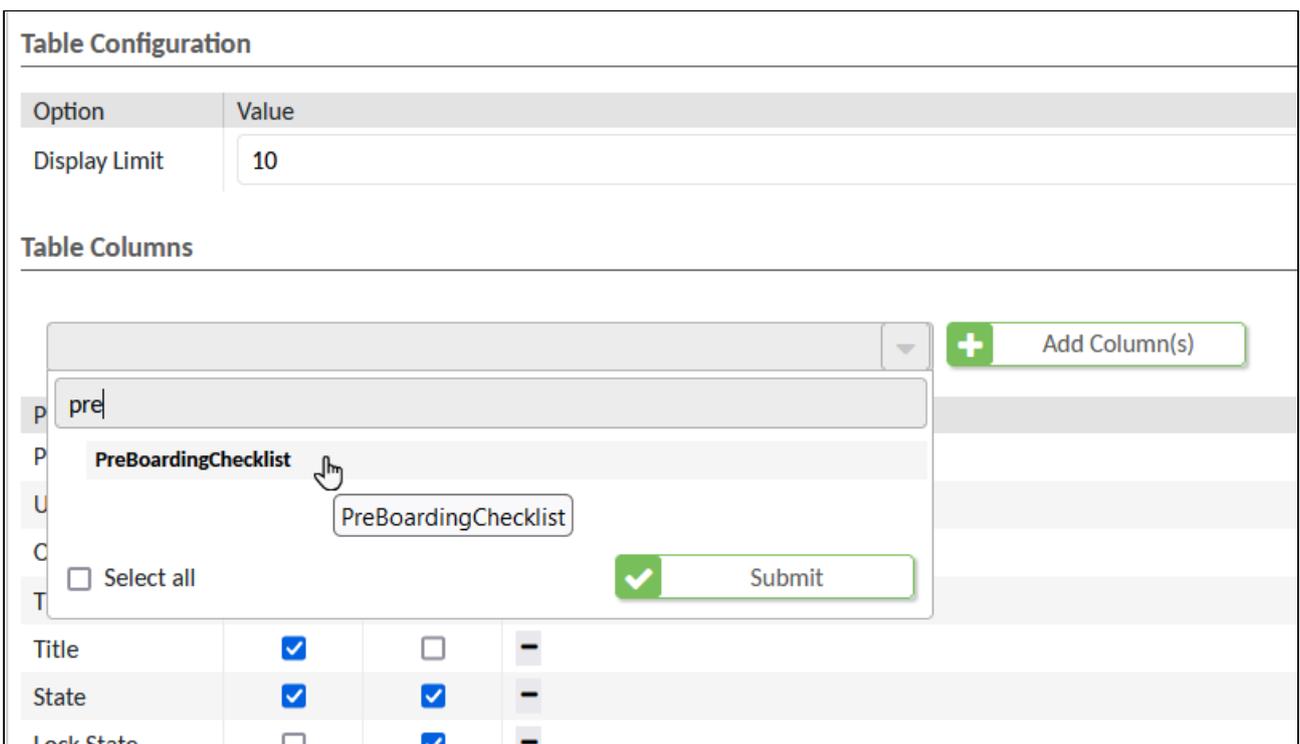


Fig.: Integration of the checklist status in the configuration of the Home Dashboard

So that not only the numerical value of the checklist status (2/10) is displayed in the table, but also the progress bar, the component ID "dynamic-field-checklist-cell" must still be specified in the extended view. To do this, click on "Advanced":



Fig.: "Advanced" button in the personalisation of the Home Dashboard

Property	Text	Icon	Column ...	Column ...	Sortable	Filterable	Filter List	Resizable	Translata...	Translata...	Size (px)	Component ID	Column Title
Priority	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	65		
Pre Boarding	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	225	dynamic-field-checklist-cell	
Ticket#	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	135		
Title	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	162		

Fig.: Completion of the component ID

Further information can also be found in the KIX Start user manual: Home Dashboard > Personalise Home Dashboard

Alternative

Configuration key	home-dashboard-ticket-table-new
--------------------------	---------------------------------

As an alternative to personalising the home dashboard, the checklist progress can also be integrated into the "new tickets" table by configuring the SysConfig key "home-dashboard-ticket-table-new". For this purpose, another column is added to the table, which contains the progress bar. This global change to the SysConfig key changes the basic configuration of the table in the home dashboard for all agents.

Procedure

1. KIX Start: Navigate to *System > SysConfig*.
KIX Pro: Navigate to *System > GUI Configuration > Agent Portal*.
2. KIX Start + KIX Pro: Open the key "home-dashboard-ticket-table-new". This key defines the structure of the table.
3. KIX Start: Open the key and copy the source code from the field "value" into a JSON editor. The right column contains the source code in maximised view for easier editing.
4. KIX Start + KIX Pro: Paste the following code block (without comments!) into the editor.
 1. Scroll to the section `tableColumns [. . .]`. The table columns are defined in this section.
 2. Place the code block at the desired position between two existing table columns. A table column begins and ends with a curly bracket. Two consecutive table columns are separated

by commas. The placement in the key determines at which position the column is inserted in the table.

```
{
  "id": null,
  "name": null,
  "type": null,
  "property": "DynamicFields.PreBoardingChecklist",
  "showText": true,
  "showIcon": false,
  "showColumnTitle": true,
  "showColumnIcon": true,
  "size": 225,
  "sortable": true,
  "filterable": true,
  "hasListFilter": false,
  "dataType": "STRING",
  "resizable": true,
  "componentId": "dynamic-field-checklist-cell",
  "defaultText": null,
  "translatable": true,
  "titleTranslatable": true,
  "useObjectServiceForFilter": false
},
```

```

{
  "id": "home-dashboard-ticket-table-new",
  "name": "Translatable#New Tickets Table",
  "type": "Table",
  "objectType": "Ticket",
  "loadingOptions": {},
  "displayLimit": 10,
  "tableColumns": [
    {},
    {
      "id": null,
      "name": null,
      "type": null,
      "property": "DynamicFields.PreBoardingChecklist",
      "showText": true,
      "showIcon": false,
      "showColumnName": true,
      "showColumnIcon": true,
      "size": 200,
      "sortable": true,
      "filterable": true,
      "hasListFilter": false,
      "dataType": "STRING",
      "resizable": true,
      "componentId": "dynamic-field-checklist-cell",
      "defaultText": null,
      "translatable": true,
      "titleTranslatable": true,
      "useObjectServiceForFilter": false
    }
  ]
}

```

5. KIX Start: In the editor, minimise the source code to remove unnecessary spaces and line breaks and copy the source code to the clipboard.
6. KIX Start: Paste the source code from the clipboard back into the SysConfig key
7. KIX Start + KIX Pro: Apply the changes with "Save".
8. Click on "Load Frontend Configuration" to update the table in the Home Dashboard.
The agents must also reload your frontend.
Afterwards, the column in the dashboard table "new tickets" is inserted and the progress bar is displayed.

11.2.5.4 References:

- Dynamic fields:



- Dynamic Fields
- Object and field types of dynamic fields
- Integrating a Dynamic Field
- Displaying Values of Dynamic Fields
- Ticket templates and actions in KIX Pro:
 - [Templates](#) (see page 191)
 - [Actions](#) (see page 137)
 - [Create and configure actions](#) (see page 145)
- JSON-Editor
 - <http://jsonformatter.io>
 - KIX Pro: JSON-Editor (see page 59)
- Anpassung Home Dashboard: Home Dashboard > Personalize your home dashboard

11.2.6 Data structure of checklists

Basically, Dynamic Fields of type "Checklist" consist of JSON content describing the complete content of the checklist. This results in extended possibilities for the use and configuration of checklists.

The data structure corresponds to the following structure (based on [EBNF¹⁰](#)):

Structure of the checklist values

```

CheckList := [ <CheckListItem>+ ];

CheckListItem := {
  "id": "intvalue",
  "title": "Some string describing the task",
  "input": "TextArea | Text | ChecklistState"
  "value": "- | pending | OK | NOK | n.a. | ... or just a string for Text-inputs...",
  "description": "Optional description for the task.", ? # optional
  "sub": <CheckList> ? # optional
}

```

This means that when actions or templates are created, a checklist can be completely reset by assigning a JSON string. An adjustment of the dynamic field is not necessary for this. This can also be done automatically by a job (configuration in the menu: *Automation > Jobs*).

In conjunction with the variable filter "jq", existing checklists can be modified, e.g. to automatically add, check off or remove checklist tasks (see also <https://jqplay.org>).

Example of a checklist

```

1  [
2    {
3
4      "id": "100",
5      "title": "Announce shut off",
6      "description": "Announce shut off to all possibly affected personell
7      directly or indirectly working with the device.",
8      "input": "ChecklistState",
9      "value": "-"
10   },
11  {
12     "id": "200",
13     "title": "Identify energy source(s)",
14     "description": "Check for connected external and internal energy
15     sources.",
16     "input": "ChecklistState",

```

¹⁰ https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form



```
15     "value": "-"
16   },
17   {
18     "id": "300",
19     "title": "Isolate energy source(s)",
20     "description": "Document measures you took in order to isolate energy
21 source(s).",
22     "input": "ChecklistState",
23     "value": "-",
24     "sub": [
25       {
26         "id": "310",
27         "title": "Isolation by",
28         "input": "Text",
29         "value": ""
30       }
31     ],
32   },
33   {
34     "id": "400",
35     "title": "Lock & Tag energy source(s)",
36     "description": "Lock energy sources to avoid accidental re-energizing
37 while working on the device. Tag the device as out of order due to
38 maintenance actions.",
39     "input": "ChecklistState",
40     "value": "-",
41     "sub": [
42       {
43         "id": "410",
44         "title": "Information by",
45         "input": "TextArea",
46         "value": ""
47       }
48     ],
49   },
50   {
51     "id": "500",
52     "title": "Ensure that equipment isolation is effective",
53     "description": "Before starting maintenance or repair tasks ensure
54 that isolation is working.",
55     "input": "ChecklistState",
56     "value": ""
57   }
58 ]
```

12 Liability Disclaimer KIX Pro

12.1 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.

12.2 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.



12.3 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.

Copyright ©2024

KIX Service Software GmbH, Chemnitz

Manufactured on behalf of the KIX Service Software GmbH, Chemnitz

Program development: KIX Service Software GmbH, Schönherrstrasse 8, 09113 Chemnitz

Documentation: bluescript, Zeppelinstraße 9, 08451 Crimmitschau

13 Purpose for which the use of KIX Pro is intended within a medical context

KIX Pro is not suitable, intended or approved for the identification, prevention, monitoring, treatment, relief or compensation of illnesses, injuries and disabilities. KIX Pro is also not intended for the examination, replacement or modification of the anatomical structure or of a physiological process. KIX Pro must also not be used to directly control a diagnostic or therapeutic product. Furthermore, KIX Pro 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 Pro is exclusively intended for carrying out administration and database-related tasks. If the above restrictions are complied with, KIX Pro 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 Pro 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 Pro also allows data exchange via an interface, please note that KIX Pro must not be used for data modification or for any type of data control for medical or therapeutic purposes.

KIX Pro 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.