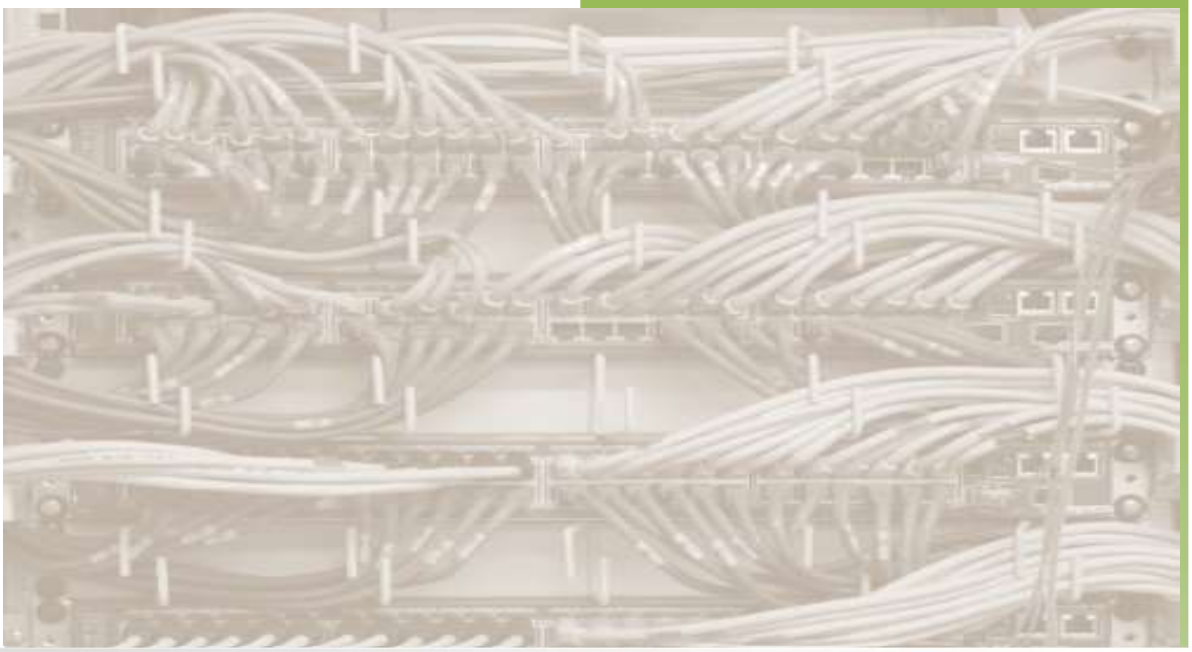

Minimum technical requirements Section: Software development and software adjustments (customising)

Responsible:

Robert Gladewitz

Kevin Glück

DBFZ Leipzig

Version: 22.03.2023

Next review: 04.2024

Validity: 04.2024

Table of Contents

1 General

1.1 Purpose of the document and general guidelines

This document defines the minimum technical requirements for software which is custom-developed or associated with other procurements (e.g. measuring devices) within the scope of tenders, projects and contracts that are directly or indirectly financed by the DBFZ. Any deviations from these minimum technical requirements must be contractually specified and agreed upon in advance with IT. It is not permissible to deviate from the written form requirement. Written or verbal changes made by the software supplier, e.g. with project sponsors or corporate entities outside of management, are not permitted.

The use of demonstrably higher-quality components than those shown is permissible and is usually regarded positively during the offer evaluation.

All technical minimum requirements (German version) of the DBFZ are available at https://www.dbfz.de/tma.

1.2 Abbreviations and definitions of terms

ADS	Active Directory Services
BSD	Berkeley Software Distribution
CSS	Cascading Style Sheets
GPL	GNU General Public License
JSON	JavaScript Object Notation
LPGL	GNU Lesser General Public License
MIT	Massachusetts Institute of Technology
PHP	Hypertext Preprocessor
Update	Updating the software as part of existing or minor new functionalities with the aim of troubleshooting or ergonomic improvements.
Upgrade	Enhancement of the software with the aim of creating new functionalities, which takes precedence over troubleshooting and ergonomic improvements.

1.3 Scope

This document describes all specifications for software developments and software adjustments (customising) which are commissioned by the DBFZ or are to be transferred to the DBFZ through the adoption of processes performed on behalf of the DBFZ.

The following document and content types are typically created during the development of software components:

- Source Code
- Machine code (executable code / libraries)
- Visualisation data
- Configuration data
- Images and texts
- Designs and design specifications
- Templates

The specifications in this document refer to all mentioned document types. Adjustments that do not meet the requirements outlined in this document must be documented, justified and included as attachments to the procurement offer. If existing software is used, any deviations from the specifications at the time of handover to the DBFZ must be documented.

License law caused exclusions or adjustments must be coordinated with DBFZ IT prerequisite to any assignment and are only possible with written confirmation from IT or management. Corresponding inquiries must be addressed to the awarding authority within the framework of formalised award procedures.

The specifications defined here also apply to website projects, cloud applications and SaaS.

2 Specifications

All document and content types created for the DBFZ within the framework of commissioned software developments or software adjustments must be delivered as free content (open content). In regards to copyright, the supplier guarantees an unrestricted, non-exclusive, single-purpose right of use which is valid indefinitely and without geographical limitations and which is transferable together with the delivery item (e.g. software sale by DBFZ).

Please note: The guidelines in this document apply without any limitations to all adjusted components for software modifications, such as the setup of building automation components or process control systems. Configuration adjustments, customising, definitions and drawings are to be applied equally to the content types defined in this document for software development.

2.1 Source codes and content types

All relevant defined documents and content types must be made available in full (source code, open source) for developed software components and software modifications.

Closed code deliveries (closed source) are only permitted if further development, publication or usage rights are not affected, for example in the case of proprietary modules or libraries. In this case, the supplier is obliged to ensure that the module can be used, adapted and developed by providing complete documentation. The documentation should include a comprehensive presentation of the processing, derivation of data and the resulting outcomes.

2.1.1 Programming language

For software development projects, the programming language to be used must be agreed in advance with the responsible individuals at the DBFZ. The programming language should be specified in the case of formalised award procedures.

Inline documentation is to be used for all source texts allowed by this tool.

A delimitable software project is to be implemented uniformly in a programming language or technology. Permissible exceptions include product-dependent specifications, such as the use of TypoScript and PHP in Typo3 projects.

2.1.2 Use of runtime environments and interpreters

The specifications in this document only apply to the newly created or adapted content when using ready-made runtime environments, modules or software components.

This is the case, for example, with embedded or special solutions (e.g. building automation, process control technology for test systems or individual automation solutions), as the manufacturer components are not co-developed and may therefore be subject to special licenses.

The specifications in this document apply without restriction to all developed adaptations. It must be ensured that the DBFZ, as the client, can continue to develop or adapt the software and systems based on the available development files (source files) without additional effort, additional software components, license costs or purchased services. This means that the necessary runtime environments, interpreters or development software with the necessary software components and their licenses must be included in the scope of the delivery.

When using runtime environments, complete documentation of all versions, modules and components used must always be provided. This also applies to stored functionalities that are not used in the related software development or software adaptation. The rights to further use of the runtime environments and interpreters are to be included in the scope of the delivery.

2.1.3 Use of frameworks and templates

Before starting a new development project, the user must always check whether a desired functionality can already be provided by existing frameworks and templates. When using frameworks and templates, the following additional specifications must be observed:

2.1.3.1 Maintenance / up-to-dateness

Maintenance must be guaranteed over the planned operating or reference period, e.g. by providing updates or upgrades. Upgrades within the subscription period are included in the purchase price.

2.1.3.2 Preferred sources for templates and frameworks

For Linux distributions or Windows, the frameworks are provided by the manufacturer/distributor. If using Windows, for example, this would correspond to the .NET Framework and if using Linux, templates such as Smarty, OpenJDK or Apache Velocity would be provided.

Templates managed by distributions are typically preferred as long as there are no functional restrictions. Care must be taken to ensure that compatibility is maintained with newer versions that become available, such as those provided through distribution updates and upgrades.

2.1.3.3 Graphical elements and interfaces

Different resolutions and formats need to be taken into account when developing barrier-free graphic interfaces and websites. The graphical interface must adapt flexibly to the various input devices, such as smartphones, tablets, laptops and desktop computers, and must therefore be usable on all devices without restrictions. A flexible CSS template system such as Bootstrap or Foundation is required as a basis. When using Javascript, ready-made libraries such as jQuery or comparable technologies must be used.

2.1.3.4 Technologies

Current frameworks typically use the technology provided by manufacturers. The following technology is currently preferred:

- Microsoft .Net 64Bit
- OpenJDK (>20) 64Bit
- SOAP / Web Services
- HTML5 / XHTML / JSON
- CSS3 / jQuery or comparable JavaScript libraries
- BAC Net

2.1.3.5 Prohibited technology

Technology must not be used if its discontinuation has already been announced by the manufacturer. The following technology in particular should be mentioned here:

- Microsoft COM / DCOM
- Microsoft Active X
- RPC (binary transfer)
- Oracle Java 1.8 or older

The use of discontinued technology is possible as an exception if existing projects are developed further or if this enables the use of existing hardware components. Usage must be agreed with DBFZ IT as early as possible in the planning stage, but at the latest before the start of the procurement process.

2.1.3.6 Architectures (64Bit)

The development of new applications and fundamental updates to existing applications are to be implemented exclusively on the basis of 64-bit technology. The use of 64-bit framework libraries or development environments is mandatory. Permitted architectures:

- AMD64 (X64)
- Power64
- Mips64
- Sparc64
- ARM (64)

2.1.3.7 Protection in the event of version updates of dependent libraries

When creating software, care must be taken to ensure that the source text is always fully compatible with newer versions of the libraries used. For .Net, Java, and PHP, for example, early features are marked as obsolete or deprecated. The use of functions already marked as "obsolete" or "deprecated" during the development phase is expressly prohibited.

2.1.4 Programming style / code style

The programming style recommended by the programming language should be used to create code. The specifications from ISO/IEC9126 must also be observed and implemented.

If the DBFZ has, for example, defined its own programming style for a project, this must be communicated when the order is placed and used by the contractor.

Temporary source code lines must always be documented directly with their risks in the source text. This also applies when using workarounds. The responsible individual at the DBFZ must also be informed immediately about the application of such changes in the source code.

2.1.5 Documentation

In addition to inline documentation, full documentation must also be created and provided to the DBFZ. Both printed and electronic documents are permitted. The documentation must be up-to-date with the software provided.

Reference to online documentation is only permitted in the case of standard products such as programming environments, templates or frameworks. This is expressly not permitted for the document content generated in the order.

The following table describes the documentation content.

Documentation Part	Absolutely Necessary	Notices / Restrictions
Objective	Yes	
Development documentation (function diagrams, etc.)	Yes	All common diagram forms (structogram, Jackson diagram, function diagram, etc.) are permitted
Purpose	Optional	
Organisation of the file structure	Yes	
Documentation of the additional modules	Yes	In the case of closed code, all functions, including those modules not used in development, must be documented
Method documentation	Yes	In addition to the inline documentation
Installation documentation	Yes	
Data documentation	Yes	Data types
User documentation	Yes	Documentation for use, for the end user.
Test documentation	Optional	

Table 1: Content for documentation

Both functions (procedures, subprograms) and the variables used must typically be documented.

2.2 Use rights

In all software projects, the creator/contractor guarantees that, upon payment of the financial interests, there is an unrestricted right of use for all document and content types specified in the defined scope.

The DBFZ generally assumes that further internal development or further development by third parties is permitted without restrictions.

For additional modules with restrictions, the contractor must guarantee the unlimited right of use. Time restrictions or restrictions on persons or devices are not permitted.

2.3 Transfer

The DBFZ reserves the right to pass on the document and content types defined in the scope without restrictions or financial interests of the DBFZ. The DBFZ is responsible for the conditions of transfer.

2.4 Publication

All software projects have the unrestricted right to publish the document and content types defined in the scope. This right is not limited to scientific work.

2.5 Alternative permitted types of license for created code

The client may specify the form of the license to a limited extent. The following license models can be used:

License	Comments
GPL V2	Only if Version 3 is not possible due to the components used
GPL V3	
BSD license	
Apache license	Only after consultation with the management, the specific contractual conditions must be defined
LGPL	Only after consultation with the project manager
MIT license	

Table 2: Permitted license models

Separate license models for the libraries used and the necessary programs (e.g. development environments) are permitted, but must not affect the specifications of this document for the newly created content.

2.6 Security

During the development of safety-critical applications and services, it is essential to ensure the provision of a subsequent operation and maintenance phase. The costs must be determined by a maintenance contract drawn up during the tender.

2.6.1 Encryption

Encryption which is considered to be secure must be used for the development of software in the case of communication components. If full encryption is not possible due to the components

or techniques used, the authentication must at least be secured by means of secure encryption. The transmission of hash authentication data (SHA etc.) is not permitted. The following symmetric encryption methods are currently considered secure:

Encryption	Comment
AES (Rijndael)	Minimum key length 128, recommended 256
Twofish	
MARS	
Serpent	

Table 3: Symmetric encryption types

Unless otherwise technically or contractually specified, a minimum key length of 256 bits must generally be used. The methods RC4, DES and 3DES, which are still frequently used, are not permitted.

The following methods are permitted for applications with asymmetric encryption:

- RSA (key length min. 2048 Bit)
- Elgamal

A combination of asymmetric and symmetric encryption is permitted and desirable if this increases the security of the software.

Secure methods must be used for a key agreement with secured connections. The key exchange protocol Diffie-Hellman key exchange ensures the security of SSL connections. Permanently assigned shared keys are generally only permitted if required by the technology.

2.6.2 Checksum methods for passwords and hashsums

The following specifications are defined for checksums.

Procedure	**Passwords**	**Files/data**	**Comment**
MD5	No	Yes	
****SHA1**	No	Yes	
SHA2			

Table 4: Hash sum types

**SHA1 can be used for passwords in exceptional cases after consultation with IT if this is necessary due to compatibility restrictions or limitations in the development environment.

“Salt and pepper” methods should be used whenever possible to increase security. The use of both methods together is also permitted.

2.7 Maintenance and defect resolution

A contract offer for software maintenance is to be included as part of the software development contract bid during a procurement procedure.

A maintenance contract should typically be concluded for all software components for which a contractual term is to be expected. The DBFZ is responsible for deciding whether to conclude a maintenance contract.

In the case of websites or technical systems, the framework contract for maintenance must be defined in such a way that reaction, service and recovery times are specified. The debugging itself must be covered by a maintenance contract at no additional cost.

The response and recovery times to be specified are defined in the following table.

Bug category	Reaction time	Recovery time
Bug that prevents operation	24 hours	48 hours
Bug that affects operation	24 hours	48 hours
Minor bug	48 hours	one week

Table 5: Response and recovery times

2.8 Security vulnerabilities

When developing or adapting software, care must be taken to ensure the use of the latest version with the status "stable" or an LTS version of the templates or frameworks it is based on.

If security vulnerabilities, discovered by the manufacturer or other institutes such as DFN-CERT, actively being exploited they must be promptly resolved within a maximum of two days.

2.9 Use of trusted repositories and development sources

Software development must generally be implemented using distribution-based repositories. Manufacturer-maintained repositories can be used if they correspond to the principles of stable software provision.

A prerequisite for such a source is that all delivery mechanisms are published, checked, accredited and standardised. Transparent security report and security response mechanisms must also be provided in public bug reports and corresponding CVE publications.

Closed reporting and monitoring systems for security errors are expressly prohibited.

Examples of trusted Repositories

Name	Maintained by	Comment
Ubuntu	Distribution	Debian-based, canonical, required for LTS license
Debian	Distribution	
Node source	Manufacturer/Community	deb.nodesource.com
RedHat	Distribution	
IBM	Manufacturer	
Zabbix	Manufacturer	

Table 6: Trusted repositories

The approval of new repositories must be confirmed by IT and the IT security officer. Approvals must be documented in writing in the DBFZ ticket system and must take place in a defined system.

2.10 Restrictions on untrusted sources

Special restrictions and guidelines apply to the use of untrustworthy sources. Untrustworthy sources refer to those sources that lack documented monitoring and security reporting, publish only partial CVEs and do not generate DFN Cert reports for errors and bugs. In particular, these include:

- GIT (all Git sources, GitHub/GitLab)
- Docker (with the exception of signed vendor-supplied containers)
- PIP
- PPA
- NPM (Javascript)
- PEAR (PHP)
- Anaconda, Miniconda
- Jupyter
- Private sources

The mentioned sources can only be used for internal projects if provision of standard packages through trusted repositories is not feasible.

2.10.1 Untrustworthy sources for public services

The use of these sources is not permitted for software used in published applications and public services.

2.10.2 Untrustworthy sources for internal services

If untrustworthy sources need to be used for internal projects and software projects, the corresponding versions used (each version!) must be archived in storage areas provided by the DBFZ. It must be ensured that such sources are not used without prior archiving.

Periodic CVE scans must also be set up and performed. If software parts can be compromised due to known vulnerabilities, these must be made available immediately by updating and the necessary adjustments must be made.

If immediate action is not feasible, the DBFZ must be informed and the service in question must be deactivated or the affected software must be deleted or deactivated for the users.