<This form is used for all interface designs and contains generic chapters like Document introduction, purpose, etc. and depending on the type of interface (to be mentioned in introduction dedicated chapters have been added to this form.

* For Software specific interfaces these sections are identified as <Sw specific>
* For Electrical and or Mechanical interfaces these sections are identified as <…>
* For External interfaces…

Guideline for documenting a Software Interface[[1]](#footnote-2)

* Keep in mind who will be using the interface documentation and what types of information they will need.
* Avoid documenting more than is necessary. For example, you probably need less detail in the interface documentation of a module used only by another developer on the team than you need for an interface that is part of a commercially available API.
* This form describes the “maximum” approach, that is, a fully documented interface. Depending on the importance of the interface, you should decrease the amount of information and the effort spent in the interface documentation.
* Focus on how elements interact with their environments, not on how elements are implemented. Restrict the documentation to effects that are externally visible.
* Expose only what users of the interface need to know. Including a piece of information in the documentation is an implicit promise that the information is reliable and stable. Once information is exposed, other elements rely on it, and changes will have a more widespread effect.
* Be as specific and as precise as you can, remembering that interface documentation that can be interpreted differently by various parties is likely to cause problems and confusion.

>

|  |  |
| --- | --- |
| Author: |  |
|  |  |
|  |  |
|  |  |
| Department: | Research & Development |

# Approval

| **Role** | (approval via electronic signature in document management tool) |
| --- | --- |
| R&D: Department Manager SD |
| R&D: Strategic Architect |
| R&D: Platform Architect |
| Q&R: Dev QA Officer |

# Revision History

| **Revision Date** | **Author** | **Source** | **Changes/Comments** |
| --- | --- | --- | --- |
| <yyyy Mon dd> | <Author> | <Source ID> <E.g. DCO#, ECR/ECO#, EIW#> | <Description of change>  <Creation: Standard text “Initial version”  Change: Describe the modification(s) and the impact here or refer to specific section in the document. Important element of reviewing!> |
|  |  |  |  |

<List the changes, date and author related to the ‘Source’. The ‘Source’ refers to the reason for the change.>

# Open Issues

<

State “None” in this section, if there are no open issues.

If there are open issues: the author can decide to define an open issue bullet list or to create an open issue table of contents to have a direct link to the open issue in the document.

Open issues have to be clarified in a future release.

It is recommended to define 'due date' as well as 'owner' for all open issues, and to track the open issues in the progress meeting.

Open issue1 header.  
Issue 1 content, due date, and owner

Open issue 2 header  
Issue 2 content, due date, and owner

……..

……..

Open issue n header  
Issue n content, due date, and owner

OR

**Open issue table of contents**

Open issue 1 header ………………………………………………page xx

….

….

Open issue n header ………………………………………………page xy

>

# Table of Contents

[Approval 1](#_Toc484641490)

[Revision History 1](#_Toc484641491)

[Open Issues 1](#_Toc484641492)

[Table of Contents 1](#_Toc484641493)

[1. Document Introduction 1](#_Toc484641494)

[1.1. Purpose 1](#_Toc484641495)

[1.2. Scope and stakeholders 1](#_Toc484641496)

[1.3. Brief description and intended use 1](#_Toc484641497)

[1.4. How to read this document 1](#_Toc484641498)

[1.5. References 1](#_Toc484641499)

[1.6. Definitions & Abbreviations 1](#_Toc484641500)

[2. Rationale 1](#_Toc484641501)

[3. Versioning 1](#_Toc484641502)

[4. Syntax <SW only> 1](#_Toc484641503)

[4.1. Data types and constants 1](#_Toc484641504)

[4.2. Signature 1](#_Toc484641505)

[5. Semantics <SW only> 1](#_Toc484641506)

[5.1. State behavior 1](#_Toc484641507)

[5.2. Timing constrains 1](#_Toc484641508)

[6. Your chapters 1](#_Toc484641509)

[7. Usage guide <optional> 1](#_Toc484641510)

[8. Deployment 1](#_Toc484641511)

[9. Design issues 1](#_Toc484641512)

[10. Bibliography <optional> 1](#_Toc484641513)

[Control of the Form 1](#_Toc484641514)

[History of the Form 1](#_Toc484641515)

# Document Introduction

## Purpose

<Describe the purpose of the document and its related objectives. What is the goal of the document, why is it important. Preferably, use a bullet list and start with a verb>

<Use an explicit interface symbol if any element has more than one interface and you wish to emphasize the interface for an element: for example, if you are making provisions for multiple elements that realize the same interface.>

## Scope and stakeholders

<Describe the boundaries of the applicability of the document.

Example: This document is applicable to all project X members.>

Examples of stakeholders of interface documentation with the required information[[2]](#footnote-3):

* A developer creating the interface, who needs the most comprehensive documentation of the interface the element provides. The developer needs to see any assertions about the interface that he or she will realize in the code. A special kind of developer is the maintainer, who makes assigned changes to the element and its interface.
* A tester, who needs detailed information about all the resources and functionality provided by an interface. The tester can test only to the degree of knowledge embodied in the interface description. If required behavior for a resource is not specified, the tester will not know to test for it, and the element may fail to do its job.
* A developer using an interface, who needs detailed information about the resources provided in the interface to implement elements that will use it. A special case is the integrator, who puts the system together from its constituent elements and has a stronger interest in the behavior of the resulting assembly. In a software product-line context, this stakeholder exploits the variability available in the elements to build different products.
* Analysts, whose information needs depend on the types of analyses conducted. For a performance analyst, for example, the interface document should give information that can feed a performance model, such as execution time required by resources.
* Architects looking for assets to reuse in a new system, which often starts by examining the interfaces of elements from a previous system. The architect may also look in the commercial marketplace to find off-the-shelf elements that can be purchased and do the job. To see whether an element is a candidate, the architect is interested in the capabilities of the interface resources, their quality attributes, and any variability that the element provides.
* Project managers, who is likely to use interface documents for planning purposes. Project managers can apply metrics (such as function-point analysis) to gauge the complexity and then infer estimates for how long it will take to develop an element that realizes the interface. Project managers can also spot special expertise that may be required, and this will assist them in assigning the work to qualified personnel.>

## Brief description and intended use

<Defined the type of interface described in the document like Software Interface, Mechanical Interface, Electrical Interface or a combination of this.

## How to read this document

<Legend of used notations, items, colors, specific syntax used>

## References

<Use [REF-number] as Reference. Do not use hyperlinks as Reference/Identification. Use same title as defined in the archiving system. Empty Identifications are not allowed. When only DHF-number is used latest revision is referred. Use none when no references are used.>

| Reference | Identification | Title / additional remarks |
| --- | --- | --- |
| [REF-<1>] | <Identification ID.  E.g. DHF# (+Rev#)> | <Title of document> |
| [REF-<n>] |  | <Title of document> |
|  |  |  |
|  |  |  |

## Definitions & Abbreviations

<Use this section only if applicable. Use it for definitions and abbreviations used in this document, which are not trivial.>

| Term | Description |
| --- | --- |
| <def> | <Definition explanatory text> |
| <abbr> | <Abbreviation is written out> |
|  |  |
|  |  |

# Rationale

< Describe the business rationale for the architecture and the architecture views at large and answer the question: Why do we need this interface? >

# Versioning

<Describe the versioning strategy; version exchange mechanisms for client and server side, if any; version of the interface applicable for this document. How are the different interface versions managed in the installed base, forward compatibility of server and backwards compatibility of client and vice versa>

|  |  |
| --- | --- |
| **Interface** | **Version** |

|  |  |
| --- | --- |
|  |  |

# Syntax <SW only>

<Includes any information needed to write a syntactically correct program that uses the resource. The syntax includes the name of the resource, names and data types of arguments, if any, structure or data type of return values, if any, (network) communication concepts including protocol design, layering, naming conventions and so forth>

## Data types and constants

< Sometimes we need to create new data types (such as records, structs, classes, enumerations, or unions) for the data passed to or returned by resources in the interface. These data types may be defined in the scope of the interface and should be described in the interface documentation. For example, in an airline reservation system, interface IReservation may provide a resource makeReservation() that returns a new data type ReservationRecord. This new data type described in the interface documentation may contain flight number, departure date and time, seat assignment, class, fare, and other data elements. Also custom defined error codes should be described here.

If the data type is defined by another element, a reference to the definition in that element’s documentation is sufficient.

In any case, programmers writing elements using such a resource need to know

1. how to declare and assign values to variables of the data type,
2. what operations and comparisons may be performed on members of the data type,
3. how to convert values of the data type into other data types, where appropriate.

Likewise, new constants are sometimes created in interfaces to hold commonly used values and make programming against the interface more convenient. For example, interface Sequencer of the Java sound API has an operation setLoopCount(int count) to set the number of repetitions of the loop for playback on a MIDI device. For convenience, the interface defines a constant called LOOP\_CONTINUOUSLY that can be passed as an argument to that operation>

### Global User-Defined Types

#### PRIMITIVE TYPES

|  |
| --- |
|  |
|  |
|  |

#### ENUMS

|  |
| --- |
|  |
|  |

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

#### RECORD TYPES

|  |
| --- |
| **extends** |
|  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
|  |

#### VECTOR TYPES

|  |  |
| --- | --- |
| = |  |

#### Primitive Type Mappings

|  |  |
| --- | --- |
| **Primitive type** | **Target C++ type** |

|  |  |
| --- | --- |
|  |  |

### Interface

#### PRIMITIVE TYPES

|  |
| --- |
|  |
|  |
|  |

#### ENUMS

|  |
| --- |
|  |
|  |

|  |  |
| --- | --- |
|  |  |

|  |
| --- |
|  |

#### RECORD TYPES

|  |
| --- |
| **extends** |
|  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
|  |

#### VECTOR TYPES

|  |  |
| --- | --- |
| = |  |

#### Primitive Type Mappings

|  |  |
| --- | --- |
| **Primitive type** | **Target C++ type** |

|  |  |
| --- | --- |
|  |  |

## Signature

<A signature consists of the function prototype. What it tells you is the general information about a function, its name, parameters, what scope it is in, and other miscellaneous information. Apart from the methods, arguments, return values, also the usage, pre-conditions, post conditions, and object invariants should be described in the form of contracts[[3]](#footnote-4). Also do not forget to mention exceptions are handled>

### Interface

#### COMMANDS

|  |  |
| --- | --- |
| ( , ) | |
| Description |  |

|  |  |
| --- | --- |
| Parameter |  |

|  |  |
| --- | --- |
| Returns |  |

|  |
| --- |
|  |

#### SIGNALS

|  |  |
| --- | --- |
| ( , ) | |
| Description |  |

|  |  |
| --- | --- |
| Parameter |  |

|  |
| --- |
|  |

#### EVENTS

|  |  |
| --- | --- |
| ( , ) | |
| Description |  |

|  |  |
| --- | --- |
| Parameter |  |

|  |
| --- |
|  |

# Semantics <SW only>

< What is the result of using the interface methods? What does the method do from the perspective of the function invoking it? Describe in this Section both the happy flow and bad weather scenarios>

<Semantics come in a variety of guises, including: [[4]](#footnote-5):

1. **Assignment** of values to the parameters and returned values, including their purpose and semantics. The value assignment might be as simple as setting the value of a return argument or as far-reaching as updating a database table
2. **External changes** in the visible state brought about by using the resource. For example, invoking a resource called open() on interface IConnection may change the state of the connection to enable it to start exchanging data. Are these changes persistent or transient? If transient, what is the duration or termination condition?
3. **Events** that will be signaled or messages that will be sent as a result of using the method.
4. The **side effects** on other environmental methods as the result of using this resource. For example, if you ask a resource to destroy an object, trying to access that object in the future through other resources will produce quite a different outcome—an error— as a result
5. **Humanly observable results**. For example, calling a program that turns on a display in a cockpit has a very observable effect: the display comes on.
6. Whether the execution of the resource will be **atomic or may be suspended or interrupted**, and whether the interaction is **synchronous or asynchronous**, if such a distinction is applicable.
7. **Usage restrictions**. Under what circumstances may this resource be used? Perhaps data must be initialized before it can be read, or perhaps a particular method cannot be invoked unless another is invoked first. Perhaps there is a limit on the number of actors that can interact via this resource at any instant. Perhaps there is a limit of one actor that has ownership and is able to modify the element, whereas others have only read access. Perhaps the resource is thread safe; that is; it can be invoked simultaneously by multiple actors. Usage restrictions are sometimes documented by defining exceptions that will be raised if the restrictions are violated.>

< You need to document what quality attribute characteristics, such as performance or reliability, the interface makes known to the element’s users. This information may be in the form of constraints on implementations of elements that will realize the interface. The qualities you choose to concentrate on and make promises about will depend on the context. For example, it may specify that certain operations should provide a specific response time, availability level, and capacity in terms of number of concurrent requests. >

## Interface

### State Behavior

|  |  |  |
| --- | --- | --- |
| **Global Variables** | | |
| **Type** | **Name** | **Comment** |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Variable Initializations** | | |
| **Variable** | **Value** | **Comment** |

|  |  |  |
| --- | --- | --- |
|  |  |  |

#### STATE MACHINE

Transitions defined in every state except states , :

|  |  |  |
| --- | --- | --- |
| **Event** | **Guard** | **Actions** |

|  |  |  |
| --- | --- | --- |
| -( , ) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Initial State** | | | |
| **Event** | **Guard** | **Target State** | **Actions** |

|  |  |  |  |
| --- | --- | --- | --- |
| -( , ) |  |  |  |

### Timing constraints

#### Constraint

### Data constraints

#### Constraint

# Your chapters

<Add here your dedicated interface design content>

# Usage guide <optional>

< It’s helpful to complement the interface specification with examples that show the usage protocol for one or more methods of the interface. Code snippets are common in the usage guide, but sequence diagrams and other behavioral diagrams up to fully executable models are good choices, especially when a certain sequence of steps for the resource usage is required. Try to craft some clear and simple examples of the most common ways the interface might be used. >

# Deployment

<It is convenient to document a set of assumptions that the element’s designer has made about the execution platform and its technology. In this form, they can be reviewed by experts who can confirm or repudiate the assumptions before the design has progressed too far.>

# Design issues

<Record the reasons behind the design of the interface. The rationale should explain the motivation behind the design, constraints and compromises, alternative designs that were considered and rejected and why, and any insight the architect has about how to change the interface in the future>

# Bibliography <optional>

<It is allowed to add a bibliography section in a document to refer to ‘sources’ used to create the content of a document. The referred sources are informative and not under control of IGT Systems (e.g. literature study report). The reference can be for instance a filename of the attachment (part of an electronic document) or ISBN number. Do not use a hyperlink as source and take care that the Intellectual Property or Copyrights are not violated.

| **Reference** | **Source** |
| --- | --- |
| [BIB-<1>] | <ISBN number: Book title> |
| [BIB-<2>] | <System Design Specification - Product Name (baseline attached)> |
|  |  |
| [BIB-<n>] | <Title of document> |

The removal of a source or a change in the content of these informative source shall never affect the basic safety or essential performance of the product.>

<Always put the Philips shield at the bottom of the last page of the document.

To meet the Philips branding requirements, and to prevent the shield from wandering around, the Philips shield has been put in a 3-row table with fixed row heights.

Keep the Philips shield and the proprietary statement together!>

|  |
| --- |
|  |
| Description: Description: Description: Description: Shield_RGB_2013 |
|  |
| Philips’ proprietary information. Unauthorized use is prohibited. |

**<DELETE THIS PAGE BEFORE FINALIZING THE DOCUMENT CREATED FROM THIS FORM>**

<This page is NOT part of the controlled document you create from this form/template. This section contains information on the History and Control of the form/template.>

# Control of the Form

<Choose either Form or Template as section header.>

|  |  |
| --- | --- |
| Approver | (approval via electronic signature in document management tool) |
| GEN: Process Owner |
| Q&R: Q&R Manager |

# History of the Form

<Choose either Form or Template as section header>

| Revision | Doc. Date | Author | CR-ID | Description |
| --- | --- | --- | --- | --- |
| 03 | 2017 Jun 30 | Rob Albers,  Benno Tonissen | 2017/xxx | * Comply with new format * Added dedicated software interface design specification sections with guidance. |

<List the changes and dates of approved revisions of the form/template. List three revisions maximum, with the most recent change mentioned first. Approved revisions are identified as 'NumberNumber', starting with 00. Always include the change request identification number under CR-ID. Make sure the date in the table matches the form / template date in the footer of the document.>

1. Bachman, Felix, et al. Documenting Software Architecture: Documenting Interfaces. SEI, 2002. [↑](#footnote-ref-2)
2. Bachman, Felix, et al. Documenting Software Architecture: Documenting Interfaces. SEI, 2002. [↑](#footnote-ref-3)
3. https://en.wikipedia.org/wiki/Design\_by\_contract [↑](#footnote-ref-4)
4. Bachman, Felix, et al. Documenting Software Architecture: Documenting Interfaces. SEI, 2002. [↑](#footnote-ref-5)