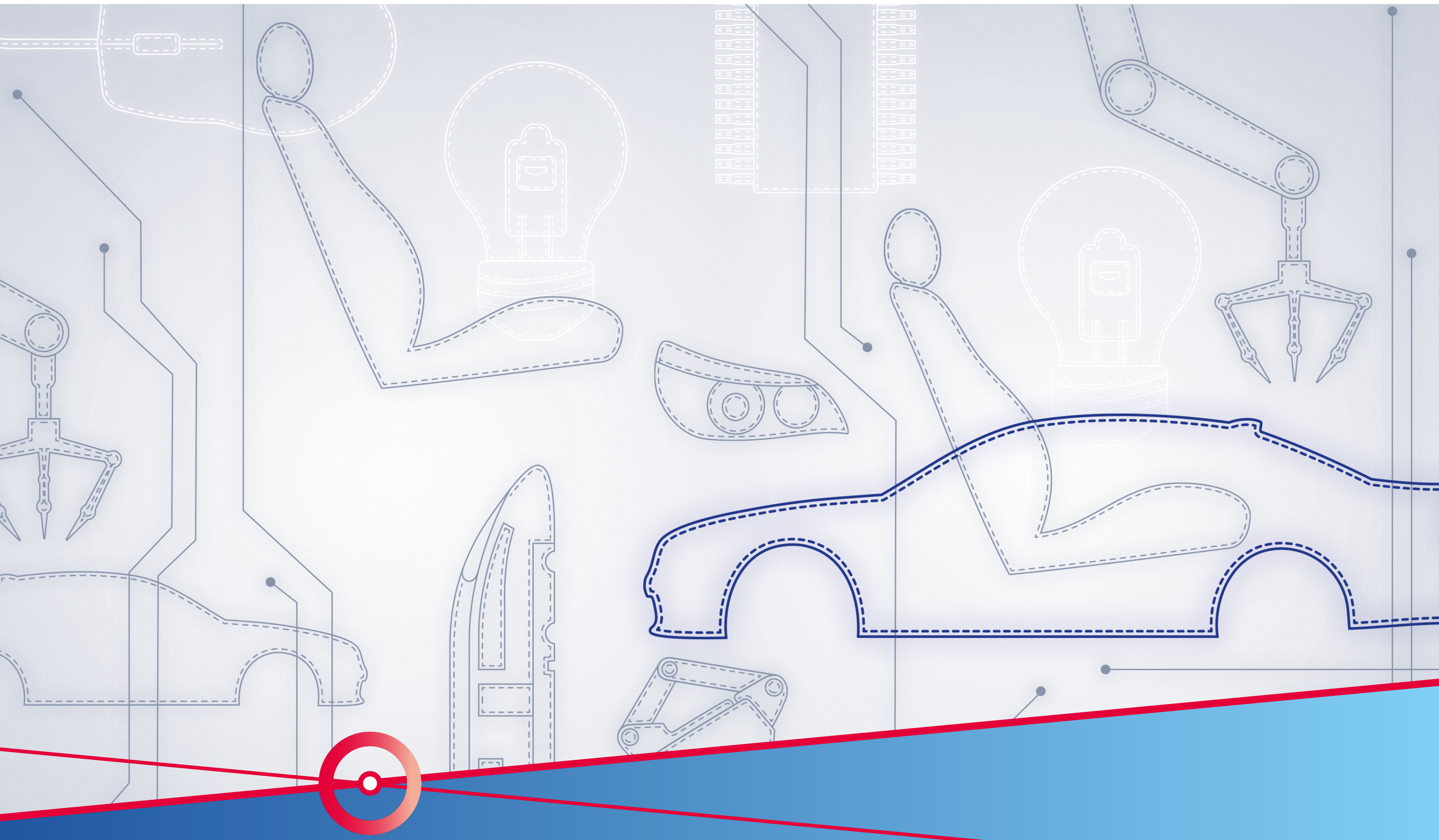
DRIVING EMBEDDED EXCELLENCE



RTA Application Note

PDU Group Handling with BswM

RTA-CAR

Copyright

Document: AN-000/EN-05-2019

Contents

1 Introduction

1.1 Scope

This application note describes how to configure RTA-BSW in order to switch on/off I-PDUs communication.

A possible use case for this strategy is a case where the user needs to handle different Variants, and each Variant is characterized by different I-PDUs (and therefore each Variant has a different dbc associated).

Generally speaking, the strategy explained in this AN can be applicable in all the cases where the ASW needs to detect a change in the system/ECU status and consequently restrict/extend the communication turning off/on some of the I-PDUs.

1.2 Assumptions

1.2.1 Software

You must have the following ETAS software installed:

- ISOLAR-AB 5.0.1
- RTA-BSW 3.2

This application note could be used as a reference for any generic AR based project.

1.2.2 Prerequisites

An assumption of this Application Note is that the user is familiar with the BswM module and he is also familiar with Mode Management. The AN contains indications on how to configure BswM for Mode Management, but it is out of scope to provide theoretical details and concepts on Mode Management.

For theoretical concepts and a more detailed overview on BswM Mode Management please refer to the Technical Note: **“TN_AR_010 Communication Control (UDS service 0x28) implementation with RTA-BSW 3.1”**.

2 Strategy details

In this section it is detailed the strategy to follow in order to handle different variants, characterized by different dbc files, from the communication point of view.

The use case is the following:

1. The user has more ECU variants to handle
2. Each ECU variant is characterized by different dbc files
3. After the startup phase, the ASW shall be able to turn on just the I-PDUs that are associated to the current ECU Variant, and turn off all the other not relevant I-PDUs.

The strategy uses the PDU Group feature from Com Module and BswM Mode Management. The PDUs are grouped into PduGroups. Each PduGroup is associated to a specific variant.

This strategy has some advantages:

- It is possible to handle different variants using the Mode Management
- It easy and fast to configure, high reusability

The main requirements to implement this strategy are:

1. All PDUs and Signals shall be defined in the System
2. PDUs and Signals shall have a unique name (AUTOSAR requirement)

In the proposed strategy, the actors are:

1. ASW: A SWC must act as Mode Requester
    - Check the current variant
    - Request to the BswM to switch to the Mode foreseen for that particular variant
2. Com Module
    - Receives the request from the BswM to switch on/off one or more PDU groups
3. BswM as Mode Manager
    - The BswM handles the Mode Switch request
    - The BswM send the request to the Com to switch on/off the PDU groups (using the configured Action Lists)

3 Strategy Implementation

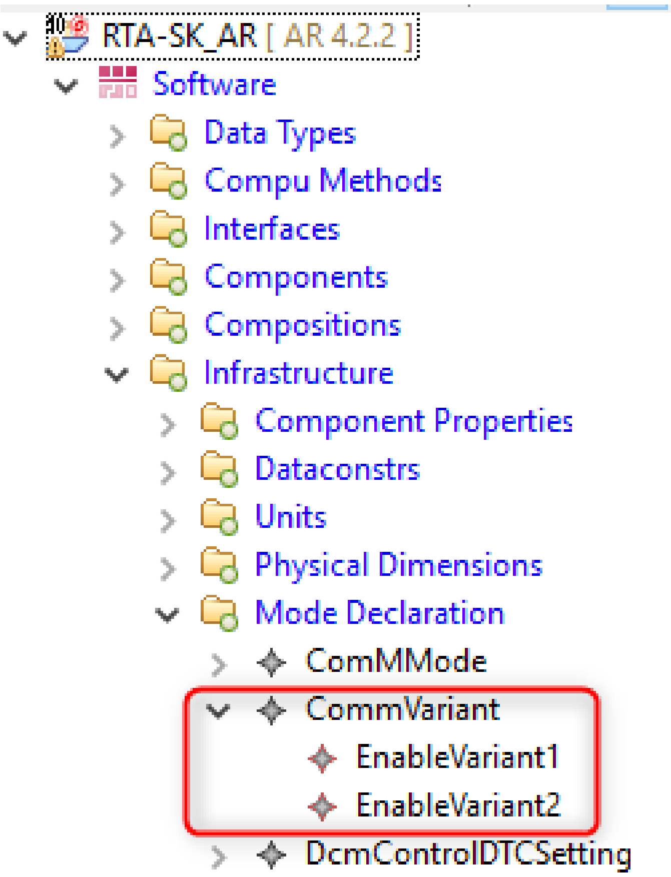
In this chapter it is described how to implement the strategy previously described, for each of the actor identified before.

3.1 Application software configuration

The ASW configuration is split in two parts. In this section we create all the ASW elements that are needed for the BswM configuration. In the second part of the ASW configuration, we will complete the workflow with the elements that will be generated after the configuration of the BSW.

3.1.1 Creation of the Mode Declaration

The first step is to create the Mode Declaration and therefore create the two modes that we need. In our example, we suppose to have to variants to handle.

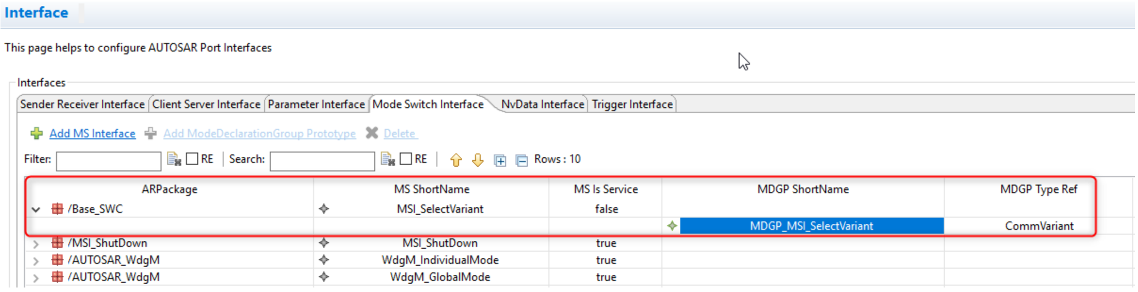


- RTA-SK_AR [AR 4.2.2]
  - Software
    - Data Types
    - Compu Methods
    - Interfaces
    - Components
    - Compositions
    - Infrastructure
      - Component Properties
      - Dataconstrs
      - Units
      - Physical Dimensions
      - Mode Declaration
        - ComMMode
        - CommVariant
          - EnableVariant1
          - EnableVariant2
        - DcmControlDTCSetting

3.1.2 Creation of the Interface

Now it is possible to create the interface needed by the ASW to communicate the mode switch request to the BSW.

In order to create this interface, double click on an existing interface to open the editor. Switch to the tab “Mode Switch Interface” and an MS interface and a Mode Declaration Group Prototype as shown in the screenshot:



Interface

This page helps to configure AUTOSAR Port Interfaces

Interfaces

Sender Receiver Interface | Client Server Interface | Parameter Interface | Mode Switch Interface | NvData Interface | Trigger Interface

Add MS Interface | Add ModeDeclarationGroup Prototype | Delete

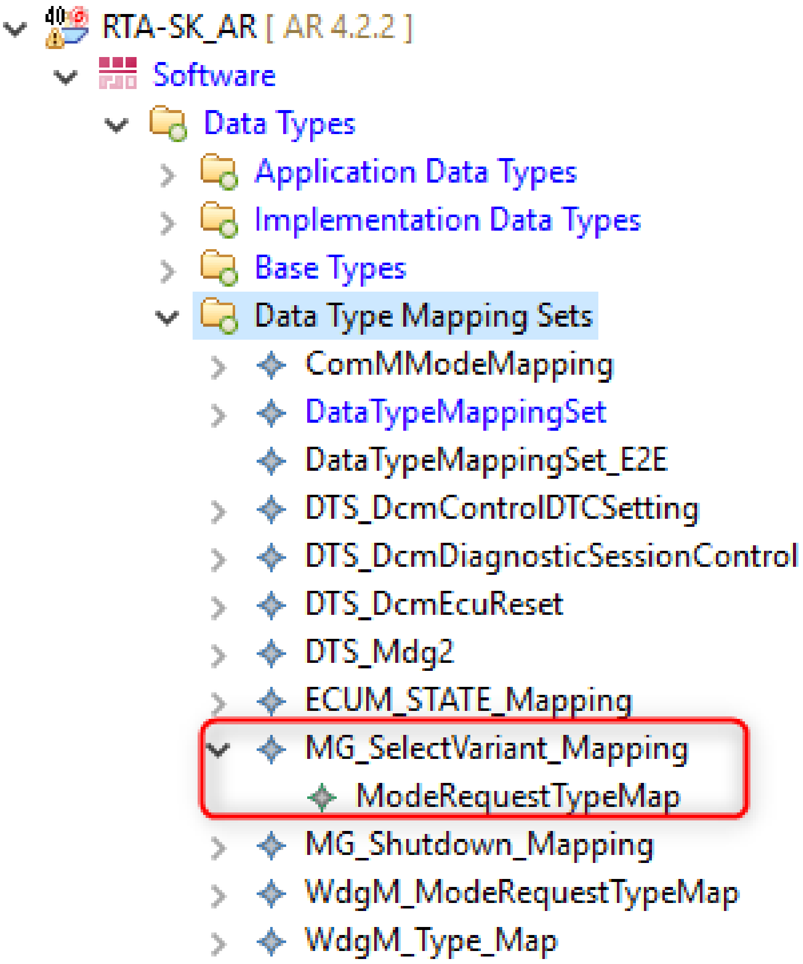
Filter: □ RE | Search: □ RE | Rows : 10

ARPackage	MS ShortName	MS Is Service	MDGP ShortName	MDGP Type Ref
/Base_SWC	MSI_SelectVariant	false		
			MDGP_MSI_SelectVariant	CommVariant
/MSI_ShutDown	MSI_ShutDown	true		
/AUTOSAR_WdgM	WdgM_IndividualMode	true		
/AUTOSAR_WdgM	WdgM_GlobalMode	true		

3.1.3 Creation of the Data Mapping

Another step needed before to configure the BSW is to create the Data Mapping that will be referenced in the BswM module later on.

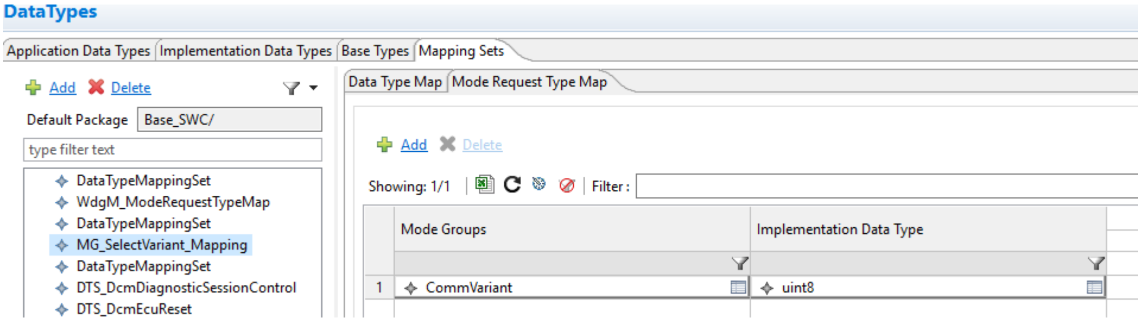
To create the Data Mapping, using the AR explore open the sub-folder “Data Type Mapping Sets” and add a new element into it, as shown below:



- RTA-SK_AR [AR 4.2.2]
  - Software
    - Data Types
      - Application Data Types
      - Implementation Data Types
      - Base Types
      - Data Type Mapping Sets
        - ComMModeMapping
        - DataTypeMappingSet
        - DataTypeMappingSet_E2E
        - DTS_DcmControlDTCSetting
        - DTS_DcmDiagnosticSessionControl
        - DTS_DcmEcuReset
        - DTS_Mdg2
        - ECUM_STATE_Mapping
        - MG_SelectVariant_Mapping
          - ModeRequestTypeMap
        - MG_Shutdown_Mapping
        - WdgM_ModeRequestTypeMap
        - WdgM_Type_Map

Double click on the created element to open the editor. Switch to the “Mode Request Type Map” and then create a new map. Assign uint8 as “implementation Data Type”, for example.

Now you have everything is needed in the ASW to proceed with the BSW configuration. Please follow the next sections for details in the configuration.


DataTypes

Application Data Types | Implementation Data Types | Base Types | Mapping Sets

Add Delete

Default Package: Base_SWC/

type filter text

- DataTypeMappingSet
- WdgM_ModeRequestTypeMap
- DataTypeMappingSet
- MG_SelectVariant_Mapping
- DataTypeMappingSet
- DTS_DcmDiagnosticSessionControl
- DTS_DcmEcuReset

Data Type Map | Mode Request Type Map

Add Delete

Showing: 1/1 Filter:

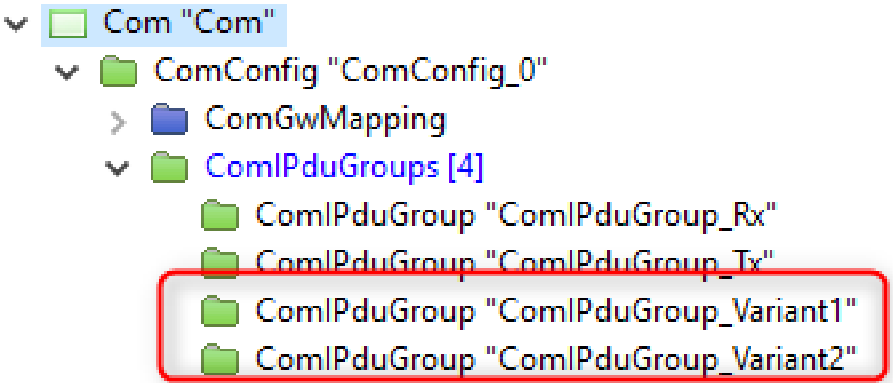
	Mode Groups	Implementation Data Type
1	CommVariant	uint8

Figure 3.1: 1557399001140

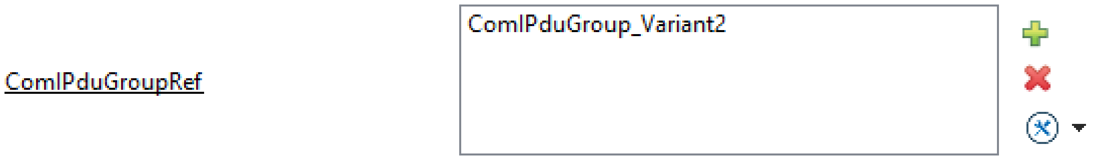
3.2 Com Module configuration

In the Com module the PDU Groups shall be created and the relative PDUs shall be added.

The first step is to create the ComIPduGroups, in our example we create two additional groups for the two variants.


- Com "Com"
  - ComConfig "ComConfig_0"
    - ComGwMapping
    - ComIPduGroups [4]
      - ComIPduGroup "ComIPduGroup_Rx"
      - ComIPduGroup "ComIPduGroup_Tx"
      - ComIPduGroup "ComIPduGroup_Variant1"
      - ComIPduGroup "ComIPduGroup_Variant2"

Now, it is possible to reference the IPDUs to a particular group. Open an existing PDU and change the reference as shown in the picture below:


ComIPduGroupRef: ComIPduGroup_Variant2

In the screenshot, you can see that the IPDU has been assigned to the group created for Variant2.

The user shall repeat the configuration assigning all the IPdu to the correct group.

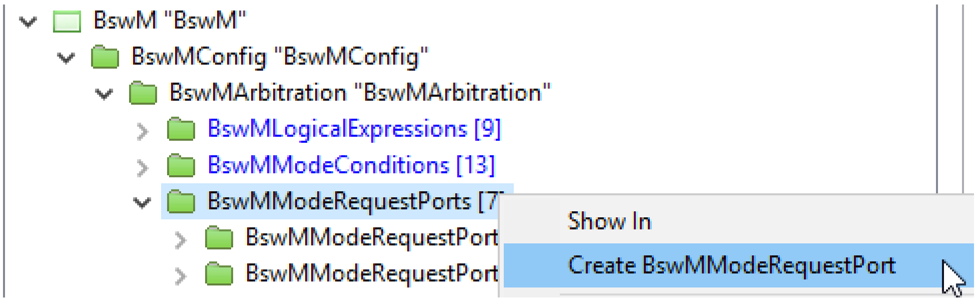
3.3 BswM as Mode Manager

As previously described, the BswM shall act as a Mode Manager. The BswM is the only BSW module that is, according to AUTOSAR specifications, able to call the Com API *Com_IpduGroupControl* and enable/disable the transmission of one or more PDU groups.

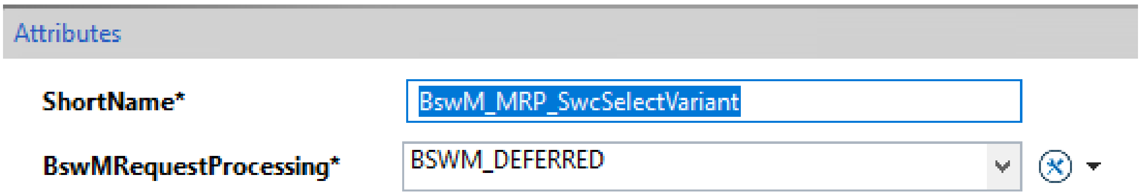
In order to do so, you need to follow the configuration described in this section.

3.3.1 Adding a new ModeRequestPort

The Mode Request Port must be added to the BswM configuration. This port is needed by the BswM in order to receive the mode switch request from the ASW. Please add the new BswMModeRequestPort as shown in the picture below.



Call the created MRP “BswM_MRP_SwcSelectVariant” and configure it as shown below:

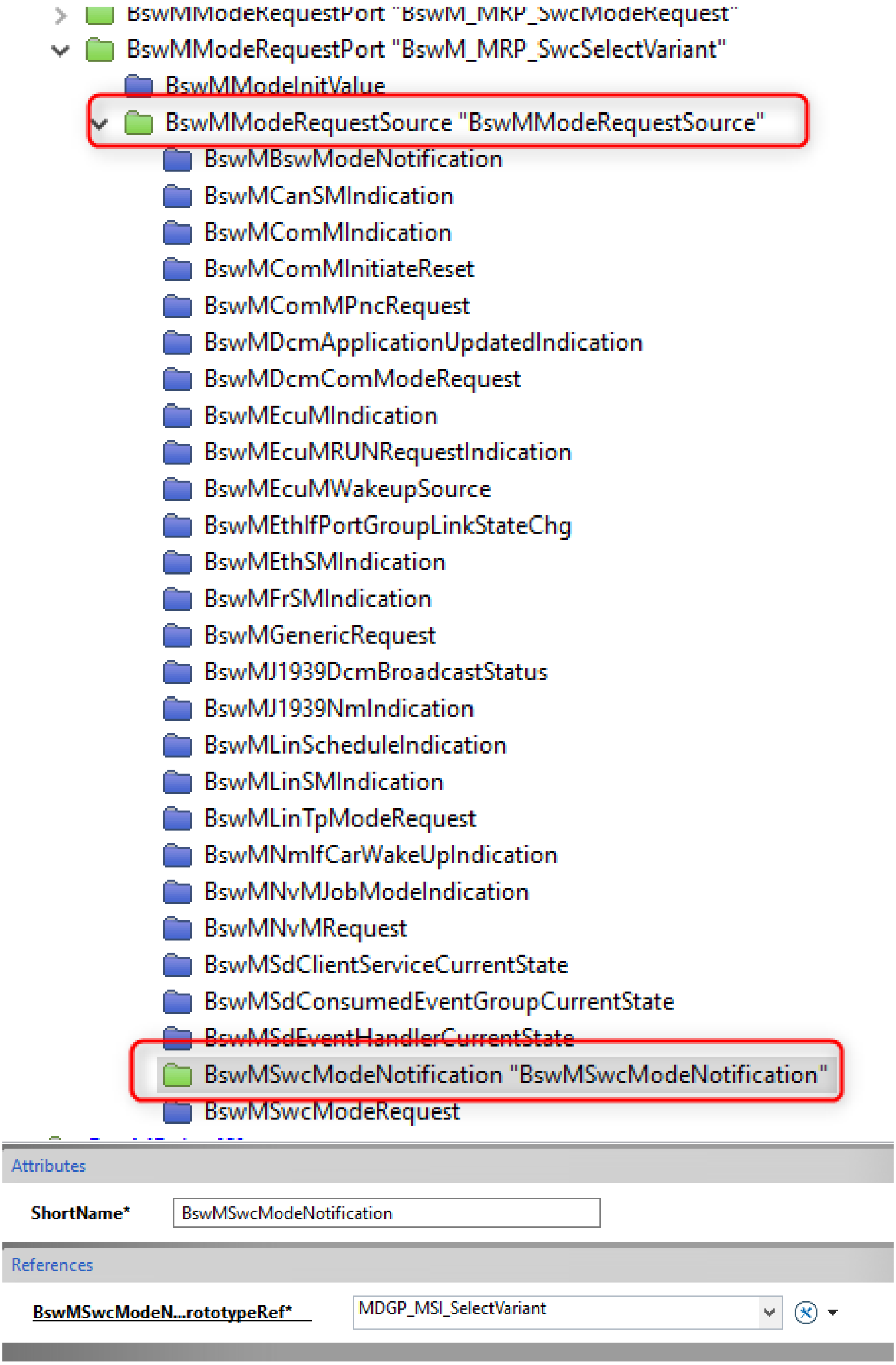


Attributes

ShortName*: BswM_MRP_SwcSelectVariant

BswMRequestProcessing*: BSWM_DEFERRED

Under the container “BswM_MRP_SwcSelectVariant”, create the child “BswMModeRequestSource” and then the “BswMSwcModeNotification”. Then configure this last one as shown in the pictures, assigning the reference to the Mode Request Group (in our case, called “MDGP_MSI_SelectVariant”):



- BswMModeRequestPort "BswM_MRP_SwcModeRequest"
- BswMModeRequestPort "BswM_MRP_SwcSelectVariant"
  - BswMModeInitValue
  - BswMModeRequestSource "BswMModeRequestSource"
    - BswMBswModeNotification
    - BswMCanSMIndication
    - BswMComMIndication
    - BswMComMInitiateReset
    - BswMComMPncRequest
    - BswMDcmApplicationUpdatedIndication
    - BswMDcmComModeRequest
    - BswMEcuMIndication
    - BswMEcuMRUNRequestIndication
    - BswMEcuMWakeupSource
    - BswMEthIfPortGroupLinkStateChg
    - BswMEthSMIndication
    - BswMFrSMIndication
    - BswMGenericRequest
    - BswMJ1939DcmBroadcastStatus
    - BswMJ1939NmIndication
    - BswMLinScheduleIndication
    - BswMLinSMIndication
    - BswMLinTpModeRequest
    - BswMNmIfCarWakeUpIndication
    - BswMNvMJobModeIndication
    - BswMNvMRequest
    - BswMSdClientServiceCurrentState
    - BswMSdConsumedEventGroupCurrentState
    - BswMSdEventHandlerCurrentState
    - BswMSwcModeNotification "BswMSwcModeNotification"
    - BswMSwcModeRequest

Attributes

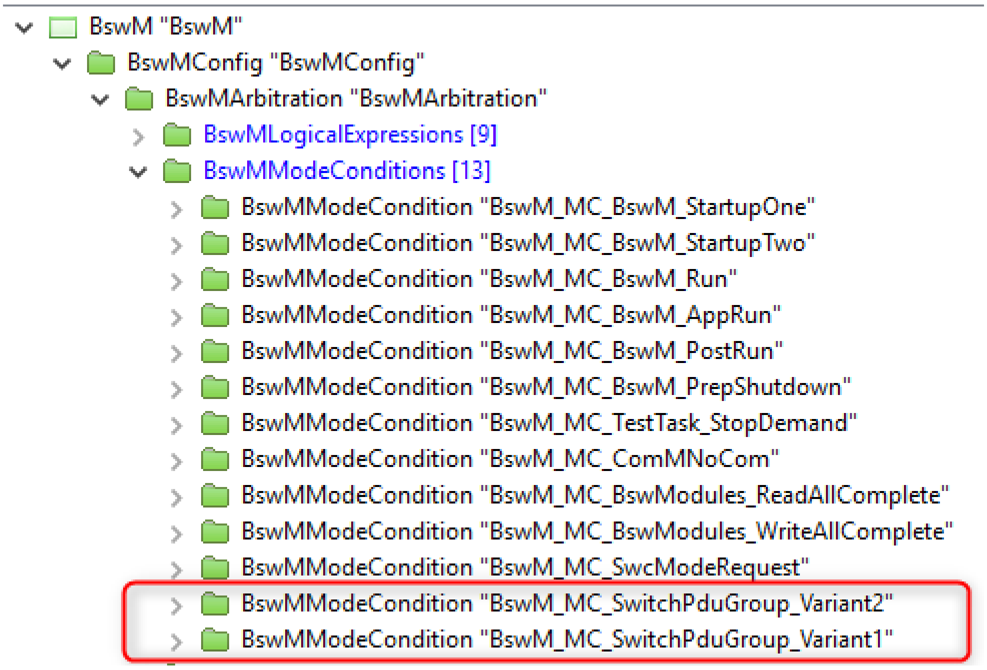
ShortName* BswMSwcModeNotification

References

BswMSwcModeN...rototypeRef* MDGP_MSI_SelectVariant

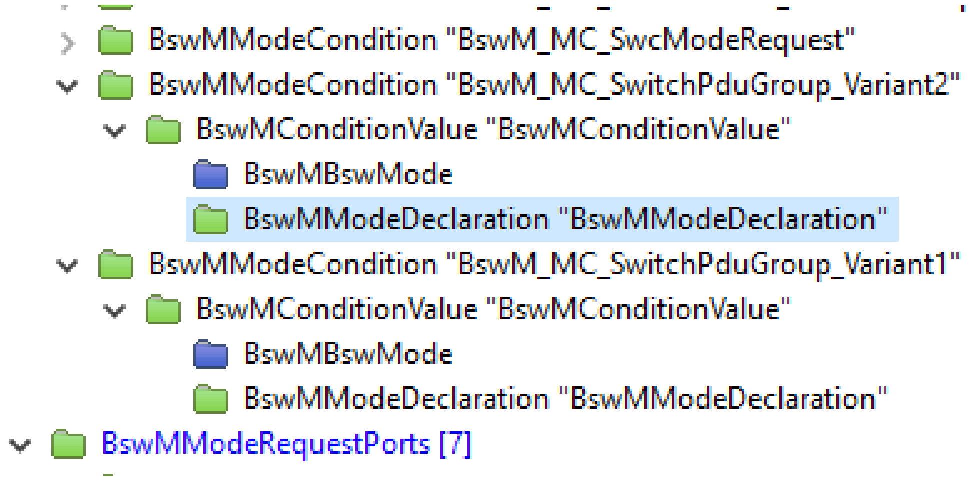
3.3.2 Create new BswMModeConditions

The second step is to configure new BswMModeConditions. Supposing to have two variants to handle, and consequently have to PDU Groups to switch on/off, you can create two new Mode Conditions.



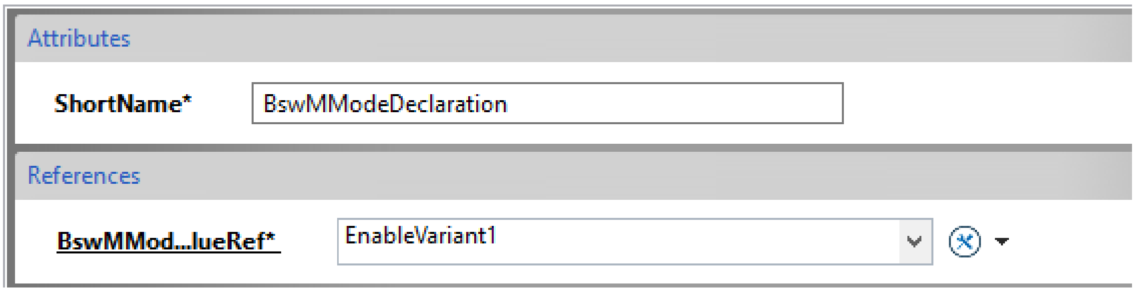
```
BswM "BswM"
  BswMConfig "BswMConfig"
    BswMArbitration "BswMArbitration"
      BswMLogicalExpressions [9]
      BswMModeConditions [13]
        BswMModeCondition "BswM_MC_BswM_StartupOne"
        BswMModeCondition "BswM_MC_BswM_StartupTwo"
        BswMModeCondition "BswM_MC_BswM_Run"
        BswMModeCondition "BswM_MC_BswM_AppRun"
        BswMModeCondition "BswM_MC_BswM_PostRun"
        BswMModeCondition "BswM_MC_BswM_PrepShutdown"
        BswMModeCondition "BswM_MC_TestTask_StopDemand"
        BswMModeCondition "BswM_MC_ComMNoCom"
        BswMModeCondition "BswM_MC_BswModules_ReadAllComplete"
        BswMModeCondition "BswM_MC_BswModules_WriteAllComplete"
        BswMModeCondition "BswM_MC_SwcModeRequest"
        BswMModeCondition "BswM_MC_SwitchPduGroup_Variant2"
        BswMModeCondition "BswM_MC_SwitchPduGroup_Variant1"
```

Configure the now the sub-container BswMModeDeclaration for each of Mode Condition:



```
BswMModeCondition "BswM_MC_SwcModeRequest"
BswMModeCondition "BswM_MC_SwitchPduGroup_Variant2"
  BswMConditionValue "BswMConditionValue"
    BswMBswMode
    BswMModeDeclaration "BswMModeDeclaration"
BswMModeCondition "BswM_MC_SwitchPduGroup_Variant1"
  BswMConditionValue "BswMConditionValue"
    BswMBswMode
    BswMModeDeclaration "BswMModeDeclaration"
BswMModeRequestPorts [7]
```

The BswMModeDeclaration shall contain the reference to the Mode value against which will be done the comparison. In one case will be “EnableVariant1”, while in the second case will be “EnableVariant2”.



Attributes

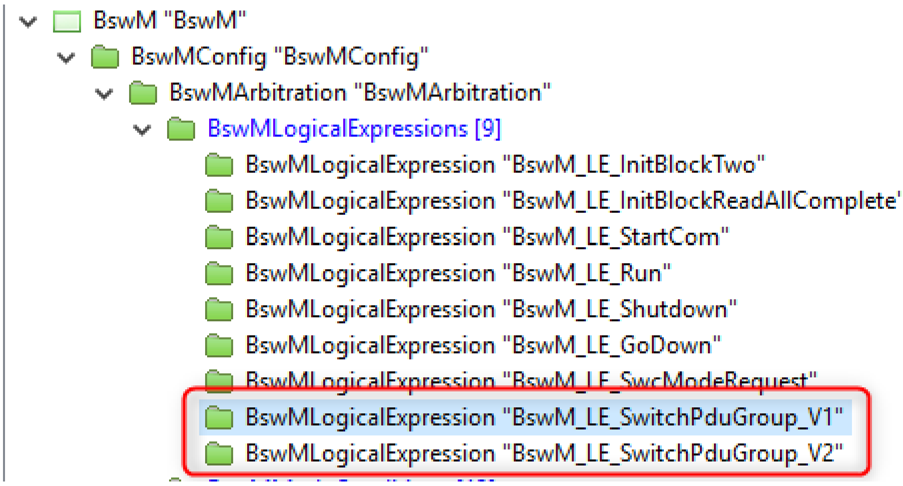
ShortName* BswMModeDeclaration

References

BswMMod...lueRef* EnableVariant1

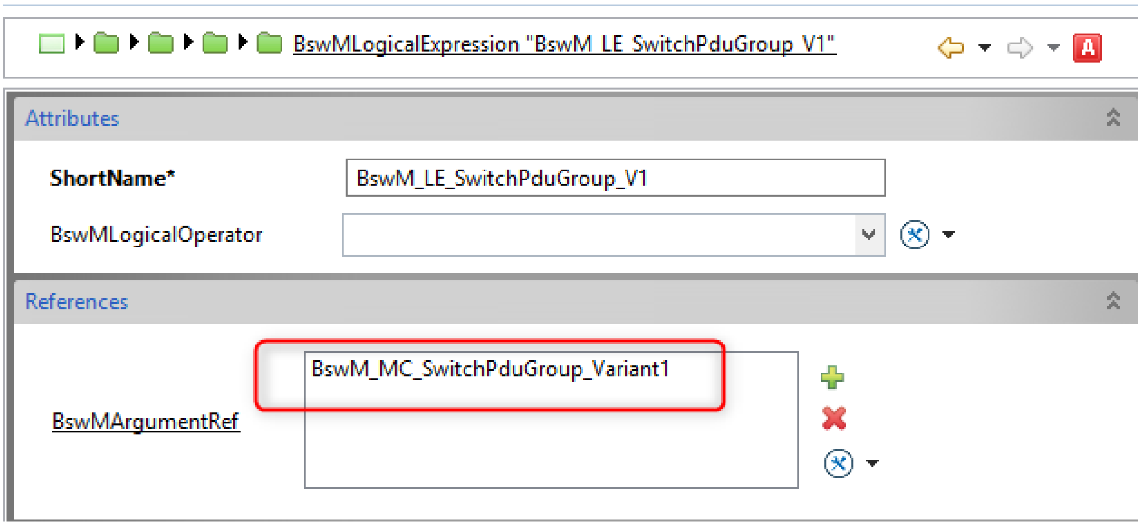
3.3.3 Configuration of new Logical Expressions

Now it is possible to add new Logical Expression. Again, two Logical Expression are needed, one for each variant we want to handle.



```
BswM "BswM"
  BswMConfig "BswMConfig"
    BswMArbitration "BswMArbitration"
      BswMLogicalExpressions [9]
        BswMLogicalExpression "BswM_LE_InitBlockTwo"
        BswMLogicalExpression "BswM_LE_InitBlockReadAllComplete'
        BswMLogicalExpression "BswM_LE_StartCom"
        BswMLogicalExpression "BswM_LE_Run"
        BswMLogicalExpression "BswM_LE_Shutdown"
        BswMLogicalExpression "BswM_LE_GoDown"
        BswMLogicalExpression "BswM_LE_SwcModeRequest"
        BswMLogicalExpression "BswM_LE_SwitchPduGroup_V1"
        BswMLogicalExpression "BswM_LE_SwitchPduGroup_V2"
```

The Logical expression may contain a “dummy” check, therefore the BswMLogicalOperator is not mandatory to be configured. Instead, you shall add the reference to the BswM MC you created before. For example, for the Logical Expression related to the Variant1, the configuration will be the following:



BswMLogicalExpression "BswM_LE_SwitchPduGroup_V1"

Attributes

ShortName* BswM_LE_SwitchPduGroup_V1

BswMLogicalOperator

References

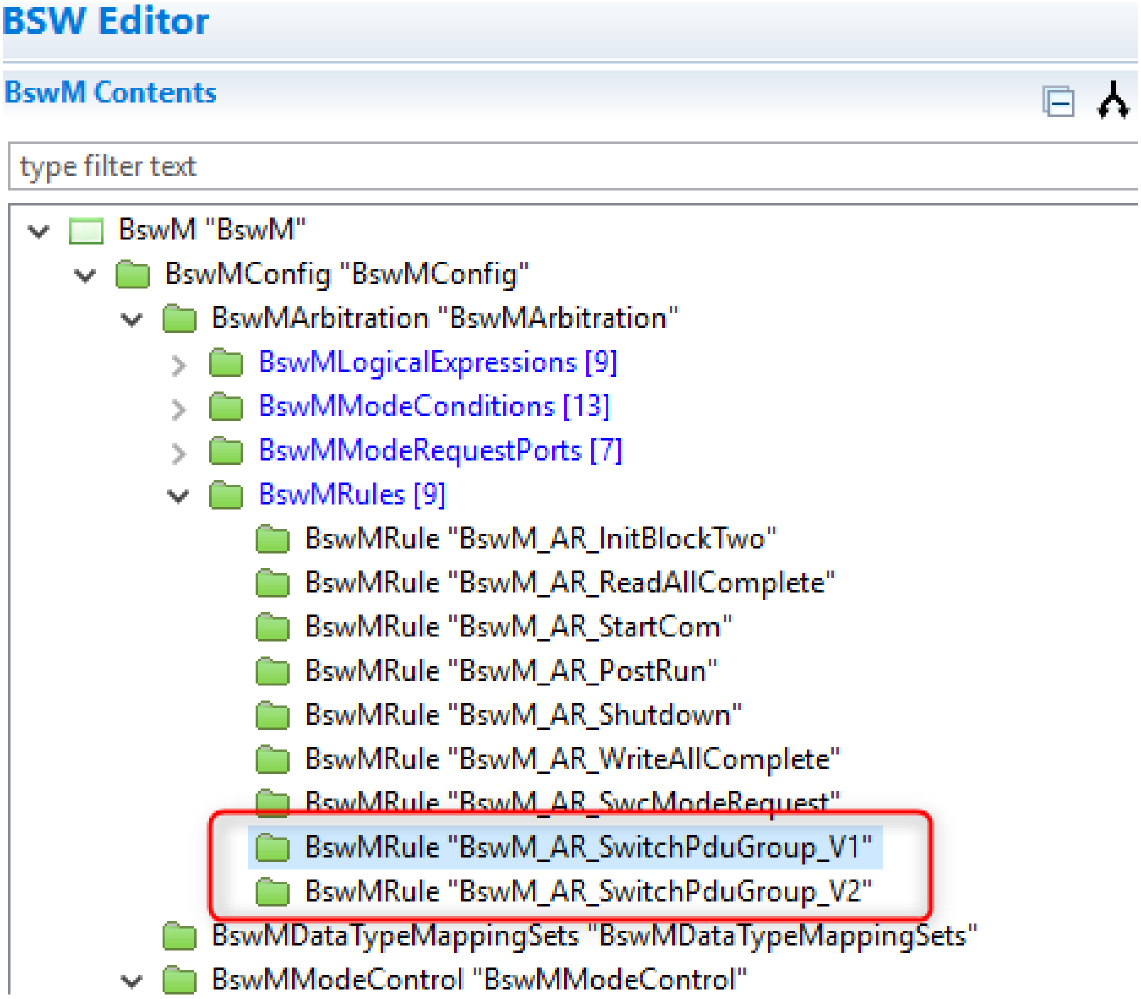
BswMArgumentRef BswM_MC_SwitchPduGroup_Variant1

The Logical Expression for Variant2 will be similarly configured with the reference to the MC of the second Variant.

3.3.4 Creation of the new Action Rules

Now it is possible to configure the Action Rules associated to the request to switch between the two variants.

You can add two AR, one for each variant to be handled, as shown in the picture below.



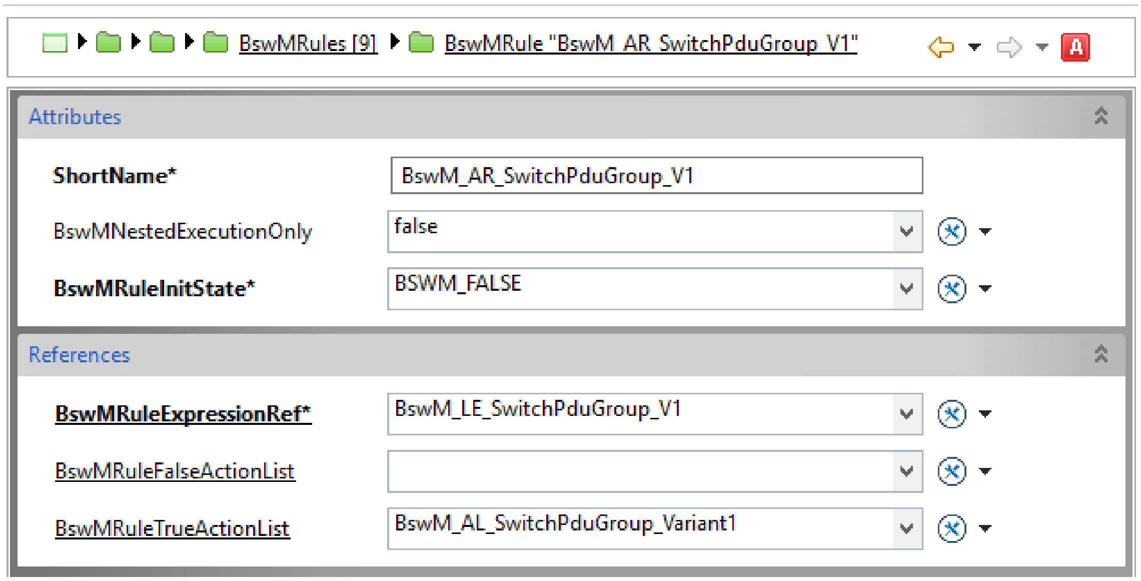
BSW Editor

BswM Contents

type filter text

```
BswM "BswM"
  BswMConfig "BswMConfig"
    BswMArbitration "BswMArbitration"
      BswMLogicalExpressions [9]
      BswMModeConditions [13]
      BswMModeRequestPorts [7]
      BswMRules [9]
        BswMRule "BswM_AR_InitBlockTwo"
        BswMRule "BswM_AR_ReadAllComplete"
        BswMRule "BswM_AR_StartCom"
        BswMRule "BswM_AR_PostRun"
        BswMRule "BswM_AR_Shutdown"
        BswMRule "BswM_AR_WriteAllComplete"
        BswMRule "BswM_AR_SwcModeRequest"
        BswMRule "BswM_AR_SwitchPduGroup_V1"
        BswMRule "BswM_AR_SwitchPduGroup_V2"
    BswMDataTypeMappingSets "BswMDataTypeMappingSets"
    BswMModeControl "BswMModeControl"
```

More in detail, the configuration of each AR will be as shown in the pictures below.



BswMRules [9] ▶ BswMRule "BswM_AR_SwitchPduGroup_V1"

Attributes

ShortName*: BswM_AR_SwitchPduGroup_V1

BswMNestedExecutionOnly: false

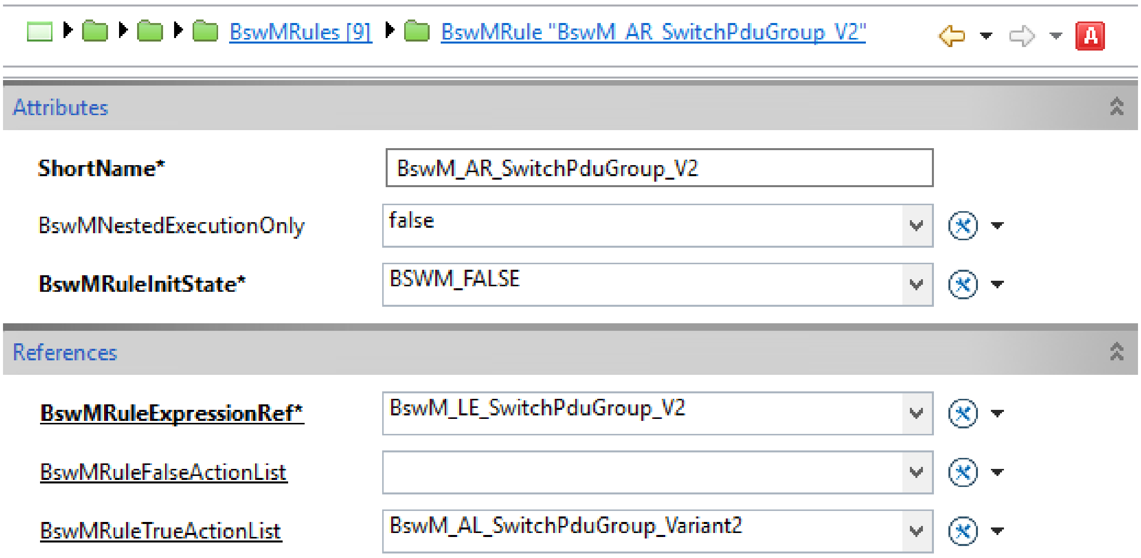
BswMRuleInitState*: BSWM_FALSE

References

BswMRuleExpressionRef*: BswM_LE_SwitchPduGroup_V1

BswMRuleFalseActionList:

BswMRuleTrueActionList: BswM_AL_SwitchPduGroup_Variant1



BswMRules [9] ▶ BswMRule "BswM_AR_SwitchPduGroup_V2"

Attributes

ShortName*: BswM_AR_SwitchPduGroup_V2

BswMNestedExecutionOnly: false

BswMRuleInitState*: BSWM_FALSE

References

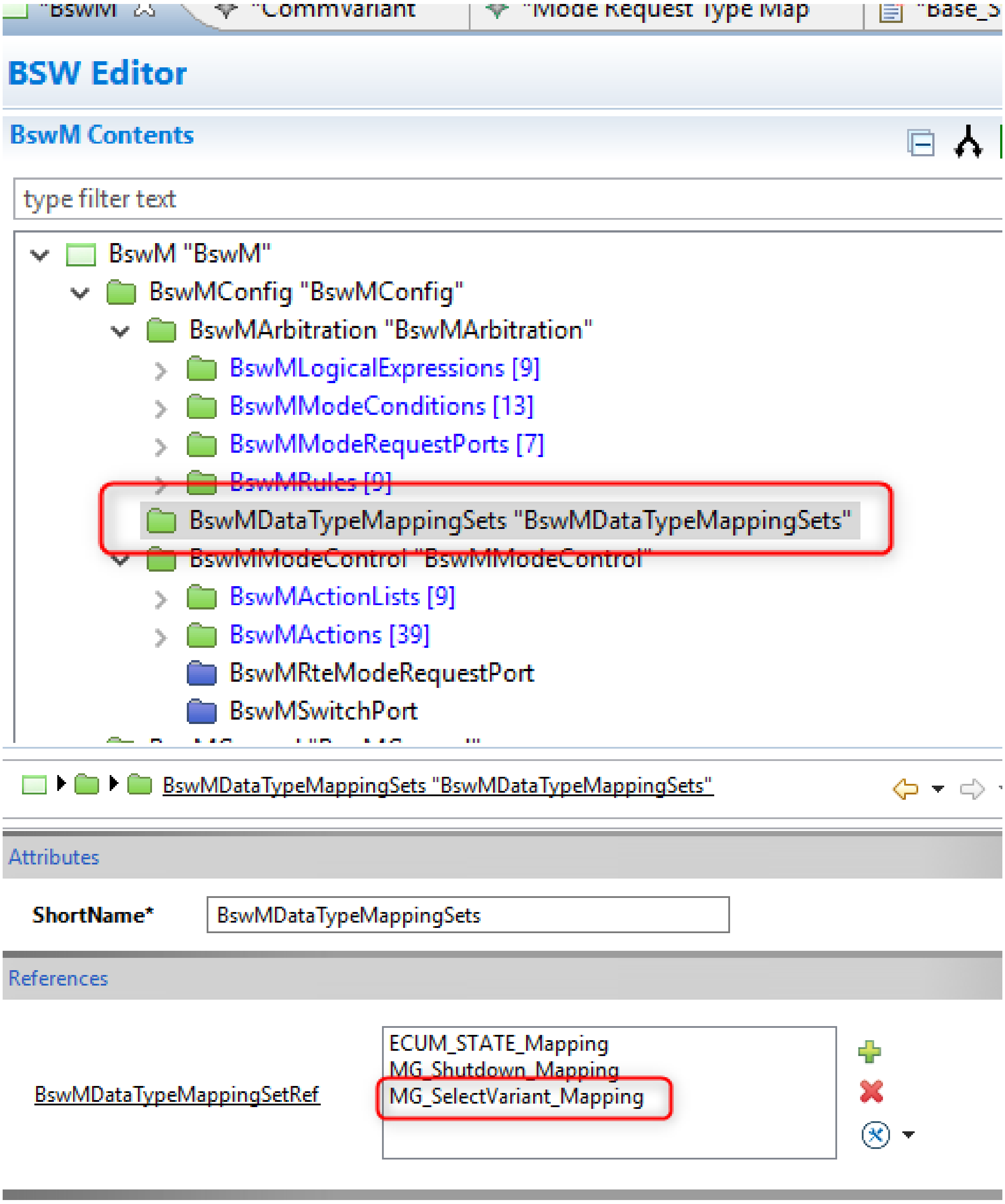
BswMRuleExpressionRef*: BswM_LE_SwitchPduGroup_V2

BswMRuleFalseActionList:

BswMRuleTrueActionList: BswM_AL_SwitchPduGroup_Variant2

3.3.5 Add the Mapping set reference

The BswM needs the reference to the Data Mapping Set, previously created in the ASW.



- BswM "BswM"
  - BswMConfig "BswMConfig"
    - BswMArbitration "BswMArbitration"
      - BswMLogicalExpressions [9]
      - BswMModeConditions [13]
      - BswMModeRequestPorts [7]
      - BswMRules [9]
    - BswMDataTypeMappingSets "BswMDataTypeMappingSets"
    - BswMModeControl "BswMModeControl"
      - BswMActionLists [9]
      - BswMActions [39]
      - BswMRteModeRequestPort
      - BswMSwitchPort

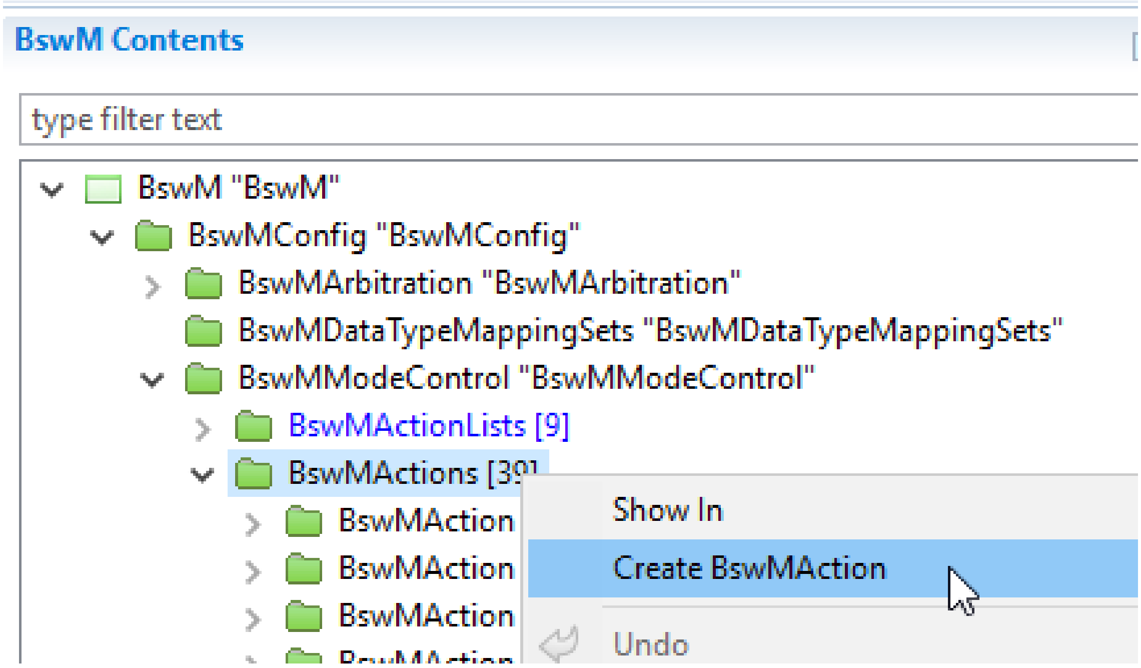
BswMDataTypeMappingSets "BswMDataTypeMappingSets"

ShortName* BswMDataTypeMappingSets

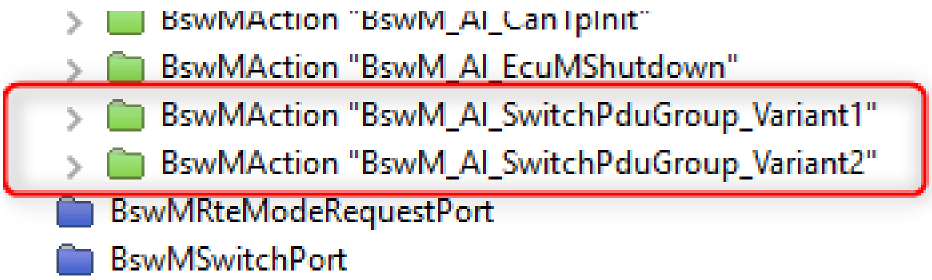
BswMDataTypeMappingSetRef
ECUM_STATE_Mapping
MG_Shutdown_Mapping
MG_SelectVariant_Mapping

3.3.6 Create new BswMActions

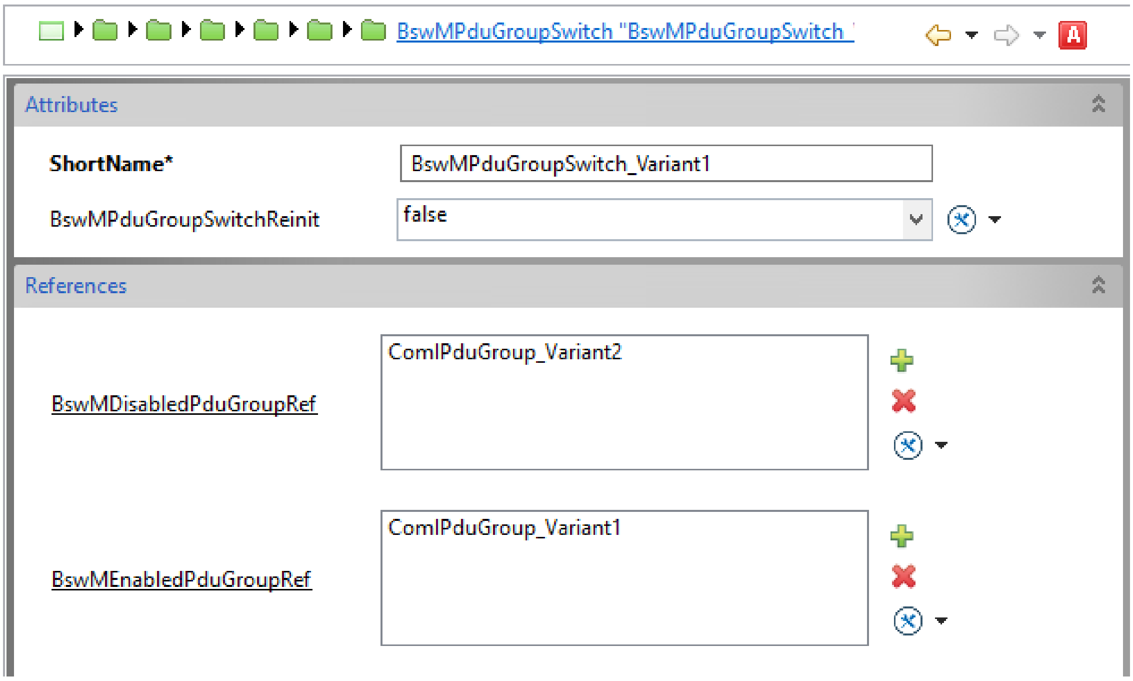
You can now add new BswM Action Items (AI), which will be executed when the associated Action List (AL) is triggered (see next sections for the reference on the AL creation).



It is needed to create two new AI, one for each variant:


BswMAction "BswM_AI_CanTpInit"
BswMAction "BswM_AI_EcuMShutdown"
BswMAction "BswM_AI_SwitchPduGroup_Variant1"
BswMAction "BswM_AI_SwitchPduGroup_Variant2"
BswMRteModeRequestPort
BswMSwitchPort

For each AI, the correct action to be selected for our purpose is “BswMPduGroupSwitch”. For each variant, you can then configure which PDU groups shall be disabled and which ones shall be enabled, as shown in the picture below. The AI for the Variant 2 shall be configured similarly.



BswMPduGroupSwitch "BswMPduGroupSwitch

Attributes

ShortName* BswMPduGroupSwitch_Variant1

BswMPduGroupSwitchReinit false

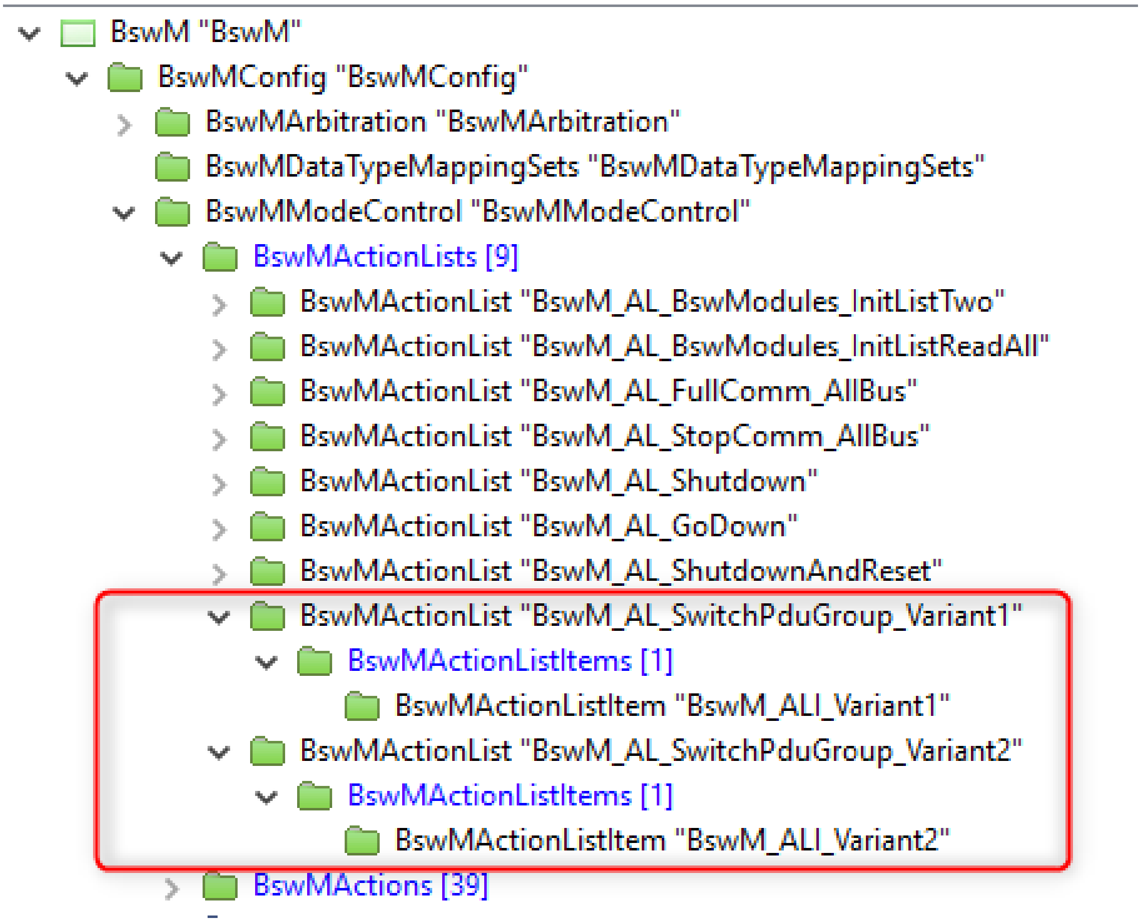
References

BswMDisabledPduGroupRef ComIPduGroup_Variant2

BswMEnabledPduGroupRef ComIPduGroup_Variant1

3.3.7 Creation of the new Action Lists

The last step for the BswM configuration is to create the new Action Lists needed. Each of them shall contain the needed Action Items:



```
BswM "BswM"
  BswMConfig "BswMConfig"
    BswMArbitration "BswMArbitration"
    BswMDataTypeMappingSets "BswMDataTypeMappingSets"
    BswMModeControl "BswMModeControl"
      BswMActionLists [9]
        BswMActionList "BswM_AL_BswModules_InitListTwo"
        BswMActionList "BswM_AL_BswModules_InitListReadAll"
        BswMActionList "BswM_AL_FullComm_AllBus"
        BswMActionList "BswM_AL_StopComm_AllBus"
        BswMActionList "BswM_AL_Shutdown"
        BswMActionList "BswM_AL_GoDown"
        BswMActionList "BswM_AL_ShutdownAndReset"
        BswMActionList "BswM_AL_SwitchPduGroup_Variant1"
          BswMActionListItems [1]
            BswMActionListItem "BswM_ALI_Variant1"
        BswMActionList "BswM_AL_SwitchPduGroup_Variant2"
          BswMActionListItems [1]
            BswMActionListItem "BswM_ALI_Variant2"
      BswMActions [39]
```

3.3.8 BSW Generation and BswM Service SW-C

Now the BswM configuration is complete and it is possible to generate the BSW. After the generation, you can notice that also the BswM Service SW-C has been updated with new interfaces to handle the Mode Switch Request coming from another SW-C. The next sections contains details on how to integrate the ASW with the changes done in the BswM.

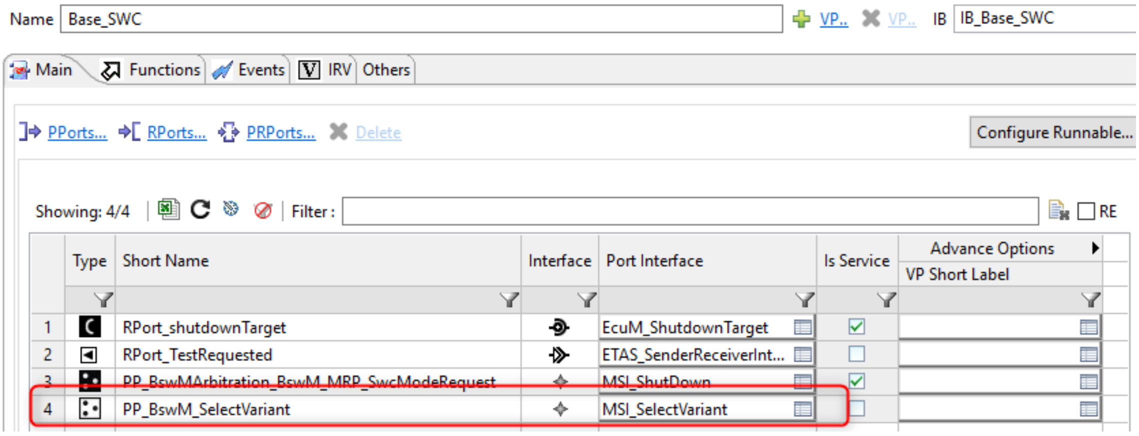
3.4 Application Software - Integration with the BswM

Now it is needed to modify the ASW to communicate with the BswM and request for Mode Switch.

3.4.1 Modification to the SW-C

In our example, we decided that Base_SWC will be the SWC in charge to request the Mode Switch to the BswM. For this reason, we need to add to it the needed interface for mode management.

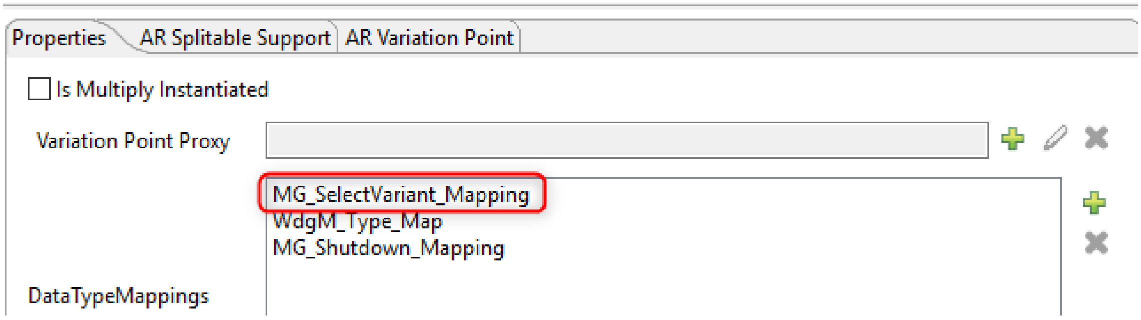
Add the ProvidePort selecting MSI_SelectVariant as Port Interface.



Name: Base_SWC   VP..   VP..   IB: IB_Base_SWC

	Type	Short Name	Interface	Port Interface	Is Service	Advance Options
						VP Short Label
1		RPort_shutdownTarget		EcuM_ShutdownTarget	☑	
2		RPort_TestRequested		ETAS_SenderReceiverInt...	□	
3		PP_BswMArbitration_BswM_MRP_SwcModeRequest		MSI_ShutDown	☑	
4		PP_BswM_SelectVariant		MSI_SelectVariant	□	

Add the DataMappingSet related to the SelectVariant:

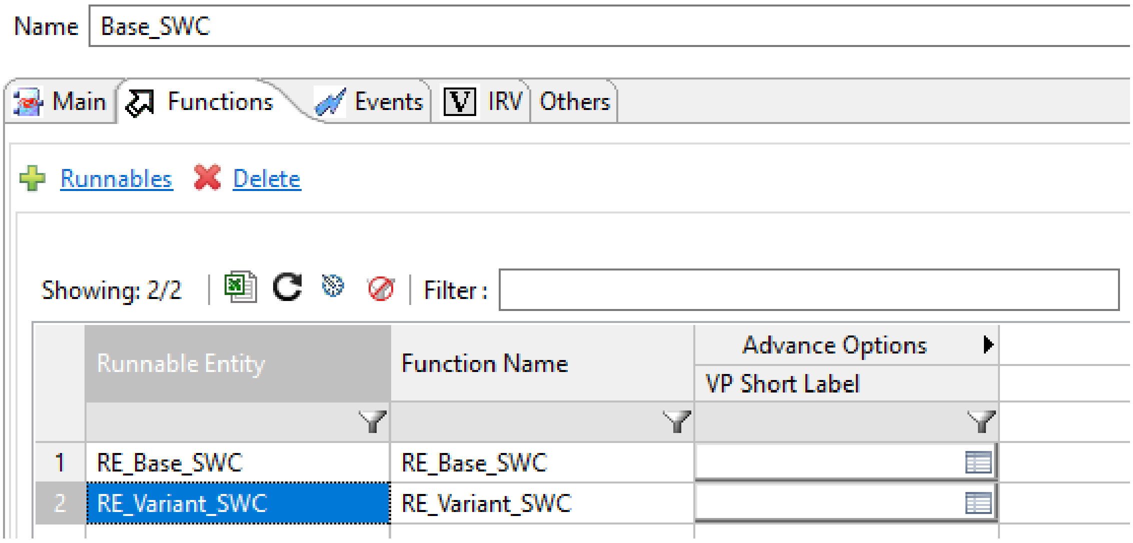


□ Is Multiply Instantiated

Variation Point Proxy:

DataTypeMappings:
MG_SelectVariant_Mapping
WdgM_Type_Map
MG_Shutdown_Mapping

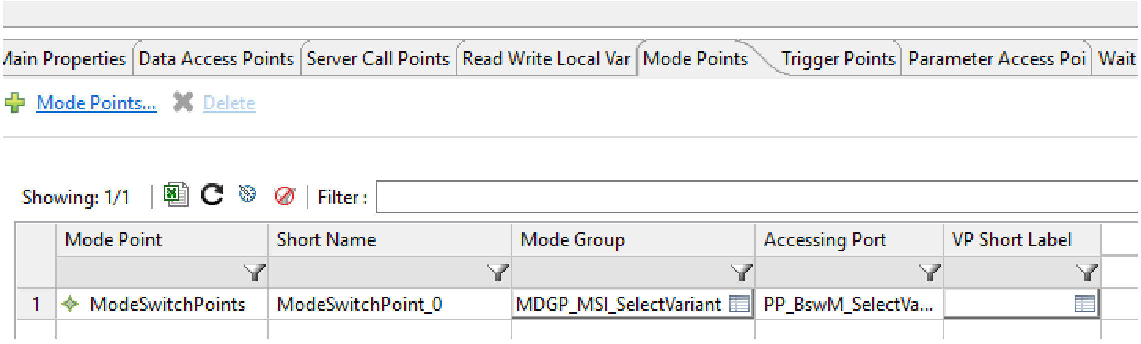
Add a new Runnable:



Name: Base_SWC

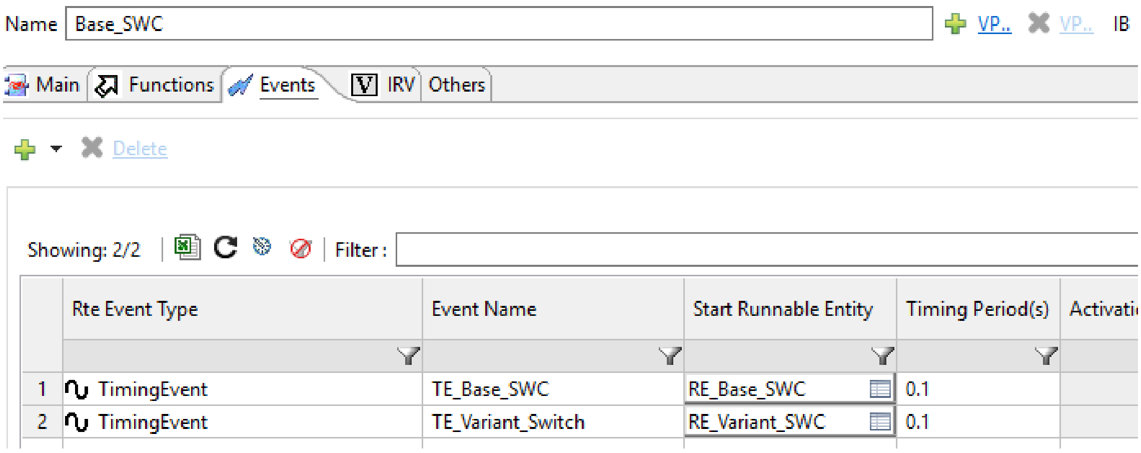
	Runnable Entity	Function Name	Advance Options
			VP Short Label
1	RE_Base_SWC	RE_Base_SWC	
2	RE_Variant_SWC	RE_Variant_SWC	

And associate a Mode Point to the created Runnable Entity:


Main Properties | Data Access Points | Server Call Points | Read Write Local Var | Mode Points | Trigger Points | Parameter Access Poi | Wait
Mode Points... Delete
Showing: 1/1
Filter :

	Mode Point	Short Name	Mode Group	Accessing Port	VP Short Label
1	ModeSwitchPoints	ModeSwitchPoint_0	MDGP_MSI_SelectVariant	PP_BswM_SelectVa...	

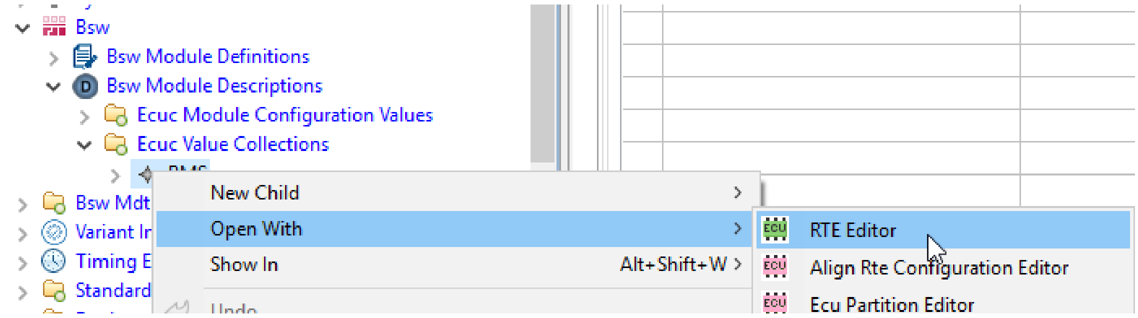
Add a new timing event for the RE:



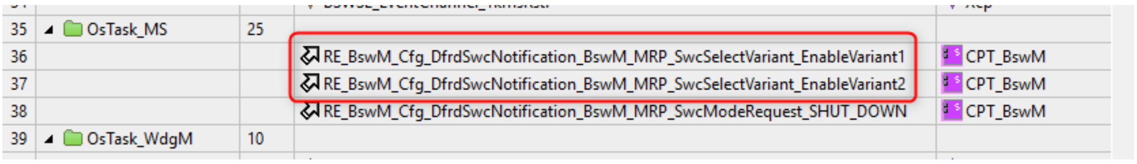
Name: Base_SWC

	Rte Event Type	Event Name	Start Runnable Entity	Timing Period(s)	Activati
1	TimingEvent	TE_Base_SWC	RE_Base_SWC	0.1	
2	TimingEvent	TE_Variant_Switch	RE_Variant_SWC	0.1	

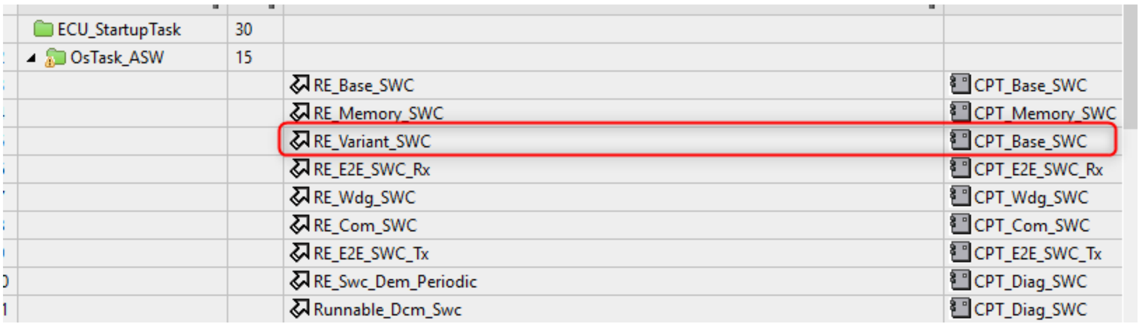
Using the AR Explorer, open the Ecuc Value Collections with the RTE Editor:



Finally, the Runnable Entities generated for the BswM shall be mapped to the Os Task, as well as the new Runnable Entity created for the SW-C.



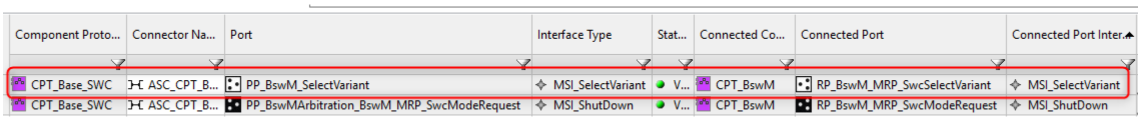
35	OsTask_MS	25		
36			RE_BswM_Cfg_DfrdSwcNotification_BswM_MRP_SwcSelectVariant_EnableVariant1	CPT_BswM
37			RE_BswM_Cfg_DfrdSwcNotification_BswM_MRP_SwcSelectVariant_EnableVariant2	CPT_BswM
38			RE_BswM_Cfg_DfrdSwcNotification_BswM_MRP_SwcModeRequest_SHUT_DOWN	CPT_BswM
39	OsTask_WdgM	10		



ECU_StartupTask	30		
OsTask_ASW	15		
		RE_Base_SWC	CPT_Base_SWC
		RE_Memory_SWC	CPT_Memory_SWC
		RE_Variant_SWC	CPT_Base_SWC
		RE_E2E_SWC_Rx	CPT_E2E_SWC_Rx
		RE_Wdg_SWC	CPT_Wdg_SWC
		RE_Com_SWC	CPT_Com_SWC
		RE_E2E_SWC_Tx	CPT_E2E_SWC_Tx
		RE_Swc_Dem_Periodic	CPT_Diag_SWC
		Runnable_Dcm_Swc	CPT_Diag_SWC

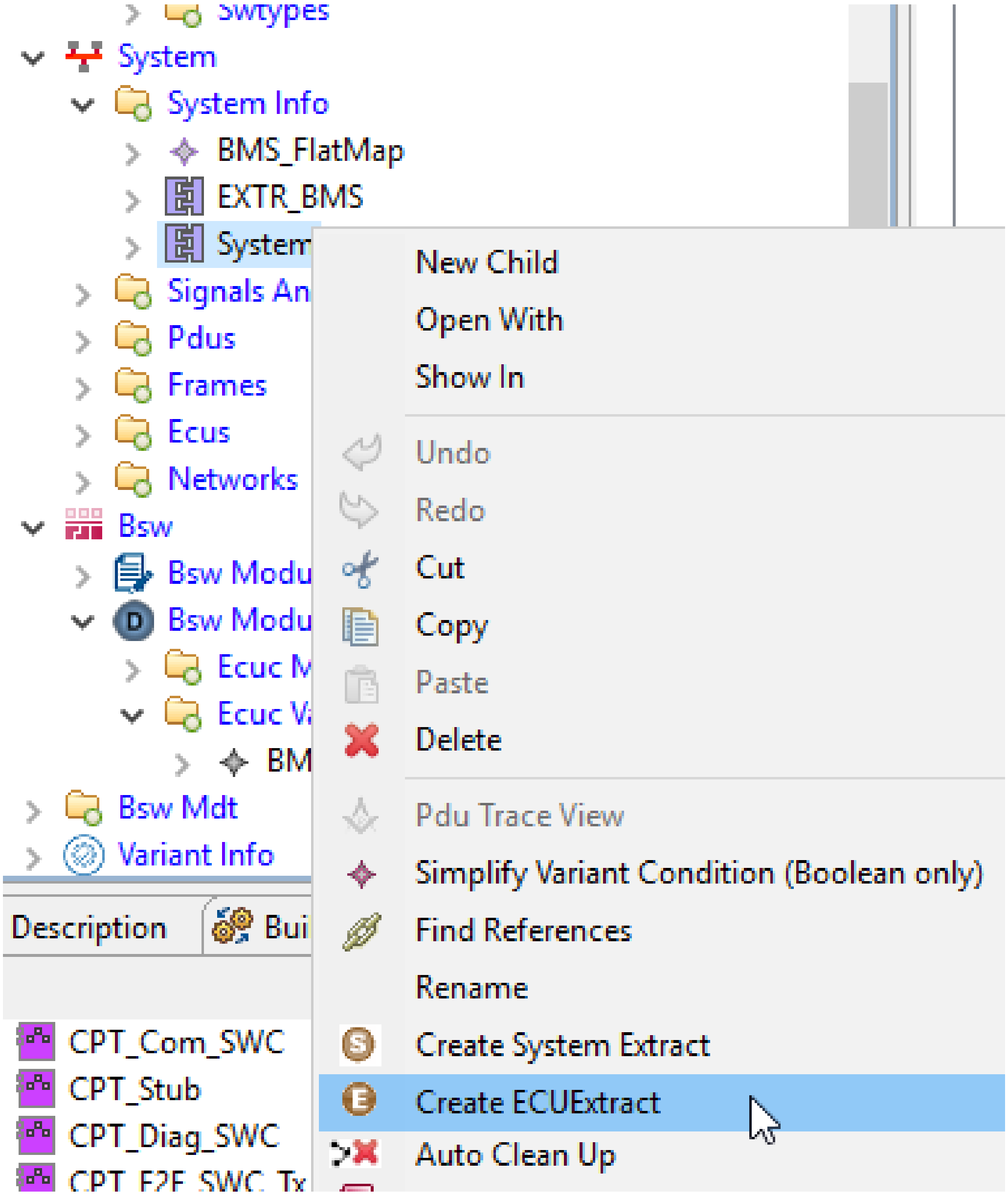
3.4.2 Connections on the Top Level Composition

The last step for the configuration is to connect the Base_SWC with the BswM Service SW Component. Open the TopLevelComposition with the Composition Editor, switch to the tab “Manual Connection Editor” and create the new connection as below:

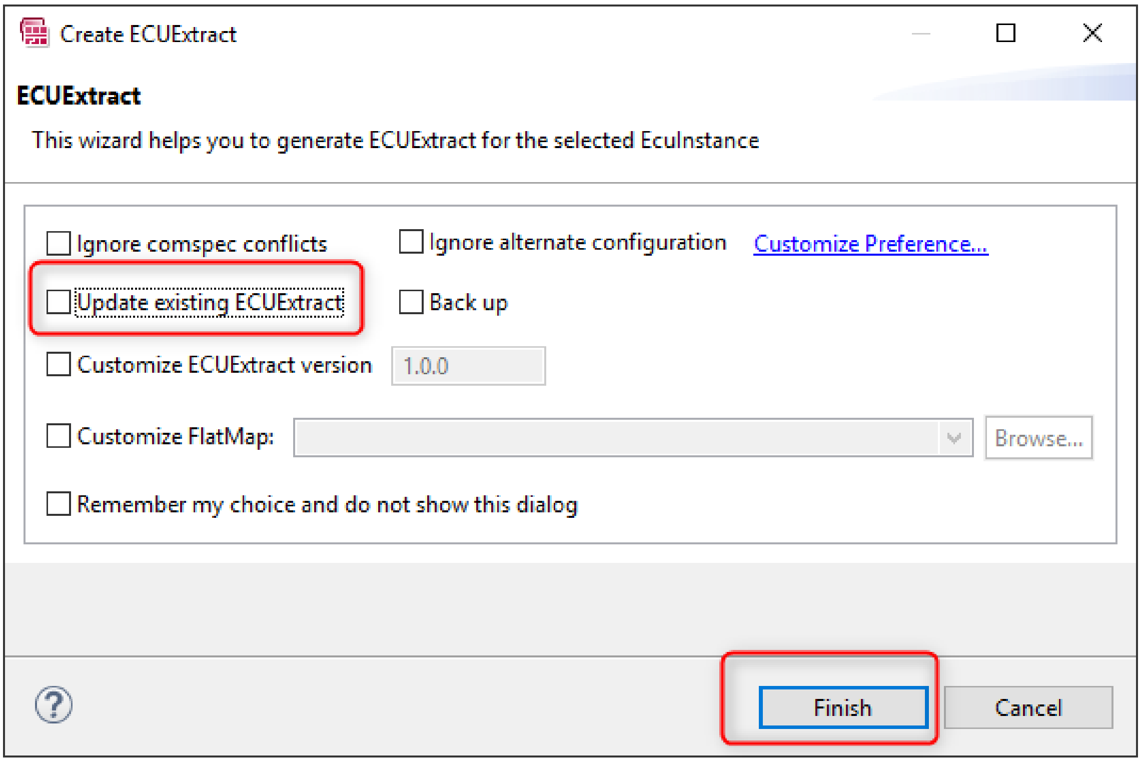


Component Proto...	Connector Na...	Port	Interface Type	Stat...	Connected Co...	Connected Port	Connected Port Inter...
CPT_Base_SWC	ASC_CPT_B...	PP_BswM_SelectVariant	MSI_SelectVariant	● V...	CPT_BswM	RP_BswM_MRP_SwcSelectVariant	MSI_SelectVariant
CPT_Base_SWC	ASC_CPT_B...	PP_BswMArbitration_BswM_MRP_SwcModeRequest	MSI_ShutDown	● V...	CPT_BswM	RP_BswM_MRP_SwcModeRequest	MSI_ShutDown

Now the Composition is updated and it is possible to re-generate the ECU Extract. To do so, right click on the System and then on “Create ECUExtract”.



Untick the box “Update existing ECUExtract” and then the Finish button.



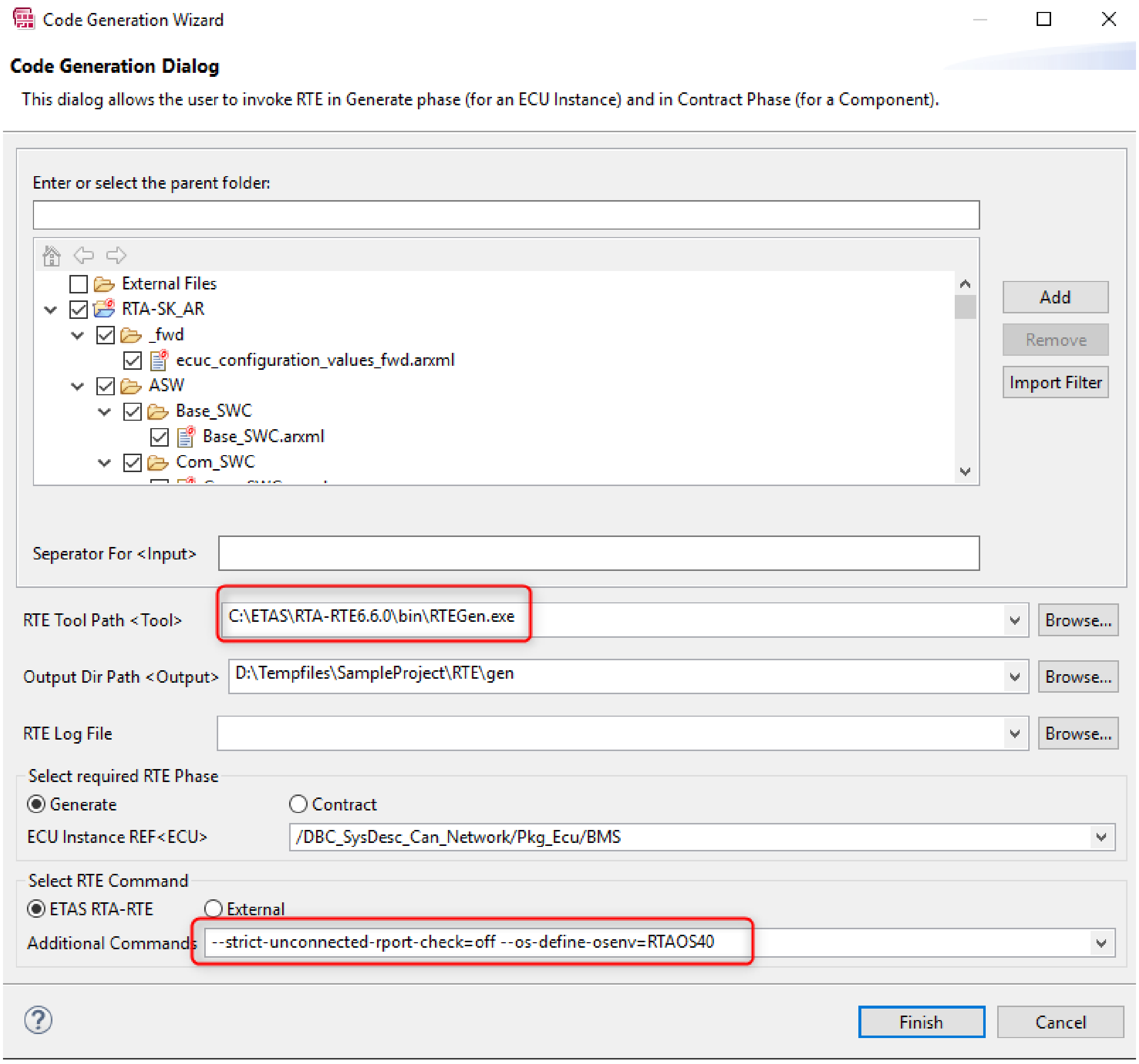
Create ECUExtract

ECUExtract

This wizard helps you to generate ECUExtract for the selected EcuInstance

☐ Ignore comspec conflicts
☐ Ignore alternate configuration
☐ Update existing ECUExtract
☐ Back up
☐ Customize ECUExtract version: 1.0.0
☐ Customize FlatMap:
☐ Remember my choice and do not show this dialog

Finally, generate the RTE with the options shown in the picture below.



Code Generation Wizard

Code Generation Dialog

This dialog allows the user to invoke RTE in Generate phase (for an ECU Instance) and in Contract Phase (for a Component).

Enter or select the parent folder:

- □ External Files
- ☑ RTA-SK_AR
  - ☑ _fwd
    - ☑ ecuc_configuration_values_fwd.arxml
  - ☑ ASW
    - ☑ Base_SWC
      - ☑ Base_SWC.arxml
    - ☑ Com_SWC

Seperator For <Input>

RTE Tool Path <Tool> C:\ETAS\RTA-RTE6.6.0\bin\RTEGen.exe

Output Dir Path <Output> D:\Tempfiles\SampleProject\RTE\gen

RTE Log File

Select required RTE Phase

● Generate ○ Contract

ECU Instance REF<ECU> /DBC_SysDesc_Can_Network/Pkg_Ecu/BMS

Select RTE Command

● ETAS RTA-RTE ○ External

Additional Commands --strict-unconnected-rport-check=off --os-define-osenv=RTAOS40

3.4.3 C Code for the Base_SWC

Now that the RTE has been generated and all the configuration has been updated, we can add some test code in the new Runnable created for the Base_SWC.

The aim is to test the switch between the two variants, therefore we can simply use a state variable into the runnable.

```
#define ENABLE_VARIANT_1    0u
#define ENABLE_VARIANT_2    1u

uint8 variant_state = 0u;

FUNC(void, BswM_Test_CODE) RE_Variant_SWC(void)
{
    if (variant_state == 0)
    {
        Rte_Switch_Base_SWC_PP_BswM_SelectVariant_MDGP_MSI_SelectVariant(ENABLE_VARIANT_1);
    }
    else
    {
        Rte_Switch_Base_SWC_PP_BswM_SelectVariant_MDGP_MSI_SelectVariant(ENABLE_VARIANT_2);
    }
}
```

During the testing, we can force the variable to a specific value to trigger the switch between the two Variants. Monitoring the CAN communication with the PC, you will be able to see the ECU stopping/starting the IPDU Groups according to the selected variant.

4 Conclusion

The samples and advice provided with this AN are provided to inform and guide and should not be assumed to be universally applicable. It is your responsibility to interpret the content and apply it to your specific use case.

5 Contact, Support and Problem Reporting

For details of your local sales office as well as your local technical support team and product hotlines, take a look at the ETAS website:

ETAS subsidiaries	www.etas.com/en/contact.php
ETAS technical support	www.etas.com/en/hotlines.php