
# Architecture Committee Handbook

openKONSEQUENZ

created by

Architecture Committee

We acknowledge that this document uses material from the arc 42 architecture template, http://www.arc42.de. Created by Dr. Peter Hruschka & Dr. Gernot Starke.
*Template Revision: 6.1 EN*
*June 2012*

## Revision History

| Version | Date | Reviser | Description | Status |
|---|---|---|---|---|
| 1.0 | 2016-07-04 | A. Göring | Alignment in AC/QC conference call | Released |
| 1.0.1 | 2016-07-19 | A. Göring | Added UML-Tool decision in chapter 2. Constraints, Added software-tiers image in chapter 8. | Draft for v1.1 |
| 1.1 | 2016-08-18 | A.Göring | Alignment in AC/QC conference call | Released |

## Formal

According to a decision of the Quality Committee, this document is written in english.

Document control:

Author: Andre Göring, andre.goering@offis.de (assistant of architecture committee)

Reviewed by: SC, PPC, and QC of openKONSEQUENZ

Released by: AC



Released versions will be made available via the openKONSEQUENZ web site.

## Open Issues for Architecture Committee Handbook

This is a living document. Further general architectural topics have to be detailed by the Architecture Committee and can not yet be answered without further knowledge from further openKONSEQUENZ projects. Known issues are listed red coloured.

## Module specific Architecture Documentation Hints

GREEN & Boxed: Open architecture documentation issues for module developers in **module specific** arc42-document or tender or global oK-CIM-Profile.

## Related documents

| Document | Description |
|---|---|
| BDEW Whitepaper | Whitepaper on requirements for secure control and telecommunication systems by the german BDEW Bundesverband der Energie und Wasserwirtschaft e.V. (https://www.bdew.de/internet.nsf/id/232E01B4E0C52139C1257A5D00429968/$file/OE-BDEW-Whitepaper_Secure_Systems%20V1.1%202015.pdf) |
| BSI TR-02102 | Technical Guideline according to encryption recommendations and key length by the german BSI - Bundesamt für Sicherheit in der Informationstechnik (https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html) |
| oK-Charter | The openKONSEQUENZ charter (https://wiki.eclipse.org/images/f/f5/20150623a_openKonsequenz_V14-3_%283%29.pdf) |
| oK-GUI-Styleguide | Style guide for module developers of openKONSEQUENZ modules according to the graphical user interface. (https://wiki.eclipse.org/OpenKONSEQUENZACQCRichtlinien) |
| oK-Interface-Overview | AC handbook external but related AC document (appendix), where the Interfaces and the overall oK-CIM-Profile of oK-Modules are described in short as well as showing the Building Block View Level 1. (https://wiki.eclipse.org/OpenKONSEQUENZACQCRichtlinien) |
| oK-Module-Tender-Call | The openKONSEQUENZ project planning committee prepares a document which describes the requirements to the development for each module. With this document it calls for tenders at software developers (module individual) |
| oK-Module-Tender | The software developers answer to the oK-Module-Tender-Call (module & developer individual) |
| oK-Vision | The oK document “Vision/Mission/Roadmap” - it is currently not available online. |
| oK-Website | The website of openKONSEQUENZ (www.openkonsequenz.de) |

| | |
|---|---|
| Quality Committee Handbook | Textural guideline for module developers of openKONSEQUENZ modules according to quality related issues. (https://wiki.eclipse.org/OpenKONSEQUENZACQCRichtlinien) |

# Table of Contents

# 1. Introduction and Goals

The consortium OpenKONSEQUENZ (oK) consists of Distribution System Operators (DSO), their software manufacturers, integration service providers and academic institutes. It targets to overcome vendor lock-in in the hierarchical grown IT-infrastructure of DSOs. The openKONSEQUENZ® Working group (openKONSEQUENZ® WG) wants to foster and support an open and innovative eco-system providing tools and systems, qualification kits and adapters for standardized and vendor independent e-Systems. Therefore openKONSEQUENZ defines a reference platform, where different modules for different purposes shall be implemented for and interact with existing parts of the IT landscape and other modules under development.

This handbook describes the architectural view on openKONSEQUENZ software, its current overall system and software architecture which developers shall develop in respect to and what software architecture artifacts (documentation & graphs) developers have to document during development. In especially the “green colored TODOs” mark important documentation that module developers must create.

These artifacts have to be stored in Arc42 - architecture documentation for each module in the module specific git-repository (see Quality Committee (QC) handbook for naming and directory conventions). The architecture guidelines are generally applicable. These guidelines can be adapted over time based on specific need and/or request from Architecture Committee (AC) members or third party application developers. A change management process will be set up for this in the future.

## 1.1 Requirements Overview

The openKONSEQUENZ is an umbrella for a rising number of projects with common intends:

### Functional Requirements:

- Provide a common interoperable platform, which can unite the data of different existing systems and offers a streamlined environment for extending these existing systems by new developed modules. The platform uses open interfaces and reduces or optimizes interfaces where applicable.
  - What happens: different systems share data along the platform.
  - Why:
    - Islands in IT landscape shall be connected in an easy way
    - To overcome vendor dependent interface development and vendor lock-in.
- Get software support for new and demanding requirements in context of the energy policy turnaround.
  - What happens: New modules realize functions on existing information
  - Why: New functions required to easily solve problems / processes.

### Non-functional Requirements:

- Requirements for modules according to confidentiality, availability and integrity are defined in levels normal, high and very high (normal is descripted in BDEW Whitepaper). The oK reference platform has to be designed for meeting specification for levels high to very high.
- Detailed and hard requirements for the oK reference platform shall be listed in Section 10 Quality Scenarios.

Different modules may not only underlie different functional requirements, but also different non-functional requirements. These are specificated in the tenders for each module. So the entirety of requirements for modules comes from tender documents, the AC and QC handbooks, SCRUM product backlog and SCRUM sprint backlog.

## 1.2 Quality Goals

Overall Quality Goals of openKONSEQUENZ are detailed in the openKONSEQUENZ Charter and listed in short:

- process and data integrity with standardized interfaces,
- long term maintainability for components to be usable longer than 15 years,
- compliance with frequently changing regulation,
- vendor-neutrality,
- availability as needed,
- security for critical infrastructure by design,
- innovations in products and development.

There is a need to make a difference between openKONSEQUENZ user modules, service modules and a reference platform itself to qualify goals. The platform is a standardized host for new modules. Service modules are modules that are to be used in several projects for supply of data or services. User modules are that applications, a user from a DSO uses for solving their use case. For each part, the Quality Goals have to be discussed individually. Please check also Quality Committee Handbook for quality related requirements.

### Reference Platform and Service Modules

- Flexibility - The platform and its service modules shall grant, that different systems and modules from different vendors/developers can interact and interoperate, and may be exchanged or recombined.
- Availability - All service modules that are running on the platform can only be as available as the platform - same for user modules that are based on service modules.
- Maintainability (and testability as part of maintainability) - platform and its service modules shall be used longer than 15 years.
- Integration performance - new implemented functionality of oK own modules and external modules shall be included fast / automatically.

- Security - the platform and its service modules need to underly security-by-design

### User Modules:

- Functionality - user modules must fulfil their functional requirements
- Integration performance - user modules must be easy integratable in different production environments.
- Modifiability (and testability as part of modifiability) - Good documentation (i.e. code and architecture documentation) makes code changes easier and automatic tests facilitate rigorous verification.
- Ergonomics - according to oK-GUI-Styleguide.

## 1.3 Stakeholders

### DSOs

Need software fulfilling their functional and non-functional requirements.

### AC

Manages openKONSEQUENZ Architecture demands and is responsible for this document.

### QC

Manages openKONSEQUENZ Quality demands and is responsible for the related Quality Committee Handbook.

### Module Developer

Is the software developer who develops a module (or the tender for the module). He has to take the QC handbook and this AC handbook into account, when offering a tender and when developing a module.

### System-Integrator

Needs information for integration of modules in a specifics DSOs environment.

TODO for architecture documentation
by module developer according to chapter 1:

- Document defined module specific functional and non-functional Requirements/Quality Goals/Acceptance Criteria in the module project's git-repository in an arc42-document.

# 2. Architecture Constraints

In the following section, constraints are listed. This are any requirement that limit software architects in their freedom of design decisions or the development process:

- License: Open Source-Licence “Eclipse Public License 1.0” with its demands on Intellectual Property (IP) for the usage of libraries.
- Commissioning/Hiring/Business: In the consortium, one or more of the DSOs is/are the driving force for a module. In procurement the Lead Buyer (N-ERGIE AG) is contactpoint for business questions. In contrast to this the driving DSO is handling functional/technical questions and responsible for commissioning.
- Development using SCRUM.
  - Product Owner comes from the specific module driving oK DSO
  - SCRUM Master from the module developing company.
- Quality Control and Acceptance: Sprints + 3 month test operation + See QC handbook.
- Standardization: Usage of standardized data structures (CIM) and the reference platform.

## 2.1 Technical Constraints

Technical frameworks or requirements constraining the developing of modules are largely based on the experience made with the initial pilot application implementation. These are set to allow interoperability between the various modules to come and the oK reference platform and for oK reference platform enhancements.

When an application developer would like to use different components than those listed below, he has to get approval from the Architecture Committee.

| Software platform framework and requirements | |
|---|---|
| Basis components of the reference platform | Portal Liferay;<br>Application Server Tomcat;<br>JPA EclipseLink;<br>Database PostgreSQL;<br>ESB Talend (open studio for ESB)<br>BPMN Engine Camunda |
| Runtime engine | JavaEE 7 |

| | |
|---|---|
| ESB-Interfaces | openKONSEQUENZ-CIM-Profiles for APIs based on CIM 17 or later via CIM RDF/XML for topologies or deep nested structures and XML for others via RESTful Webservices. |
| GUI | See oK-GUI-Styleguide |
| Programming Language GUI | AngularJS, Bootstrap, jQuery, REST/JSON Interfaces |
| Libraries, Frameworks, Components | Used Libraries/Frameworks have to be compatible to be used in an Eclipse Public License project.<br>Maven,<br>For Libraries for quality check and Continuous Build/Deployment/Integration see QC Handbook |
| Programming Constraints | |
| | See QC Handbook |
| UML-Tooling | For CIM related modeling, the use of the tool “Enterprise Architect” (EA) is strongly encouraged. If other tools are applied, a data transfer (export/import) to EA must be frictionless, i.e. model structure, contents, and diagrams have to be transferred to a EA model (possible). For software architecture related modeling, an open source UML tool shall be used. Up until further notice, this tool is “Modelio”. |

This framework and set of products describes the openKonsequenz reference system components. Instantiations on grid operator site may get adapted on request by the grid operator or on recommendation of the system integrator to cope with specific grid operator requirements regarding i.e. integration with existing ESB, BPM or application server technology.

## 2.2 Organizational Constraints

For architecture constraints responsible persons/groups are listed below.

| Organization and Structure |
|---|

| | |
|---|---|
| Steering Committee (SC) | Has the final say in every oK question. |
| Project Planning Committee (PPC) | Plans following projects and specifies requirements for each individual module. Calls for tenders based on their requirements, the AC handbook, QC handbook and oK-GUI-Styleguide. |
| Architecture Committee (AC) | Gives Framework and Constraints for architecture according to technology, interfaces, reference environments, architectural concepts, design decision and documentation. Every difference from their architecture guidelines in module development has to be approved and agreed by the AC. The AC may in case of deviation advise the Product Owner to refuse or accept a sprint acceptance or product acceptance in architectural questions. |
| Quality Committee (QC) | Gives Framework and Constraints for quality of the software, like code quality & styleguides, testing, build. These quality aspects also mirror back to architecture questions (and vice versa). |
| Product Owner (PO) | Gives and prioritizes requirements and acceptance criteria in Product Backlog for Sprint Planning / Backlog and checks their fulfilment in sprint reviews and may by this limit architectures solution space of a module. The PO may refuse a sprint or product acceptance cause of non-fulfilment of architecture requirements. |
| SCRUM Master (SM) | Is responsible for the success of the SCRUM. SM checks for Compliance with the SCRUM Process and ensures as well the communication between module developers and PO. SM helps PO at maintaining the Product Backlog and the developer at the definition of done. |

The modules are developed in Eclipse Projects. User modules may be developed in closed projects, where only the contractor for module development is a Committer (According to the committer rules of Eclipse Project). Service modules shall be developed (further) in oK platform project, where oK-Core-Developer (that need to be established) are Committer.

## 2.3 Conventions

- Architecture related module specific documentation has to be documented in the module specific git repository (see QC Handbook for directory & type advises) according to the Arc42 template. Documentation language is english. In especially the “green colored TODOs” mark important documents/arc42-parts, that have to be created by module developers. Use UML for graphs! (For Tooling see chapter 2.1).
- See “QC Handbook” (for coding styleguides, quality reports, naming conventions, version and configuration management...)

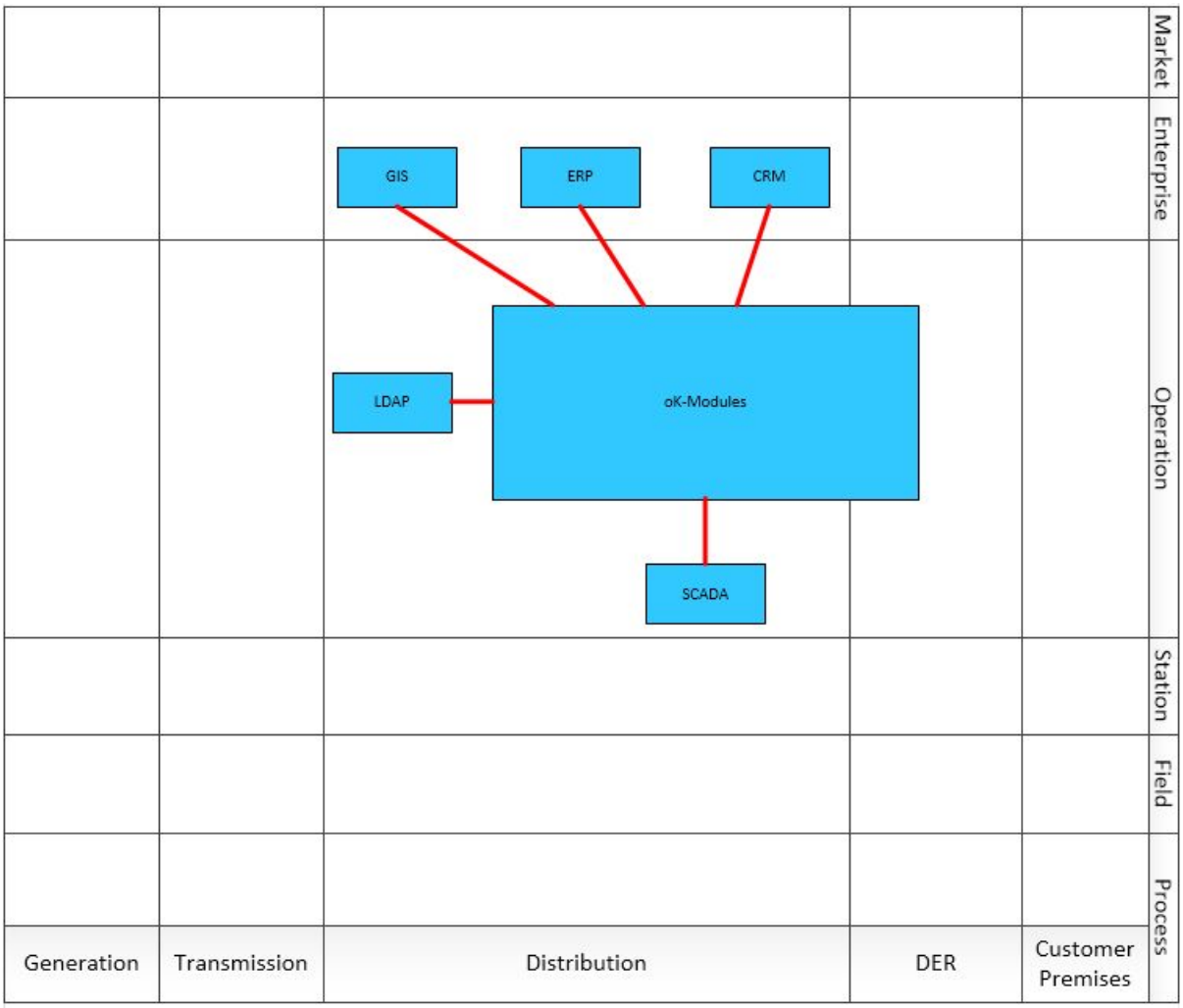
# 3. System Scope and Context

The openKONSEQUENZ develops modules for the operation of distribution grids according to current IT technologies and regulation. These modules extend the functionality of SCADA, often coupled with other IT of DSOs. The openKONSEQUENZ platform is a host for user modules and provides services with service modules.The user modules offer extended functionality, that the existing IT-Infrastructure can not provide (either in functionality or in quality). The service modules provide common services, that are needed for a variety of user modules, external modules or even the current existing IT systems. Therefore the modules communicate with each other and with the existing IT infrastructure via interfaces to gather and distribute the necessary data for operation.

- Main focus is the communication between different modules on business logic tier to ensure communication between different systems in different settings from different DSOs.
- As transport medium, the Enterprise Service Bus is required to handle all communication between user modules and service modules as well as to existing IT-Infrastructure.
- No direct communication between modules and especially no communication via direct access to the same database schemes shall exist.
- Also the tier-wise communication between graphical user interface and business logic is in focus of openKONSEQUENZ.

## 3.1 Business Context

In openKONSEQUENZ additional modules for distribution network operation are developed, that are usually located outside of a SCADA system. A longterm view is shown in the following picture according to the Smart Grid Architecture Model plane. A use of openKONSEQUENZ modules is also imaginable in the TSO domain, the components shown in the figure then move one column to the left. Module developers have to have in mind, that DSOs often separate the operation zone into common operation and SCADA.




Neighbouring systems of modules are

- existing DSO systems
  - SCADA,
  - GIS,
  - ERP,
  - CRM,
  - Weather forecasts
  - ...
- Other service modules
  - CIM Cache
  - Archive (in development)
  - Identity & Access Management (in development)
  - ...
- Other user modules
  - Eisman
  - Operation Diary (in development)
  - Switch Request Management (in development)
  - ...

## 3.2 Technical Context

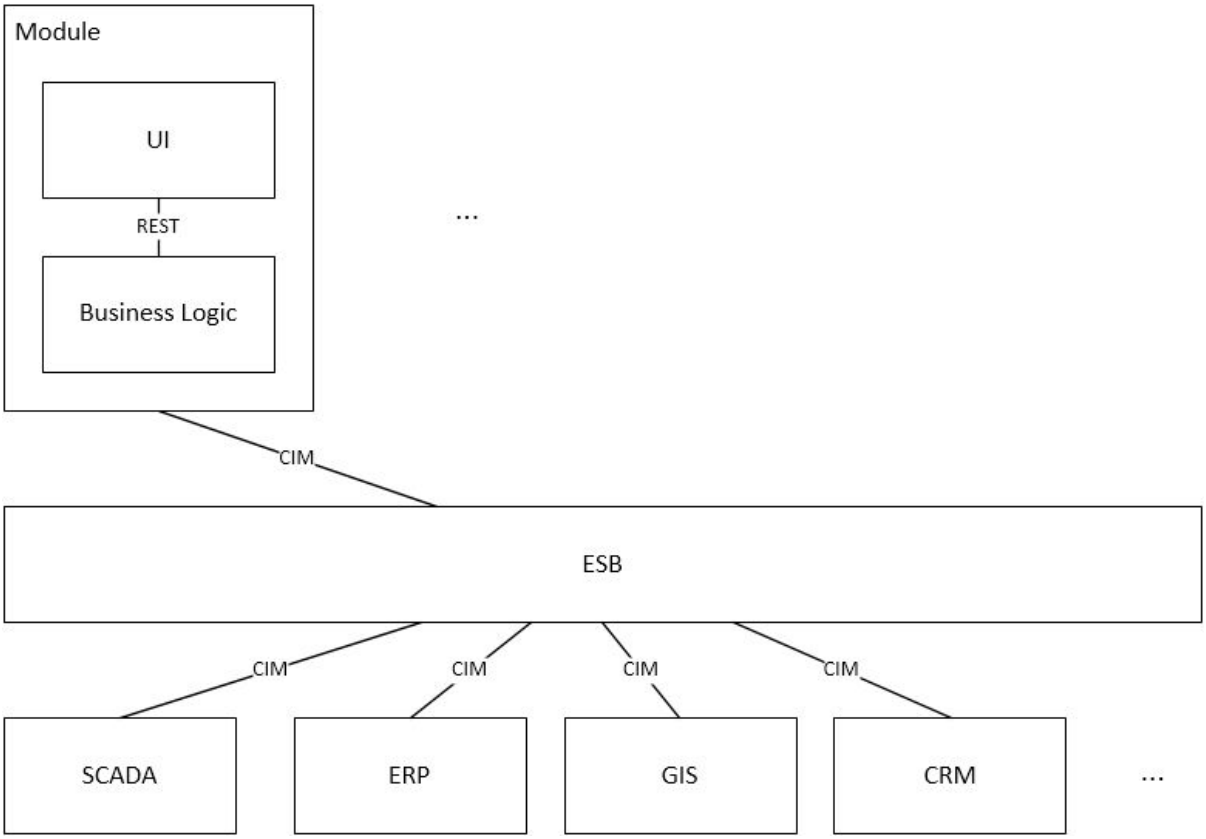
Common system landscape of DSOs is characterized by heterogeneous data models in closed legacy systems with proprietary interfaces. Because of this, there are high integration barriers and -costs that slow down grid operators and software supplies (redundant data management, inconsistencies, adapter dev.,...). The oK approach is to derive open interfaces/data models based on CIM standard with standardized semantics and less misunderstandings and proprietary knowledge.

According to the following figure, there is to differ between inner module communication between the business layer and the UI layer (called REST in the figure) of a module and module external communication or inter-module communication (called CIM in figure).
For the module external communication (called CIM in figure) interfaces to all systems of DSOs have to be taken into account under following aspects:

- As a basis, the Common Information Model - CIM (in current development Version 17 or newer) is/has to be used for definition of interfaces. It provides an ontology for equipment in the electrical domain, giving semantics and syntax of attributes, associations and classes in an object oriented way.
- This includes the use of serialization format RDF (CIM/XML) for topology or deep nested structures and XML for all others as well as the use of CIM-envelopes (See IEC 61968-100).
- If a module is not using this serialization format, it has to be reasonably explained and documented and needs an authorization of the ACQC.
- As interface-technology RESTful web services are used.
- For each external interface (interfaces between modules or external systems) the interface has to be documented in this document.
- Interfaces have to be designed under the interface segregation principle.
- For checking out interoperability of interfaces, the existing overall openKONSEQUENZ interface schema must be extended by the module developer - this shall ensure reutilization of existing parts of the schema and avoid inconsistencies in semantics/syntax.
- Dependencies of modules to services realized by other modules have to be specified and documented explicitly.
- When CIM is not appropriate (like access management), other standards in their respective domain shall be taken into account first to avoid proprietary and inaccurate interfaces. The interface has also be documented in the overall openKONSEQUENZ interface profile and it should use REST & XML - otherwise reasonably explained and documented exceptions have to be approved by ACQC.
- In longterm a development of shared DSO CIM profiles comparable to entso-e profiles is planned: The particular models of a certain grid relating messages or files

are conform to the overall oK-DSO-CIM-profile specification, which itself shall be conform to the official (future) CIM standard.

For inner module communication (called REST in the figure) RESTful Webservices shall be used with json serialization, which fits well for the JS-GUI Tier. This is common to user modules and service modules. If the latter does not need an UI, the REST called interface (in the figure) and UI can be left away.




## 3.3 External Interfaces

### Current overall oK interface profile

The overall oK interface profile is hosted in the git-Repository of openK-platform (openk-platform/openk-cim-v17) as Java packages and classes with Java documentation for the semantical use of attributes/classes and as Image of an UML class diagram. If extensions are needed for modules, the module developer has to update this overall model with needed attributes/classes and their semantical meanings. Thereby it shall be granted, that module developer take the current available model as basis, that no redundant information is in the profile and that there is no semantical inconsistency introduced. An Overview on the overall oK interface profile can be found in the oK-Interface-Overview annex.

### Short interface description for each oK-ESB-interface

For each ok-ESB-interface, a module provides/requires, a detailed description of the interface can be found in the module-specific Arc42-documentation. An overview, which

introduces all ok-ESB-interfaces and lists provider and requirer, can be found in the oK-Interface-Overview annex.

TODO for tender
by module developer according to chapter 3:

- Specify provides- and requires-interfaces for the planned module as a first draft (if there are no yet fitting oK-interfaces).

TODO for architecture documentation
by module developer according to chapter 3:

- Define module specific scope and context in the module own architecture documentation (in projects git repository)
- Get overall oK interface profile from ok-platform git repository, update it, if current version of oK interface profile lacks classes/attributes and commit changes to ok-platform git repository. Eventually update the oK-Interface-Overview appendix.
- Describe own external interfaces (requires and provides) in architecture documentation, create their UML models and XML Schema Files (in respective git-repository-directory (see QC-Handbook for directory details). Update the oK-Interface-Overview appendix.

# 4. Solution Strategy

In the tenders for a module, the module developer has to provide an overall solution architecture and to define which oK-modules are planned to be reused by the module. Also, the module developer has to specify changes to existing modules, if required.
In the tender, the developer lists the libraries that will be used (an IP-check on each new library is required and shall be the responsibility of the module developer). If libraries with the necessary capabilities are already listed as “cleared”, they shall be used as a default, any project who wants to override the default list needs to present rationale and needs the approval of the architecture committee.
IP-checks can be long-running tasks. It is strongly recommended to perform an IP-check before creating the tender. Any planning, estimation, design or implementation based on a library that does not pass the IP-check is wasted.

## Platform

Existing systems, (possible) external developed modules, user modules and service modules interact on basis of standardized interfaces and run on a reference architecture concept underlying system. A Portal and a UI-Styleguide makes it easy for Users to work with new modules.

## CIM-Cache (service module)

The CIM-Cache is a service module. It caches the needed CIM related information from external systems, that shall not steadily be requested directly by oK-Modules. The external systems shall update changes to the CIM-Cache.

TODO for module developer in tender:

- provide overall solution architecture, list of reused modules, changes to reused modules, list libraries that will be used.

TODO for module developer according to Arc42-documentation:

- Describe solution strategy for each module in module own architecture documentation

# 5. Building Block View

## 5.1 Level 1

The openKONSEQUENZ building blocks are the user modules and the service modules. They're connected to each other by provides and requires interfaces. New modules can be build using existing modules interfaces and information (via the interfaces).

TODO for architecture documentation
by module developer according Building Block View Level 1:

- Document, which modules are reused with which interfaces as UML component diagram in oK-Interface-Overview appendix.

## 5.2 Level 2

TODO for architecture documentation
by module developers according to Building Block View Level 2:

Document Level 2: Internal components of a module and a logical view have to be documented as component diagram and class diagram with UML:

- Overview of components and internal interfaces
- Detail specification of "complex'" (according to QC-Handbooks top 2 categories) modules as UML component and class diagram
- Detailed documentation as JavaDoc
  - Separation between architecture and quality is needed in order of the use and especially the maintenance of modules
- Database model as entity relationship or UML class diagram
- List existing and new Access rights, that are required by module (and how to create them).
- Document security relevant system components and implementation specification (according to BDEW Whitepaper chapter 2.1.2.1, if it has to be applied).

# 6. Runtime View

Todo for architecture documentation
by module developer according to chapter 6:

- Show Interaction of building blocks (components) at runtime for exemplary processes (use cases / scenarios) with bpmn diagrams (for functional processes) and sequence/collaboration UML diagrams for technical processes.
- Non-trivial processes in the business logic and processes, in which several modules are involved, have to be modeled as UML sequence or collaboration diagram.
  - Several modules scenarios:
    - Technical documentation of the processes as sequence diagram, which shows calls between user, modules and external systems.
    - Matching of interfaces of the modules according the functions- or service calls for quality assurance
  - Module intern scenarios:
    - UML sequence diagram (or collaboration diagram) define functions-, communications- and data-flows for a scenario

# 7. Deployment View

The module developer must develop system interface mockups (https://en.wikipedia.org/wiki/Mock_object) for the external systems or legacy systems that the developed module needs data from. These mockups shall be integrated in deployment stages as early as possible, but must be available and integrated in Quality Assurance Environment for Sprint Reviews. If there is a demand in the tender also for integration of real systems, one of the DSOs (specified in tender) has to provide appropriate test systems for integration.

There are several deployment environments required for the different participants and roles. These deployment environments are based on a reference environment where the reference platform software is preinstalled. Code check, testing and build/integration/deployment tools (etc.) according to QC Handbook will be made available on the reference system and therefore be part of the deployment template accessible for the developers.

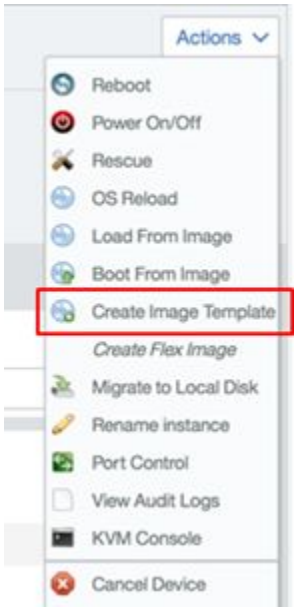
The different deployment environments are listed in following table and then detailed:

| Environment Name | Short description | Logical order of deployment | Refernce Environment usage (Y = yes - mandatory, N = no, P = possible) | Responsibility | Module developers task |
|---|---|---|---|---|---|
| Reference Environment | The oK standard environment used as reference and the oK overall test environment | 4 | Y | oK | Support deploying their modules |
| Develop Environment | Environment a developer is using for debug & test purposes | 1 | P | Developer | Up to the developer |
| Integration Environment | Environment for checks of | 2 | P (recommend | Developer | Up to the developer |

| | | | | | |
|---|---|---|---|---|---|
| | integration of branches | | ed) | | |
| QA Environment | Environment for Sprint Reviews | 3 | Y | oK | Deploy module & legacy system mocks; Product Owner acceptance |
| Demo Environment | Environment for demonstration of oK to “the world” | 5 | Y | oK | Integrator or Developer (descision of SC/PPC) |
| Customer Environment | DSO Environments for integration in DSO system landscape, test, and production | 6 | P (recommended) | DSO | - |

- Reference Environment - The reference environment is the reference instance for all other deployment environments and is operated out of an IBM Softlayer Cloud in Frankfurt am Main.
    a. The reference system has the latest accepted version of the oK reference platform and the modules.
    b. Updates to the reference system can be made after successful test and acceptance in a staging environment that is different from the reference environment.
    c. The reference system is in terms of cloud environment a ‘permanent’ system -- means always accessible.
    d. The creation of a template from the reference system and storage of the template in the library allows for easy instantiation for the various other deployment environments -- test, dev, acceptance, ... The creation of a template can be done easily through the cloud management portal

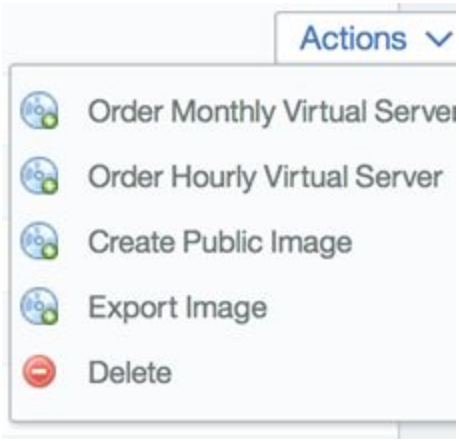
environment:




e. The current configuration of the reference system is based on the experience from the pilot and will be based on 1VM (Virtual Machine) with 8GB RAM, 100GB Disk, CentOS incl. Public and private IP-address and VPN access. Changes to the infrastructure can be easily done through the cloud environment (more memory, more disk) within minutes depending on growth requirements.

f. This system is owned by openKonsequenz

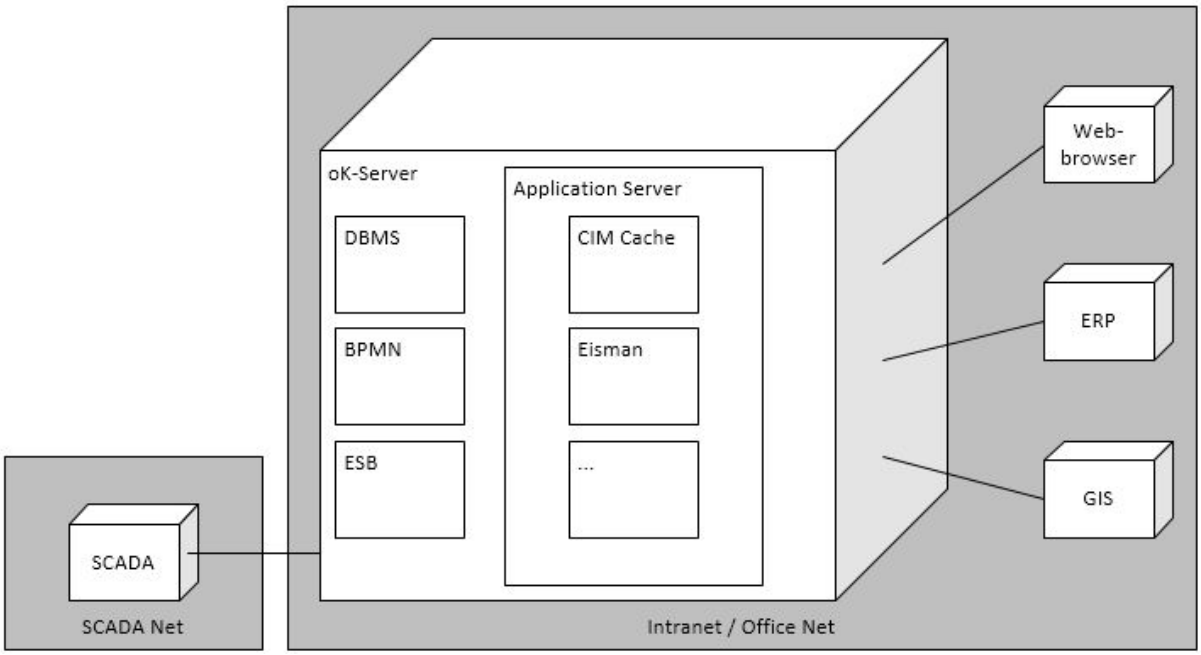
- Quality Assurance Environment - is the acceptance environment for sprint and module acceptance through the SCRUM product owner and the Lead Buyer. The module needed data of legacy systems must be available as a mockup. Once accepted, changes can be deployed into the reference system. The quality assurance environment can either be 'permanent' or 'on demand' and is owned by openKonsequenz. A quality assurance environment can easily be deployed using the reference system template within the cloud environment. Using the cloud environment specific adaptations to the hardware requirements (like cores, memory or disk) can also be validated within this environment.
- Integration Environment - is up to the developer but should be near to the quality assurance environment to ensure the same characteristics. The developer can also make use of an IaaS platform and template to deploy a template of the openK reference system. The developer is asked to have his own account to deploy the reference template within the IaaS platform used by openKonsequenz. To make it very easy for every application developer and contributor to access a copy of the reference system for integration test purpose the IBM Softlayer portal allows for easy use of the defined reference system templates.



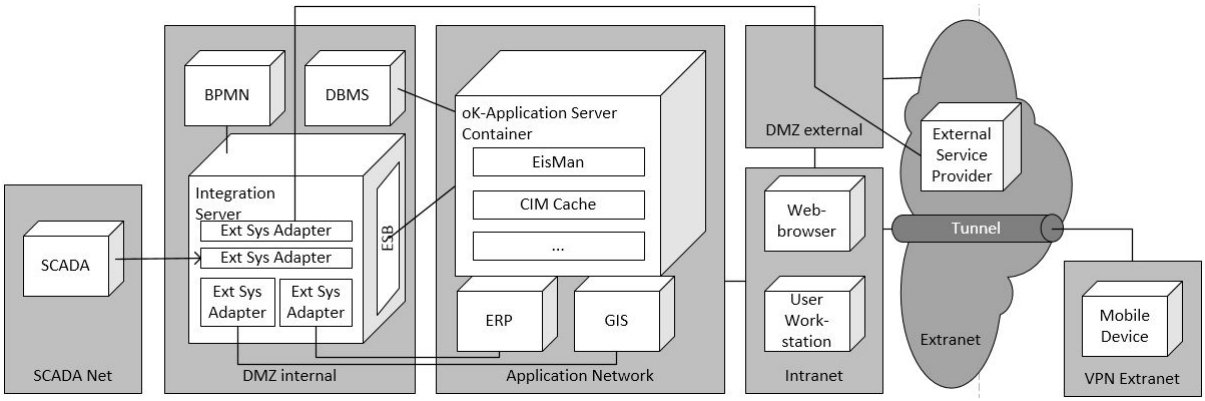

This integration system possibility from the cloud allows the application developer to run performance tests and adapt the system environment to meet the requirements and provide guidance to openKonsequenz for system sizing related to the newly designed and developed application.

- Demo Environment - it is planned to set a demo environment including existing modules. This demo environment will be accessible from external (public IP-address) and is equivalent to the reference system incl. interaction between the modules and legacy system mocks. The demo system should be a permanent system and is owned by openKonsequenz.

The reference environments associations to legacy systems are shown in the following figure. For quality assurance environment and demo environment, the external data sources (examples the figure are SCADA, ERP, GIS) are integrated in the oK-Server as system mocks.




- Standard Customer Environment - there is not standardized way, how DSO are splitting up resources on multiple hardware components (see following image for a possible scenario)




It is possible, that the SCADA System has access to the integration server, but not the other way around.

The architecture principle is based on a service oriented architecture. The decision was taken not to adopt a micro services architecture at this point in time. This may become more relevant in the future when more and more service modules and / or applications provide reusable components.

## Recommended information for developers according building up an own infrastructure or using the reference environment in an own hosted VM or on the IaaS

The openKonsequenz reference system is hosted in an IaaS (Infrastructure as a Service) environment with IBM Softlayer Cloud in Frankfurt/Main.
Image templates of the reference system will be created and made accessible.
Application developers will get access to the reference system templates. A developer can decide to use an own infrastructure, the .vhd format-templates in a virtual machine or the IaaS environment. In the latter he can easily deploy these templates for development and or testing purpose - in the case of using the IaaS environment it is typically in the responsibility of the application developer to have his own IaaS account and access. It is recommended to use at least the template in the integration environment.

TODO for architecture documentation
By module developer according to chapter 7:

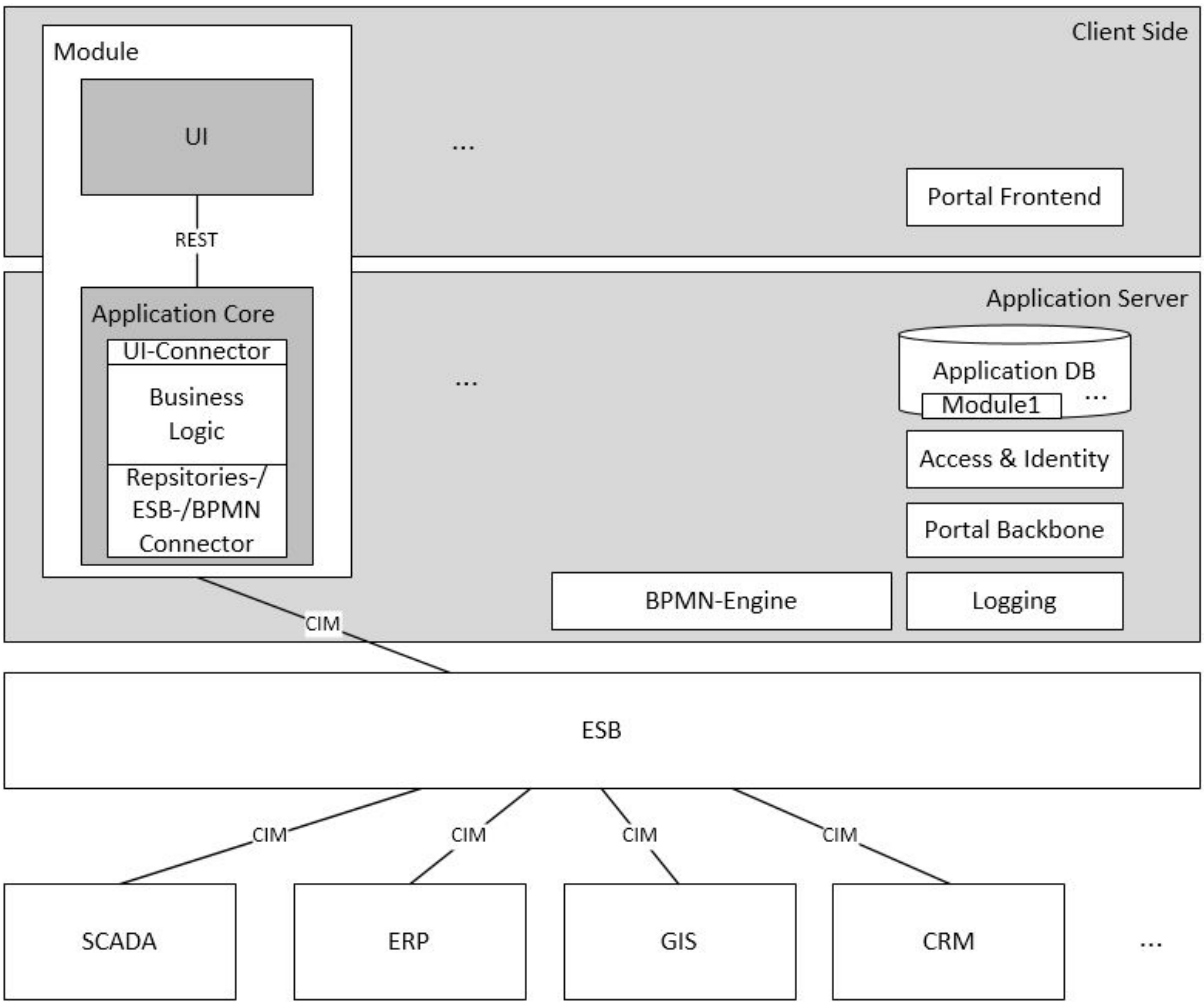
- Document how the modules builded source is deployed in the reference environment to get the module started
- Document the external or legacy system mockups (name, which data, which system(s) the data usually will come from, which external interface is used (name, version), mockups git-repository position).
- Document requirements and assumptions needed for secure system operation according to BDEW Whitepaper (if it has to be applied) chapter 2.1.2.4

# 8. Concepts

This chapter describes the general concepts and for this architecture handbook relevant detailed information - irrelevant subchapters remain to keep consistency according to the Arc42.

General concepts are:
Tiers (see also following image):

- Client-Side GUI
- Business logic on application server with interface sub tiers to client, ESB/Business process engine, and module specific persistency (The business layer (if no BPMN-process) of an application must be placed in the Java application. This layer shall explicitly not be only a pass-through layer. It needs to hold its own business objects to get separation between ESB and GUI.)
- Business process engine as possible tier between ESB and application server (Camunda is suggested by openKonsequenz AC for BPM based applications. Current applications do not yet require a BPM engine. This may change in the future. The AC is open to discuss alternative suggestions form application developers.)
- ESB
- Legacy systems adapter
- Legacy systems




Data-Exchange:

- Modules interact only via ESB and CIM interfaces (web services, REST, XML, CIM/XML (RDF)) with other oK modules or external systems.
- Each module must store its own entities in private module-DB(-scheme) (there is no data exchange between modules using the same DB scheme allowed)
- Communication between different modules or legacy systems only via ESB
- CIM-based ESB communication (IEC 61970 / IEC 61968 / IEC 619325)

Vendor-neutrality:

- Use of Open Source libraries
- Special features of ESB (and other middleware) are not used to prevent vendor-lock-in

User Interface:

- ok-GUI-Styleguide based on previous work of Minnemedia and openK pilot
  - oK-Design of minnemedia contains
    - Styleguide,
    - Html-pages
    - Bootstrap artefacts
    - JS artefacts

Development/Quality/Test:

- Development chain according to Quality Committee Handbook

In Future: Identity and Access Management (in development)

In further subchapters, details of the architecture concept are listed or have to be listed and documented for each individual module. Empty 8.x subtitles are not of importance for AC handbook but may be of importance for individual module documentation.

## 8.1 Domain Models

The interface models of the overall openKONSEQUENZ platform interface and of module interfaces shall be shown under 3.3. External Interfaces.

TODO for architecture documentation
by module developer according to Chap. 8.1:

- Domain models for the business logic of modules without relation to technology have to be documented at this point by each module developer.

## 8.2 Recurring or Generic Structures and Patterns

Intentionally left blank - not relevant at this time.

## 8.3 Persistency

Java Persistence API (JPA) reference implementation EclipseLink with PostgreSQL (without using special features for each of them to make a potential exchange of database and JPA easy). In the sharing PostgreSQL each module shall have its own database schema or more than one. Different modules are not allowed to access the same database schema. The module name must be used as prefix for the database schema names and for database users.

## 8.4 User Interface

Theok-GUI-styleguide document defines rules and regulations that each module has to follow, when it offers user interfaces. The GUI-styleguide document is not complete, and will be extended as needed. If the GUI-styleguide does not cover UI concepts needed by a module, it shall be extended during the modules development. The project may suggest an extension to the GUI-styleguide document, but additions to and changes of the GUI-styleguide document have to be reviewed, accepted and released by the architecture committee.

## 8.5 Ergonomics

See 8.4

## 8.6 Flow of Control

Intentionally left blank - not relevant at this time.

## 8.7 Transaction Procession

Intentionally left blank - not relevant at this time.

## 8.8 Session Handling

User Session Handling: User login and session management is provided by the portal (Liferay in case of oK). All modules must use the user session management of the portal to achieve an uniform experience for the user and to ease integration into DSOs environment. User session management is standardized in JSR 286 portlet specification.
If a module needs management of technical sessions (e.g. sessions with external systems, or involving multiple services), these aspects have to be described here.

TODO for architecture documentation
by module developer according to Chap. 8.8:

- Document the technical sessions management (if needed)

## 8.9 Security

According to the vision of openKONSEQUENZ (see document oK-vision) the oK-software must be secure. This chapter will contain the security concept. The concept will define the implementation of the requirements of the BDEW Whitepaper and will be aligned with its structure. For the security relevant aspects which are laid down at other locations it will contain references. With increasing amount and functionality of user modules and service modules the security concept will be extended.

### General Requirements and Housekeeping

**Cryptographic standards**
When cryptographic standards are selected, only state-of-the-art cryptographic algorithms and key lengths according to BSI TR-02102-1 shall be used. **This is particularly true for the usage of** Transport Layer Security (SSL/TLS) in connections with HyperText Transfer Protocol Secure (HTTPS) - see BSI TR-02102-2.

**Documentation**
A complete security architecture does not only comprise technical means. It also describes operational guidelines considering the available technical base as well as the personnel controlling the systems. The Documentation of Security Parameters and Security Log Events or Warnings has to be described according to chapter 5.2, 8.16 and 8.18.
The end user documentation:

- User documentation

- Administration documentation

is demanded by the QC handbook and shall include the Requirements and Assumptions needed for Secure System Operation.

The security concept is yet not considered to a sufficient extent. A task of the AC is to extend the security concept according to progress of development. Further subchapters according to security shall underlie the following structure:

- Base System
- Networks / Communication
- Application
- Development, Test and Rollout
- Backup, Recovery and Disaster Recovery

## 8.10 Safety

According to the mission statement of openKONSEQUENZ the software is located in a safety-critical environment. Until further notice, the software will not be directly coupled with or responsible for functions that might endanger human life or equipment. Safeguarding life and environment is no issue for the software.

## 8.11 Communications and Integration

Intentionally left blank - not relevant at this time.

The integration concept is yet not considered to a sufficient extent. A task of the AC is to define a concept for replacement / integration of modules.

## 8.12 Distribution

Intentionally left blank - not relevant at this time.

## 8.13 Plausibility and Validity Checks

Intentionally left blank - not relevant at this time.

## 8.14 Exception/Error Handling

Exceptions that are part of a modules external interface need to be discussed with the AC and cleared by it. Each module is responsible for its own error/exception handling as specified in the list of technologies below. All errors and exceptions shall be logged according to the logging requirements as specified in the QC Handbook.

TODO for architecture documentation
by module developer according to Chap. 8.14:

- Document which kind of exceptions and errors are handled by the system

- Document **Which kinds of errors are forwarded to which external interface** and which are handled fully internally.

## 8.15 System Management & Administration

Larger software systems are often executed in controlled environments (data centers) under oversight of operators or administrators. These stakeholders require specific information on the applications' states during runtime as well as special means of control and configuration. In oK, the development processes are separate Processes. On the one hand, oK development of prototypes and on the other hand the bilateral system integration at the DSO. Because of this, the need for a proper implementation at this point is even stronger. The system administrators need information, where and how ticket management systems can get fault information.

The system management & administration concept is yet not considered to a sufficient extent. A task of the AC is to specify a central instance for getting tickets for administration of oK-Platform.

TODO for architecture documentation
by module developer according to Chap. 8.15:

- Document where and how a ticket management system can get fault information.

## 8.16 Logging, Tracing

There are different kinds of logging. Logging for business aspects, logging for administrators, logging for developers. It is specified in the tender / by Product Owner, which events must be logged for business aspects.

TODO for architecture documentation
by module developer according to Chap. 8.16:

- What are expected messages, which meaning does they have and (if necessary) how a module differs from QC-requirements.
- Document security relevant system messages (according to BDEW Whitepaper chapter 2.1.2.3, if it has to be applied).

## 8.17 Business Rules

Intentionally left blank - not relevant at this time.

## 8.18 Configurability

- Modules have to be configurable in that way, that no rebuild is required to run code in different environments (see chapter 7. Deployment View for different

environments - in especially different operating systems and distributed servers with different access rights).

- Module specific configuration has to be done in one module-central file. All configuration parameters shall have meaningful default values. The semantics, value ranges, and interdependencies of all configuration parameters shall be documented as a part of the modules architecture description.

TODO for architecture documentation
by module developer according to Chap. 8.18:

- Check/develop module dependencies according to different environments
- Document configuration files/properties
- Provide configuration file for quality assurance environment
- Document security relevant configuration (according to BDEW Whitepaper chapter 2.1.2.3, if it has to be applied)

## 8.19 Parallelization and Threading

Intentionally left blank - not relevant at this time.

## 8.20 Internationalization

By default, all user interface elements shall be internationalized. I.e. all strings, colors, number/date formats, fonts, ... need to be configurable. Each module has to provide a german nationalization.

## 8.21 Migration

Intentionally left blank - not relevant at this time.

## 8.22 Testability

See QC handbook. Projects should support at least daily build-and-test cycles. Important keywords for this aspect are unit tests and mock objects.

## 8.23 Scaling, Clustering

Intentionally left blank - not relevant at this time.

## 8.24 High Availability

Intentionally left blank - not relevant at this time.

## 8.25 Code Generation

Intentionally left blank - not relevant at this time.

## 8.26 Build-Management

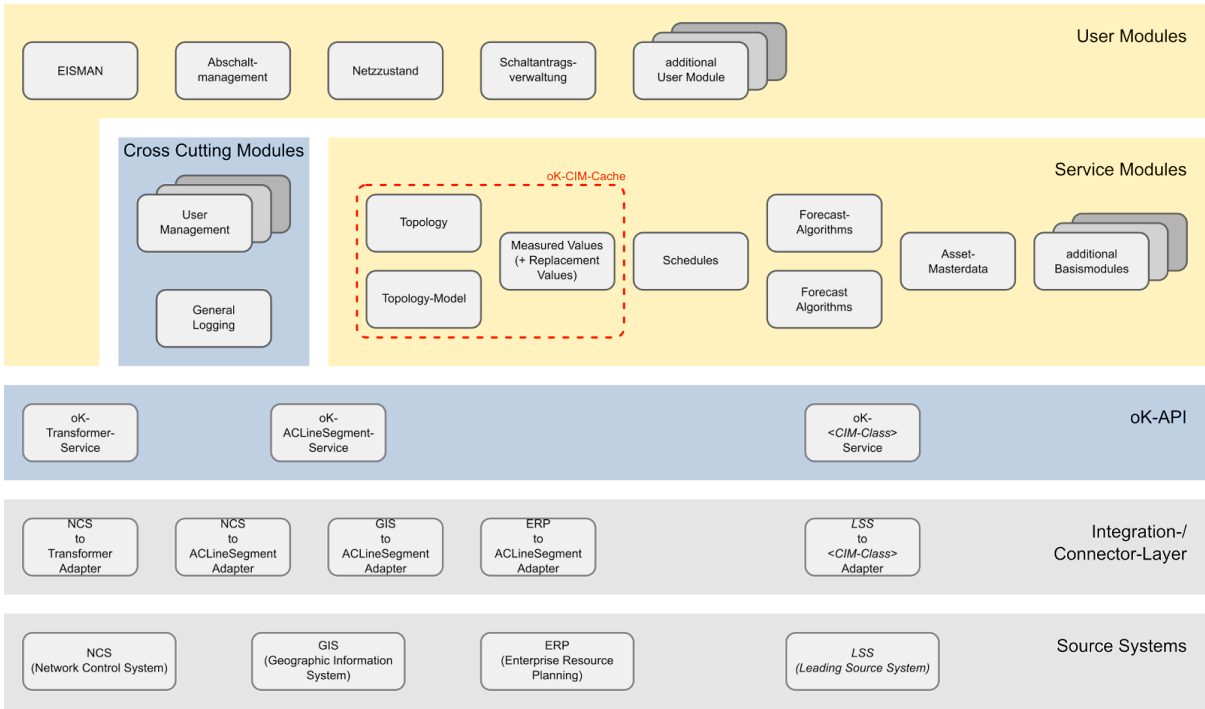
See QC handbook.

TODO for architecture documentation
by module developer according to Chap. 8.26:

- Document, how the overall system is created from its (source code) building blocks. Document, if directories are in line with the given structure in QC handbook (Which repositories contain source code, where are configuration files, test cases, test data and build scripts maven stored).

# 9. Design Decisions

Design and technical decisions are listed in the following chapter. Documentation of design decisions is useful for better understanding of existing architecture and shall avoid non-observance of existing knowledge.
In case of deviation from the openKONSEQUENZ technical decisions by a module, the module developer has to ask the ACQC committee to come to an agreement.




1. Service Components for more than one Module
    a. Problem: Several Modules need the same data/information.
    b. Constraints: Data/information shall not be saved redundant in every piece of software.
    c. Assumptions: Interfaces can be developed for exchange of such information
    d. Alternatives:
        i. Central service modules
        ii. Distributed information in modules
    e. Decision: Central services modules shall be developed for CIM-Cache. Also for Identity&Accessmanagement.
2. Module communication: Communication between modules only via ESB
    a. Problem: Modules need information from other modules
    b. Constraints: If one module changes, other modules shall not need a change.
    c. Assumptions: if modules interact directly via Databases, no module is directly responsible for the database scheme, schemes can be misunderstood or misinterpreted so inconsistency in the underlying data is supported.

    d. Alternatives: direct access to database, interface communication
    e. Decision: interface communication via ESB, because of better long term maintainability (independent Maintanance cycles between modules and seperated interchangeability during operation)
3. No multitenancy
    a. Problem: Multiple DSOs could use one module
    b. Constraints: concurrent use of modules by different DSOs
    c. Assumptions: Every DSO has its own application server for hosting modules.
    d. Alternatives: multitenancy, no multitenancy
    e. Decision: No Multitenancy
4. Portal
    a. Problem: Users shall not need to register/login for each oK-module
    b. Constraints:
        i. Single-Sign-On or Workstation Login
        ii.
    c. Assumptions: Liferay handles User Sessions, Konfiguration & Access Rights for Portal-pages.
    d. Alternatives:
        i. Liferay
        ii. oK-self implementation
    e. Decision:
        i. Liferay
5. Identity & Access Management
    a. Problem: Users shall be differentiated and have different access rights to modules.
    b. Constraints:
        i. An oK-self implementation is very cost intensive
        ii.
    c. Assumptions:
        i. Liferay can be used to manage Rights & Roles not only for GUI, but also for backend.
        ii. Liferay can incorporate existing LDAP or AD user token
    d. Alternatives:
        i. Liferay
        ii. Ok-self implementation
    e. Decision:
        i. Liferay - also for Rights&Role management with a facade (that needs to be developed).
6. ESB Talend vs. Mule
    a. Problem: Which ESB to choose
    b. Constraints:
        i. No vendor lock in, open source

ii. ESB only transport medium, no higher functions shall be used to prevent lock-in.
iii. Choice of ESB applies only for the reference platform - the DSO may use others in their production environment.
iv. Exchangeability must be ensured

c. Assumptions:
i. Open source ESB prevents vendor-lock-in and can be developed further, Talend can be used in long term.
ii.

d. Alternatives: (By previously study) Talend, Mule
e. Decision: Talend (got feedback on request)

7. ESB CIM/REST Interfaces
a. Problem: A founded way to describe DSOs data is needed
b. Constraints: Standardized Base,
c. Assumptions: RESTful Webservices can be used according to CIM Standards, REST is easier to integrate than Soap
d. Alternatives: CIM in different Versions, RESTful, Soap
e. Decision: CIM Version 17 or newer, REST

8. BPMN Camunda (not explicit specificated)
a. Problem: Process Engine
b. Constraints:
c. Assumptions:
d. Alternatives:
e. Decision: Camunda (It is suggested to use Camunda for business processes. If a software developer names an alternative, the AC will discuss this.)

9. Code-/Design-/Quality reports as .ad
a. Problem: In which format shall reports for Code, design and quality be stored (and where)
b. Constraints: Easy access (open community) - technologically and according storage
c. Assumptions: git is used as repository for code. Ascii-Documents can be opened by everyone and easily be managed in git (version history, merge)
d. Alternatives: Wiki, git, word, tex, pdf, .ad
e. Decision: Ascii-Documents in git.

10. Name - Spaceholder
a. Problem:
b. Constraints:
c. Assumptions:
d. Alternatives:
e. Decision

TODO for architecture documentation
by module developer according to Chap. 9:

- Each module has to describe its specific design decisions according to the following structure in its own architecture document:
  1. Name of decision 1
     a. Problem:
     b. Constraints:
     c. Reason for module specific and not global decision
     d. Assumptions:
     e. Alternatives:
     f. Decision:
  2. Name of decision 2
     a. Problem:
     b. Constraints:
     c. Reason for module specific and not global decision
     d. Assumptions:
     e. Alternatives:
     f. Decision:
  3. ...

# 10. Quality Scenarios

For quality KPIs and quality assurance we refer to the quality committee handbook, in which (in parts) formalized QA for continued tests is stated necessary. It gives a minimal set of test and acceptance rules and KPIs for unit tests.

The quality scenarios for the platform are not yet considered to a sufficient extent. A task of the AC is to define hard platform specific requirements for the reference platform and quality scenarios for testing these requirements.

TODO for architecture documentation
by module developer according to Chap. 10:

- Each module has to describe its own quality requirements in quality (evaluation) scenarios and quality tree for sprint and final acceptance
- Test scripts and test handbooks for acceptance of sprints/releases have to be documented.
- Test datasets (on base of oK common test data - if it exists); have to be generated, published, coordinated with ACQC and used.
- According to the SCRUM Development process, when defining quality requirements it is also necessary to define acceptance criteria (the developer shall remind the product owner to discuss it in sprint planning).
- Commisioning Tests

# 11. Technical Risks

The technical risks for the platform are not yet considered to a sufficient extent. A task of the AC is to collect and assess risks for the reference platforms architecture.

TODO for architecture documentation
by module developer according to Chap. 11:

- Document the list of identified technical risks (with probability of occurrence, amount of damage, options for risk avoidance or risk mitigation), ordered by priority

# 12. Glossary

| Short | Long (engl) | German | Description |
|---|---|---|---|
| AC | Architecture Committee | Architekturkomitee | Gives Framework and Constraints according architecture for oK projects |
| AC handbook | Architecture Committee handbook | Handbuch des Architekturkommitees | Textural guideline for module developers of openKONSEQUENZ modules according to architecture related issues (this document). |
| ACQC | Architecture Committee and Quality Committee | Architekturkomitee und Qualitätskommite | AC and QC together |
| | User Module | Fachliches Modul | An oK-application, a user from a DSO uses for solving his/her use case. |
| DSO | Distribution System Operator | Verteilnetzbetreiber (VNB) | Manages the distribution network for energy, gas or water |
| ESB | Enterprise Service Bus | | Central instance for exchange of data to overcome point-to-point connections |
| IaaS | Infrastructure as a Service | | Cloud Service for hosting. |
| IP | Intellectual Property | Geistiges Eigentum | Protections for copyright, patents,.. |

| | | | |
|---|---|---|---|
| | Service Module | Plattform Modul/Basis Modul | An oK-application that serves other oK-applications by delivering data/services |
| oK | openKONSEQUENZ | openKONSEQUENZ | Name of the consortium of DSOs |
| QA | Quality Assurance | Qualitätskontrolle | Check, if solutions fulfilling requirements to quality |
| QC | Quality Committee | Qualitätskomitee | Gives framework and constraints according to quality for oK projects |
| QC handbook | Quality Committee handbook | Handbuch des Qualitätskommitees | Textural guideline for module developers of openKONSEQUENZ modules according to quality related issues. |
| SCADA | Supervisory Control and Data Acquisition | Netzleitsystem | System, that allows DSOs view/control actual parameters of their power grid. |
| VPN | Virtual Private Network | Virtual Private Network | Extends private networks across a public network. |

TODO for architecture documentation
by module developer according to Chap. 12:

- Document important or misleading abbreviations and terms

# APPENDIX

## oK-Interface-Overview

# I Overall oK-Interface-Profile Source Position, Image and Version Information

The overall oK interface profile is hosted in the git-Repository of openK-platform (openk-platform/openk-cim-v17) as Java packages and classes with Java documentation for the semantical use of attributes/classes and as Image of an UML class diagram. If extensions are needed for modules, the module developer has to update this overall model with needed attributes/classes and their semantical meanings. Thereby it shall be granted, that module developer take the current available model as basis, that no redundant information is in the profile and that there is no semantical inconsistency introduced.

*Placeholder for Image*

**Version Information**

| **Name** | **Overall oK interface profile** |
|---|---|
| **Version** | **1.0.0** |
| **Changes w.r.t previous release** | **Initial model after Eisman I development** |
| **Who/Why/When changed it?** | **BTC - initial Concept - 2015/??/??** |
| **Contact person** | **???** |

# II Individual oK-Interfaces Overview

List of interfaces - which are provided and required by which module - with short description and link to detailed documentation.

| Name + Version | Einspeiser - translation needed 1.0 |
|---|---|
| What Data | |
| Where is detailed documentation | |
| Provider (provides) | |
| User (requires) | |

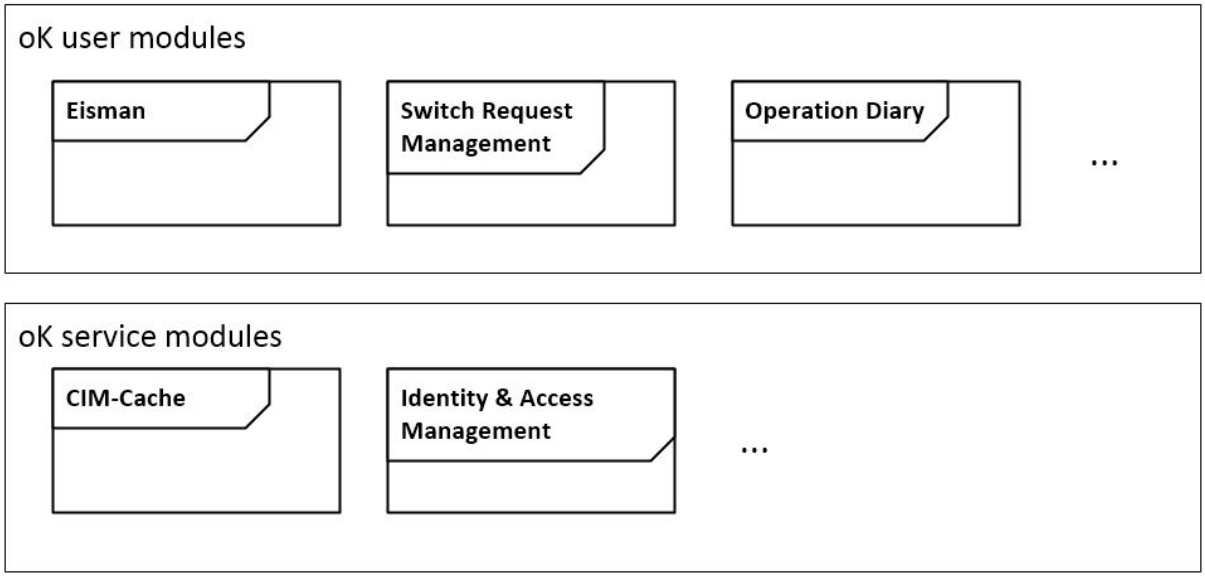
| | |
|---|---|
| Name + Version | Topology 1.0 |
| What Data | |
| Where is detailed documentation | |
| Provider (provides) | |
| User (requires) | |

| | |
|---|---|
| Name + Version | Topology_einfach - translation needed 1.0 |
| What Data | |
| Where is detailed documentation | |
| Provider (provides) | |
| User (requires) | |

| | |
|---|---|
| Name + Version | Netzzustand - translation needed 1.0 |
| What Data | |
| Where is detailed documentation | |
| Provider (provides) | |
| User (requires) | |

Standard oK-API-Interfaces are not yet considered to a sufficient extent. A task of the AC is to define the standard openKONSEQUENZ-APIs according to progress of development.

# III Building Block Level 1 View - Image and Version Information




This Image has to be transferred to a UML component diagram with requires and provides interfaces.

| | |
|---|---|
| Last Edited On Date | 2016-06-27 |
| Last Edited By | Andre Goering (during AC handbook creation) |
| Short Description | Needs to be transferred to UML component diagram, when the Tooling is fix. Needs updates according to requires and provides interfaces.<br>After development of eisman I pilot (which incorporates service module CIM-cache) and before call for tenders for Operation Diary,Switch Request Management and Identity & Access Management. |