Specification File of {m:self.name}

Document generated automatically by M2Doc from Capella model of the system

# Table of Contents

{m:'newTableOfContent'.asPagination()}

# System {m:self.name}

Description:

{m:if self.description.trim().size() <> 0}

{m:self.description.trim().fromHTMLBodyString().replaceLink(self)}

{m:else}

*No description*

{m:endif}

# Description of the system Missions

{m:if self.containedSystemAnalysis.ownedMissionPkg.eAllContents(ctx::Mission)->size() > 0}

{m:if self.containedSystemAnalysis.ownedMissionPkg.eAllContents(ctx::MissionPkg)->size() > 0}

The system Missions are sorted in the following packages:

|  |  |
| --- | --- |
| **Mission package** | **Missions** |
| {m:self.containedSystemAnalysis.ownedMissionPkg.name} | {m:for mission | self.containedSystemAnalysis.ownedMissionPkg.ownedMissions}  {m:mission.name.asBookmarkRef(mission.id)}  {m:endfor} |

{m:for package | self.containedSystemAnalysis.ownedMissionPkg.eAllContents(ctx::MissionPkg)}

|  |  |
| --- | --- |
| {m:package.name} | {m:for mission |package.ownedMissions}  {m:mission.name.asBookmarkRef(mission.id)}  {m:endfor} |

{m:endfor}

{m:endif}

{m:for mission | self.containedSystemAnalysis.ownedMissionPkg.eAllContents(ctx::Mission)}

## Mission : {m:mission.name.asBookmark(mission.id)}

Description:

{m:if mission.description.trim().size() <> 0}

{m:mission.description.trim().fromHTMLBodyString().replaceLink(mission)}

{m:else}

*No description*

{m:endif}

Involved actors:

{m:if mission.involvedSystemComponents->select(sc | sc.actor)->size() > 0}

{m:for actor | mission.involvedSystemComponents->select(sc | sc.actor)}

* {m:actor.name.asBookmarkRef(actor.id)}

{m:endfor}

{m:else}

*No actor involved*

{m:endif}

Exploited capabilities:

{m:if mission.exploitedCapabilities->size() > 0}

{m:for capability | mission.exploitedCapabilities}

### Capability : {m:capability.name.asBookmark(capability.id)}

Description:

{m:if capability.description.trim().size() <> 0}

{m:capability.description.trim().fromHTMLBodyString().replaceLink(capability)}

{m:else}

*No description*

{m:endif}

Capability inclusion relations:

|  |  |  |
| --- | --- | --- |
| **Including capabilities** | **Current capability** | **Included capabilities** |
| {m:if capability.includingAbstractCapabilities->size() > 0}  {m:for cap | capability.includingAbstractCapabilities}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No including capability*  {m:endif} | {m:capability.name} | {m:if capability.includedAbstractCapabilities->size() > 0}  {m:for cap | capability.includedAbstractCapabilities}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No included capability*  {m:endif} |

Capability extension relations:

|  |  |  |
| --- | --- | --- |
| **Extended capabilities** | **Current capability** | **Extending capabilities** |
| {m:if capability.extendedAbstractCapabilities->size() > 0}  {m:for cap | capability.extendedAbstractCapabilities}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No extended capability*  {m:endif} | {m:capability.name} | {m:if capability.extendingAbstractCapabilities->size() > 0}  {m:for cap | capability.extendingAbstractCapabilities}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No extending capability*  {m:endif} |

Capability generalization relations:

|  |  |  |
| --- | --- | --- |
| **Super capabilities** | **Current capability** | **Sub capabilities** |
| {m:if capability.super->size() > 0}  {m:for cap | capability.super}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No super capability*  {m:endif} | {m:capability.name} | {m:if capability.sub->size() > 0}  {m:for cap | capability.sub}   * {m:cap.name.asBookmarkRef(cap.id)}   {m:endfor}  {m:else}  *No sub capability*  {m:endif} |

Involved functions:

{m:if capability.involvedAbstractFunctions->size() > 0}

{m:for SF | capability.involvedAbstractFunctions}

* {m:SF.name.asBookmarkRef(SF.id)}

{m:endfor}

{m:else}

*No involved function*

{m:endif}

Involved functional chains:

{m:if capability.involvedFunctionalChains->size() > 0}

{m:for FC | capability.involvedFunctionalChains}

* {m:FC.name.asBookmarkRef(FC.id)}

{m:endfor}

{m:else}

*No involved functional chain*

{m:endif}

{m:if capability.isRepresentationDescriptionName('System Data Flow Blank')}

{m:capability.asImageByRepresentationDescriptionName('System Data Flow Blank')->first().setWidth(400)}

{m:endif}

Involved actors:

{m:if capability.involvedSystemComponents->select(sc | sc.actor)->size() > 0}

{m:for actor | capability.involvedSystemComponents->select(sc | sc.actor)}

* {m:actor.name.asBookmarkRef(actor.id)}

{m:endfor}

{m:else}

*No involved actor*

{m:endif}

Available in modes and states:

{m:if capability.availableInStates->size() > 0}

{m:for state | capability.availableInStates}

* {m:state.name}

{m:endfor}

{m:else}

*No state or mode availability defined*

{m:endif}

Involved scenarios:

{m:if capability.ownedScenarios->size() > 0}

{m:for scenario | capability.ownedScenarios}

* {m:scenario.name.asBookmarkRef(scenario.id)}

{m:endfor}

{m:else}

*No involved scenario*

{m:endif}

{m:endfor}

{m:else}

*No exploited capability*

{m:endif}

{m:endfor}

{m:else}

*No mission defined*

{m:endif}

# Description of the system Capabilities

{m:for package |self.containedSystemAnalysis.containedCapabilityPkg}

{m:for im | package.asImageByRepresentationDescriptionName('Missions Capabilities Blank')}

{m:im.setWidth(400)}

{m:endfor}

{m:endfor}

{m:if self.containedSystemAnalysis.containedCapabilityPkg.eAllContents(ctx::Capability)->size() > 0}

{m:if self.containedSystemAnalysis.containedCapabilityPkg.eAllContents(ctx::CapabilityPkg)->size() > 0}

The system Capabilities are sorted in the following packages:

|  |  |
| --- | --- |
| **Capability package** | **Capabilities** |
| {m:self.containedSystemAnalysis.containedCapabilityPkg.name} | {m:for capability | self.containedSystemAnalysis.containedCapabilityPkg.ownedCapabilities}  {m:capability.name.asBookmarkRef(capability.id)}  {m:endfor} |

{m:for package | self.containedSystemAnalysis.containedCapabilityPkg.eAllContents(ctx::CapabilityPkg)}

|  |  |
| --- | --- |
| {m:package.name} | {m:for capability |package.ownedCapabilities}  {m:capability.name.asBookmarkRef(capability.id)}  {m:endfor} |

{m:endfor}

{m:endif}

{m:for capability | self.containedSystemAnalysis.containedCapabilityPkg.eAllContents(ctx::Capability)}

{m:endfor}

{m:else}

*No capability defined*

{m:endif}

# Description of the System and its Environment

## Architecture Blank Diagrams

{m:for im | self.containedSystemAnalysis.ownedSystemComponentPkg->closure(cPkg | cPkg.ownedSystemComponentPkgs).ownedSystemComponents.asImageByRepresentationDescriptionName('System Architecture Blank')}

{m:im.setWidth(400)}

{m:endfor}

{m:for sys | self.containedSystemAnalysis. ownedSystemComponentPkg->closure(cPkg | cPkg.ownedSystemComponentPkgs).ownedSystemComponents->select(sComp | not sComp.actor)}

## System: {m:sys.name.asBookmark(sys.id)}

Modes & State Machines:

{m:if sys.ownedStateMachines->notEmpty()}

{m:for stateMachine | sys.ownedStateMachines}

* {m:stateMachine.name}

{m:if stateMachine.ownedRegions ->notEmpty() and stateMachine.ownedRegions->first().isRepresentationDescriptionName('Modes & States')}

{m:for im | stateMachine.ownedRegions->first().asImageByRepresentationDescriptionName('Modes & States')}

{m:im.setWidth(400)}

{m:endfor}

{m:elseif stateMachine.ownedRegions ->notEmpty() and stateMachine.ownedRegions->first().isRepresentationDescriptionName('Mode State Machine')}

{m:for im | stateMachine.ownedRegions->first().asImageByRepresentationDescriptionName('Mode State Machine')}

{m:im.setWidth(400)}

{m:endfor}

{m:endif}

{m:endfor}

{m:else}

*No modes & states machine*

{m:endif}

Allocated functions:

{m:if sys.allocatedFunctions->size() > 0}

{m:for SF | sys.allocatedFunctions}

* {m:SF.name.asBookmarkRef(SF.id)}

{m:endfor}

{m:else}

*No allocated function*

{m:endif}

{m:endfor}

{m:for actor | self.containedSystemAnalysis.ownedSystemComponentPkg->closure(cPkg | cPkg.ownedSystemComponentPkgs).ownedSystemComponents->select(sComp | sComp.actor)}

## Actor: {m:actor.name.asBookmark(actor.id)}

Modes & State Machines:

{m:if actor.ownedStateMachines->notEmpty()}

{m:for stateMachine | actor.ownedStateMachines}

* {m:stateMachine.name}

{m:if stateMachine.ownedRegions ->notEmpty() and stateMachine.ownedRegions->first().isRepresentationDescriptionName('Modes & States')}

{m:for im | stateMachine.ownedRegions->first().asImageByRepresentationDescriptionName('Modes & States')}

{m:im.setWidth(400)}

{m:endfor}

{m:elseif stateMachine.ownedRegions ->notEmpty() and stateMachine.ownedRegions->first().isRepresentationDescriptionName('Mode State Machine')}

{m:for im | stateMachine.ownedRegions->first().asImageByRepresentationDescriptionName('Mode State Machine')}

{m:im.setWidth(400)}

{m:endfor}

{m:endif}

{m:endfor}

{m:else}

*No modes & states machine*

{m:endif}

Allocated functions:

{m:if actor.allocatedFunctions->size() > 0}

{m:for SF | actor.allocatedFunctions}

* {m:SF.name.asBookmarkRef(SF.id)}

{m:endfor}

{m:else}

*No allocated function*

{m:endif}

{m:endfor}

# Description of the Functions

{m:if self.containedSystemAnalysis.containedSystemFunctionPkg.ownedSystemFunctions.eAllContents(ctx::SystemFunction)->size() > 0}

The Functions are sorted in the following packages:

|  |  |  |
| --- | --- | --- |
| **Parent element** | **Function package** | **Contained functions** |
| {m:self.containedSystemAnalysis.containedSystemFunctionPkg.name} | {m:for package | self.containedSystemAnalysis.containedSystemFunctionPkg.ownedSystemFunctions}  {m:package.name.asBookmark(package.id)}  {m:endfor} | {m:for SF | self.containedSystemAnalysis. containedSystemFunctionPkg.ownedSystemFunctions.containedSystemFunctions}  {m:SF.name.asBookmarkRef(SF.id)}  {m:endfor} |

{m:for package | self.containedSystemAnalysis.containedSystemFunctionPkg.eAllContents(ctx::SystemFunctionPkg)}

|  |  |  |
| --- | --- | --- |
| {m:package.eContainer(capellacore::NamedElement).name.asBookmarkRef(package.eContainer(capellacore::NamedElement).id)} | {m:package.name.asBookmark(package.id)} | {m:for SF |package.ownedSystemFunctions}  {m:SF.name.asBookmarkRef(SF.id)}  {m:endfor} |

{m:endfor}

{m:for SF | self.containedSystemAnalysis.containedSystemFunctionPkg.ownedSystemFunctions.eAllContents(ctx::SystemFunction)}

## Function: {m:SF.name.asBookmark(SF.id)}

Description:

{m:if SF.description.trim().size() <> 0}

{m:SF.description.trim().fromHTMLBodyString().replaceLink(SF)}

{m:else}

*No description*

{m:endif}

Parent / Children functions:

|  |  |  |
| --- | --- | --- |
| **Parent function** | **Current function** | **Children functions** |
| {m:SF.eContainer(capellacore::NamedElement).name.asBookmarkRef(SF.eContainer(capellacore::NamedElement).id)} | {m:SF.name} | {m:if SF.subFunctions->size() > 0}  {m:for children | SF.subFunctions}   * {m:children.name.asBookmarkRef(children.id)}   {m:endfor}  {m:else}  *No children function*  {m:endif} |

List of inputs:

{m:if SF.incoming ->size() > 0}

| **Incoming functional exchange** | **Exchanged items** |
| --- | --- |

{m:for fe | SF.incoming->filter(fa::FunctionalExchange)}

|  |  |
| --- | --- |
| {m:fe.name} | {m:if fe.exchangedItems->size() > 0}  {m:for ei | fe.exchangedItems}   * {m:ei.name.asBookmarkRef(ei.id)}   {m:endfor}  {m:else}  *No exchanged item*  {m:endif} |

{m:endfor}

{m:else}

*No input*

{m:endif}

List of outputs:

{m:if SF.outgoing->size() > 0}

| **Outgoing functional exchange** | **Exchanged items** |
| --- | --- |

{m:for fe | SF.outgoing->filter(fa::FunctionalExchange)}

|  |  |
| --- | --- |
| {m:fe.name} | {m:if fe.exchangedItems->size() > 0}  {m:for ei | fe.exchangedItems}   * {m:ei.name.asBookmarkRef(ei.id)}   {m:endfor}  {m:else}  *No exchanged item*  {m:endif} |

{m:endfor}

{m:else}

*No output*

{m:endif}

Participation to functional chains:

{m:if SF.involvingFunctionalChains->size() > 0}

{m:for FC | SF.involvingFunctionalChains}

* {m:FC.name.asBookmarkRef(FC.id)}

{m:endfor}

{m:else}

*None*

{m:endif}

Allocated to:

{m:if SF.allocatingSystemComponents->select(sComp | not sComp.actor)->size() > 0}

{m:for sys | SF.allocatingSystemComponents->select(sComp | not sComp.actor)}

* {m:sys.name.asBookmarkRef(sys.id)}

{m:endfor}

{m:elseif SF.allocatingSystemComponents->select(sComp | sComp.actor)->size() > 0}

{m:for act | SF.allocatingSystemComponents->select(sComp | sComp.actor)}

* {m:act.name.asBookmarkRef(act.id)}

{m:endfor}

{m:else}

*Function not allocated*

{m:endif}

{m:endfor}

{m:else}

*No function defined*

{m:endif}

# Description of the Functional Chains

{m:if self.containedSystemAnalysis.eAllContents(fa::FunctionalChain)->size() > 0}

{m:for FC | self.containedSystemAnalysis.eAllContents(fa::FunctionalChain)}

## Functional Chain : {m:FC.name.asBookmark(FC.id)}

{m:if FC.isRepresentationDescriptionName('Functional Chain Description')}

This functional chain is illustrated by the following diagram:

{m:FC.asImageByRepresentationDescriptionName('Functional Chain Description')->first().setWidth(400)}

{m:endif}

Description:

{m:if FC.description.trim().size() <> 0}

{m:FC.description.trim().fromHTMLBodyString().replaceLink(FC)}

{m:else}

*No description*

{m:endif}

Involved elements:

{m:if FC.involvedFunctions->size() > 0}

This functional chain involves the following elements:

| **Functional exchange** | **Source function** | **Target function** |
| --- | --- | --- |

{m:for FE | FC. involvedFunctionalExchanges}

|  |  |  |
| --- | --- | --- |
| {m:FE.name} | {m:FE.source.eContainer(capellacore::NamedElement).name.asBookmarkRef(FE.source.eContainer(capellacore::NamedElement).id)} | {m:FE.target.eContainer(capellacore::NamedElement).name.asBookmarkRef(FE.target.eContainer(capellacore::NamedElement).id)} |

{m:endfor}

{m:else}

*No involved elements*

{m:endif}

Involving capabilities:

{m:if FC.involvingCapabilities->size() > 0}

{m:for capability | FC.involvingCapabilities}

* {m:capability.name.asBookmarkRef(capability.id)}

{m:endfor}

{m:else}

*No involving capability*

{m:endif}

{m:endfor}

{m:else}

*No functional chain defined*

{m:endif}

# Description of Scenarios

{m:if self.containedSystemAnalysis.containedCapabilityPkg. eAllContents(interaction::Scenario)->size() > 0}

{m:for scenario | self.containedSystemAnalysis.containedCapabilityPkg. eAllContents(interaction::Scenario)}

## Scenario: {m:scenario.name.asBookmark(scenario.id)}

Description:

{m:if scenario.description.trim().size() <> 0}

{m:scenario.description.trim().fromHTMLBodyString().replaceLink(scenario)}

{m:else}

*No description*

{m:endif}

Allocated capability:

* {m:scenario.eContainer(capellacore::NamedElement).name.asBookmarkRef(scenario.eContainer(capellacore::NamedElement).id)}

Representation:

{m:if scenario.isRepresentationDescriptionName('Component Exchanges Scenario')}

{m:scenario.asImageByRepresentationDescriptionName('Component Exchanges Scenario')->first().setWidth(400)}

{m:elseif scenario.isRepresentationDescriptionName('Functional Scenario')}

{m:scenario.asImageByRepresentationDescriptionName('Functional Scenario')->first().setWidth(400)}

{m:endif}

Owned messages:

{m:if scenario.ownedMessages->size() > 0}

{m:for message | scenario.ownedMessages}

* {m:message.name}

{m:endfor}

{m:else}

*No message*

{m:endif}

{m:endfor}

{m:else}

*No scenario defined*

{m:endif}

# Data definition

{m:for im | self.containedSystemAnalysis.ownedDataPkg.asImageByRepresentationDescriptionName('Class Diagram Blank')}

{m:im.setWidth(400)}

{m:endfor}

{m:for package | self.containedSystemAnalysis.ownedDataPkg}

## {m:package.name}

{m:if package.ownedExchangeItems->size() > 0}

Exchange items definition:

{m:for EI | package.ownedExchangeItems}

{m:EI.name.asBookmark(EI.id)}

{m:if EI.ownedElements->size() > 0}

|  |  |
| --- | --- |
| **Owned element** | **Element type** |

{m:for element |EI.ownedElements}

|  |  |
| --- | --- |
| {m:element.name} | [{m:element.type.name}] |

{m:endfor}

{m:else}

*No contained elements*

{m:endif}

{m:endfor}

{m:endif}

{m:if package.ownedCollections->size() > 0}

Collections definition:

|  |  |
| --- | --- |
| **Collection** | **Collection type** |

{m:for collection | package.ownedCollections}

|  |  |
| --- | --- |
| {m:collection.name} | [{m:collection.type.name}] |

{m:endfor}

{m:endif}

{m:if package.ownedClasses->size() > 0}

Classes definition:

{m:for class | package.ownedClasses}

{m:class.name}

{m:if class.containedProperties->size() > 0}

|  |  |
| --- | --- |
| **Contained property** | **Property type** |

{m:for prop |class.containedProperties}

|  |  |
| --- | --- |
| {m:prop.name} | [{m:prop.type.name}] |

{m:endfor}

{m:else}

*No contained property*

{m:endif}

{m:endfor}

{m:endif}

{m:if package.ownedDataTypes->size() > 0}

Data types definition:

{m:for data | package.ownedDataTypes}

* {m:data.name}

{m:endfor}

{m:endif}

{m:endfor}

{m:for package | self.containedSystemAnalysis.ownedDataPkg.eAllContents(information::DataPkg)}

## {m:package.name}

{m:if package.ownedExchangeItems->size() > 0}

Exchange items definition:

{m:for EI | package.ownedExchangeItems}

{m:EI.name.asBookmark(EI.id)}

{m:if EI.ownedElements->size() > 0}

|  |  |
| --- | --- |
| **Owned element** | **Element type** |

{m:for element |EI.ownedElements}

|  |  |
| --- | --- |
| {m:element.name} | [{m:element.type.name}] |

{m:endfor}

{m:else}

*No contained elements*

{m:endif}

{m:endfor}

{m:endif}

{m:if package.ownedCollections->size() > 0}

Collections definition:

|  |  |
| --- | --- |
| **Collection** | **Collection type** |

{m:for collection | package.ownedCollections}

|  |  |
| --- | --- |
| {m:collection.name} | [{m:collection.type.name}] |

{m:endfor}

{m:endif}

{m:if package.ownedClasses->size() > 0}

Classes definition:

{m:for class | package.ownedClasses}

{m:class.name}

{m:if class.containedProperties->size() > 0}

|  |  |
| --- | --- |
| **Contained property** | **Property type** |

{m:for prop |class.containedProperties}

|  |  |
| --- | --- |
| {m:prop.name} | [{m:prop.type.name}] |

{m:endfor}

{m:else}

*No contained property*

{m:endif}

{m:endfor}

{m:endif}

{m:if package.ownedDataTypes->size() > 0}

Data types definition:

{m:for data | package.ownedDataTypes}

* {m:data.name}

{m:endfor}

{m:endif}

{m:endfor}