APPLICATION NOTE

IEC 61850 Mod/Beh

APN-101, Edition 01

SIEMENS

SIEMENS

SIPROTEC 5
IEC 61850 Mod/Beh

V9.70 and higher

Application Notes

01

Table of Contents

NOTE

For your own safety, observe the warnings and safety instructions contained in this document, if available.

Document version: 01.00

Edition: 12.2023

Version of the product described: V9.70 and higher

Table of Contents

1 Introduction

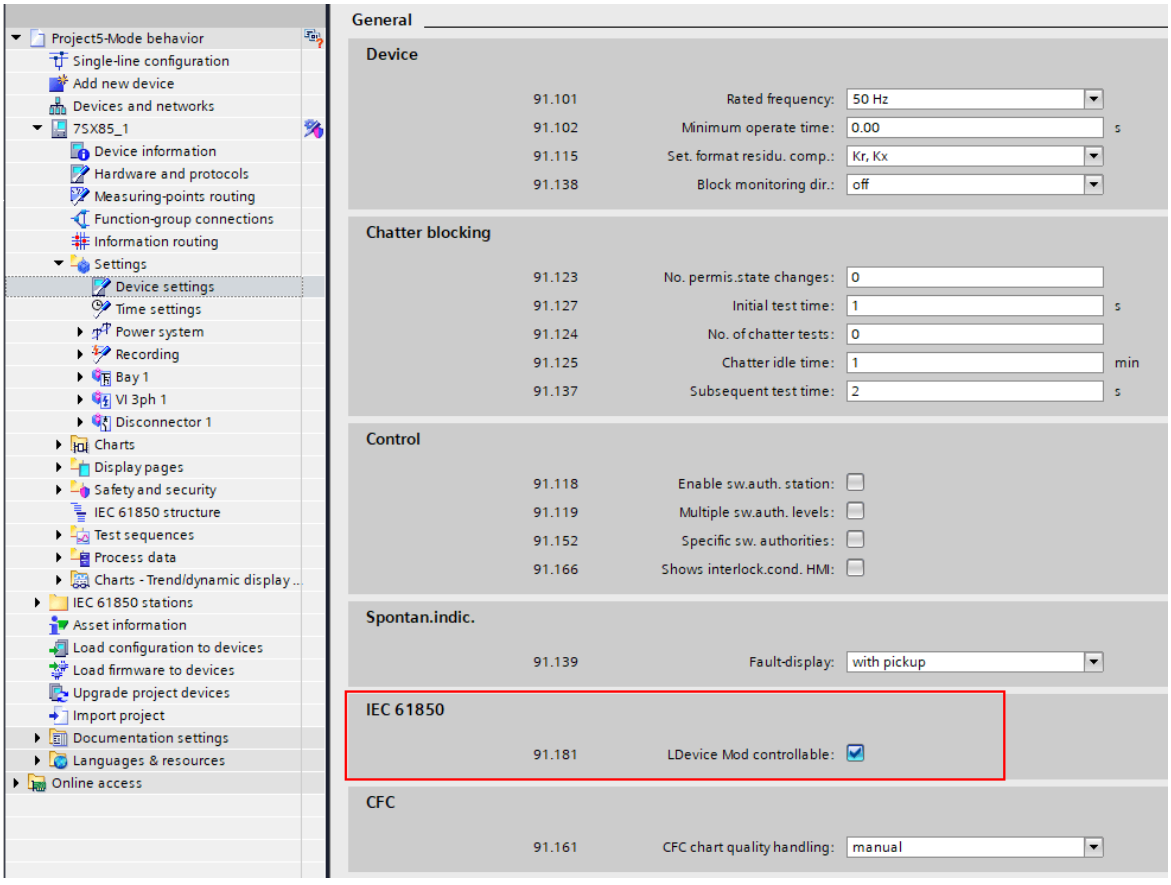
The Mod (mode) of a Logic Device can be controlled by the Logical Node Zero **LLN0** of this Logical Device. This chapter describes how to enable this feature for every Logical Device.

1.1 Mode/Behavior Compliant with IEC 61850

The mode of a Logical Device (as well as the child Logical Devices and Logical Nodes) can be controlled by the Logical Node Zero **LLN0** of this Logical Device. You can set the mode of a specific Logical Device (represented as a bay, a native or user-defined function group, or a native or user-defined function), for example, for the test purpose.

To use this feature, proceed as follows:

✧ Activate the option **`LDevice Mod controllable`**[1] in DIGSI under **Settings** > **Device settings**, as shown in the following figure.



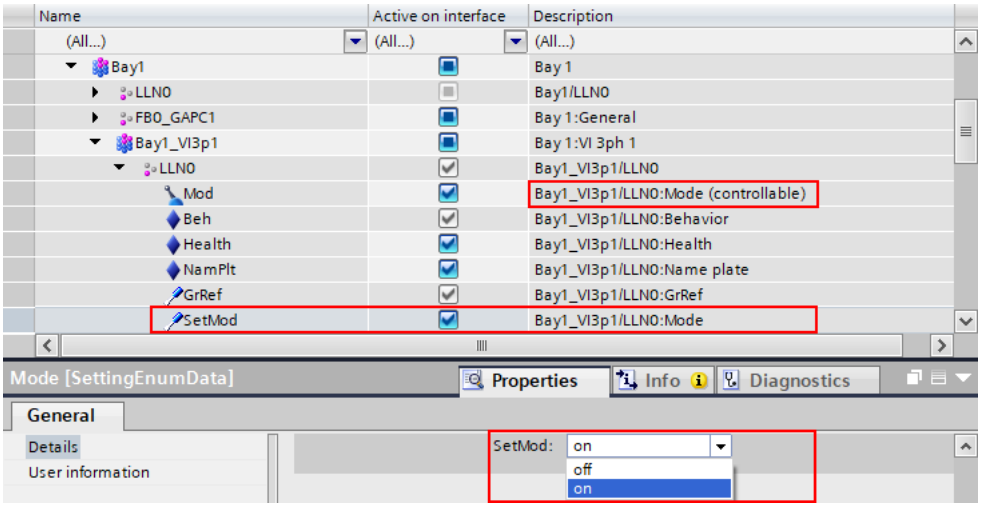
[sc_Device_setting_IEC61850_2, 1, en_US]

Figure 1-1 Activating the option **`LDevice Mod controllable`**

Then the **`Mod`** of **LLN0** is controllable, as shown in *Figure 1-2*.

- The **`Mod`** of **LLN0** is displayed as **controllable** (which was **status only** in versions earlier than V9.70) in the Editor **IEC 61850 structure**. You can change the mode of each Logical Device via the controllable **`Mod`** of **LLN0**, for example, using the IEC Browser.
- The parameter **`SetMod`** is also available for changing the mode of each LLN0. The allowed values of this parameter are ***`on`*** and ***`off`***, which means that you can enable or disable a Logical Device (as well as the child Logical Devices and Logical Nodes). This parameter belongs to the settings group and can therefore have a settings-group dependent value.

1 This parameter is only applicable if IEC 61850 is active.



[sc_Mode controllable setting, 1, en_US]

Figure 1-2 Controllable **Mod** of LLN0

The behavior of a Logical Device or Logical Node results from the combination of all mode sources, in compliance with IEC 61850. For details, refer to chapter *2.3 Function Control* in the *SIPROTEC 5 device manuals*.

After flexible engineering, the behaviors of the Logical Devices and Logical Nodes also comply with IEC 61850.

For the sampled measured values (SMVs) with the test bit transmitted via the process-bus interface, the SMV processing also complies with IEC 61850.

The following chapters describe some use cases in testing and in flexible engineering where the behaviors comply with IEC 61850.

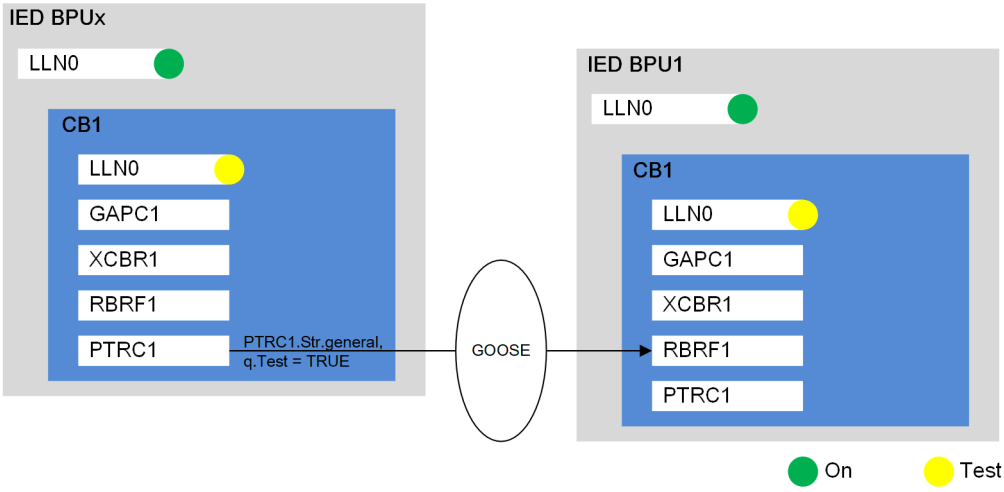
2 Behaviors during Test

This chapter describes how the device processes the incoming samples with the test quality (**q.Test** = ***TRUE***) according to the definition of Mod/Beh.

2.1 Behavior during Test of GOOSE Connections

GOOSE connections have a dedicated quality processing configuration for the signals that are received by a function. When a test mismatch occurs (the incoming signal indicates **q.Test** = ***TRUE***, while the receiver has the behavior (**Beh**) of either ***On*** or ***Relay blocked***), the configured quality processing scenario applies.

Use Case: Testing the Circuit-Breaker Failure

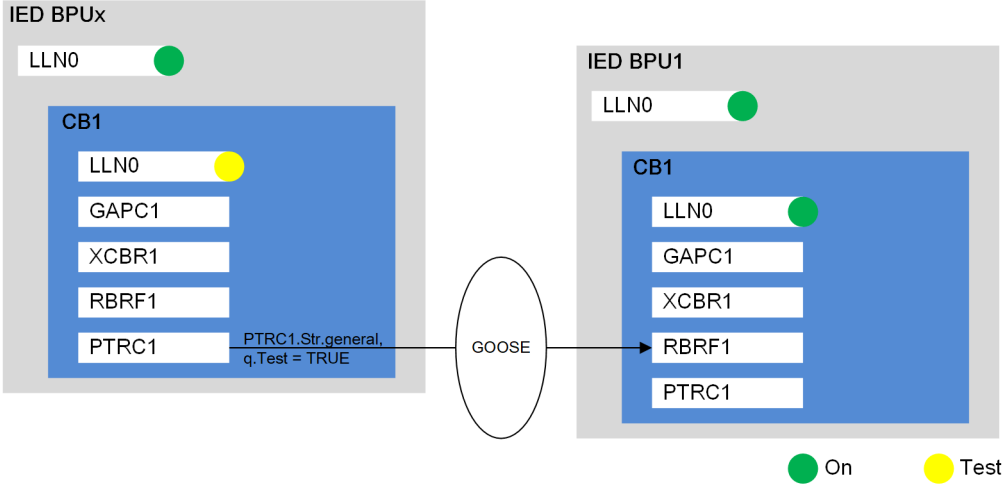


[lo_GOOSE test CB failure, 1, en_US]

Figure 2-1 Testing the Circuit-Breaker Failure

To test the circuit-breaker failure (RBRF) as shown in the preceding figure, you must set the Logical Node **IED BPU1/CB1/RBRF1** to the behavior ***Test*** (or ***Test/Relay blk.***). The publisher device sends the protection activation with the **q.Test** being set or not. In both cases, the published value with the signal *PTRC1.Str.general* is considered. The Logical Node **IED BPU1/CB1/RBRF1** sends a signal *Operate* flagged with **q.Test** = ***TRUE*** and the circuit breaker (**IED BPU1/CB1/XCBR1**) opens.

Use Case: Test Mismatch when Testing the Circuit-Breaker Failure



[lo_GOOSE test CB failure_test mismatch, 1, en_US]

Figure 2-2 Test Mismatch when Testing the Circuit-Breaker Failure

The preceding figure illustrates the test-mismatch scenario: The incoming signal indicates **`q.Test`** = ***`TRUE`***, while the receiver has the behavior ***`On`***. The expected processing of quality attributes for test mismatch must be configured in DIGSI 5. In the use case, the resulting **`q.Test`** must be set to ***`FALSE`*** in case of test mismatch. Therefore the following must be defined:

- Set **`Safe state`** under **Common settings** to ***`False`*** (*Figure 2-3*). Then set **`Define value`** under **Test mismatch** to ***`apply safe state value`*** (*Figure 2-4*).

Or alternatively,

- Set **`Define value`** under **Test mismatch** to ***`set value to "false"`***.

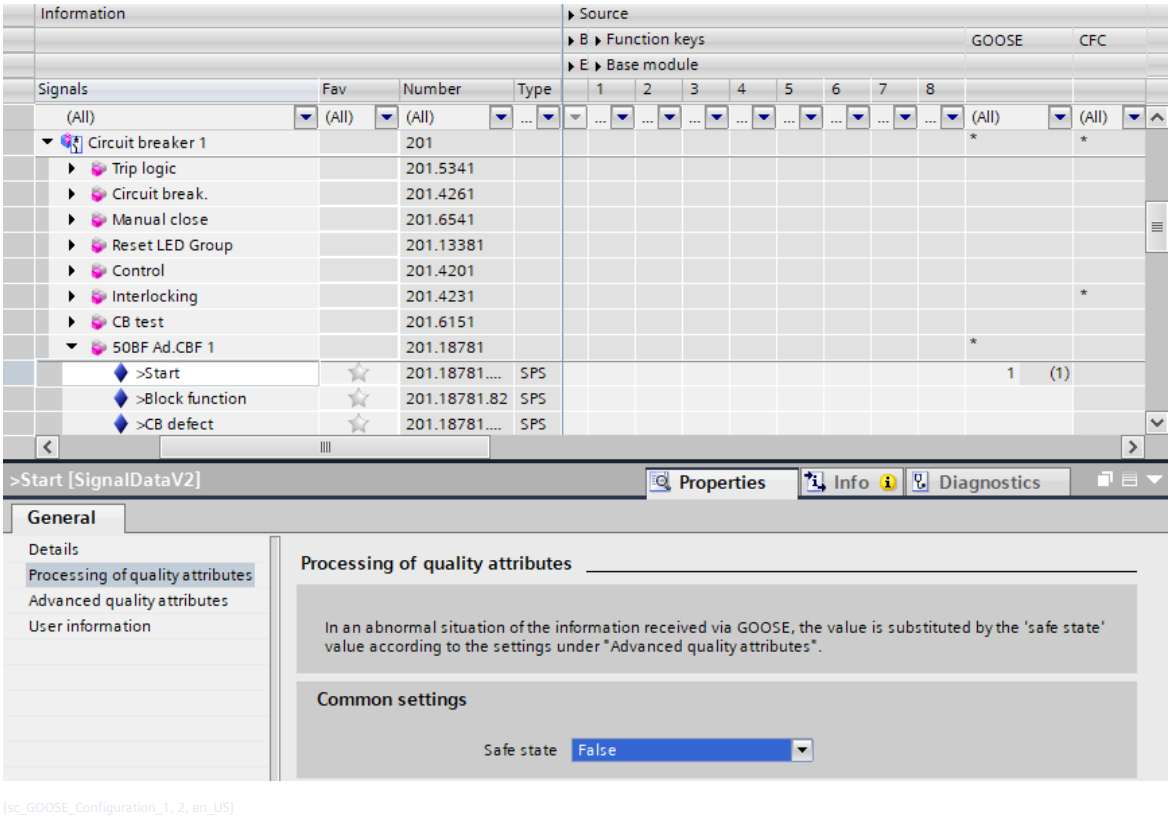
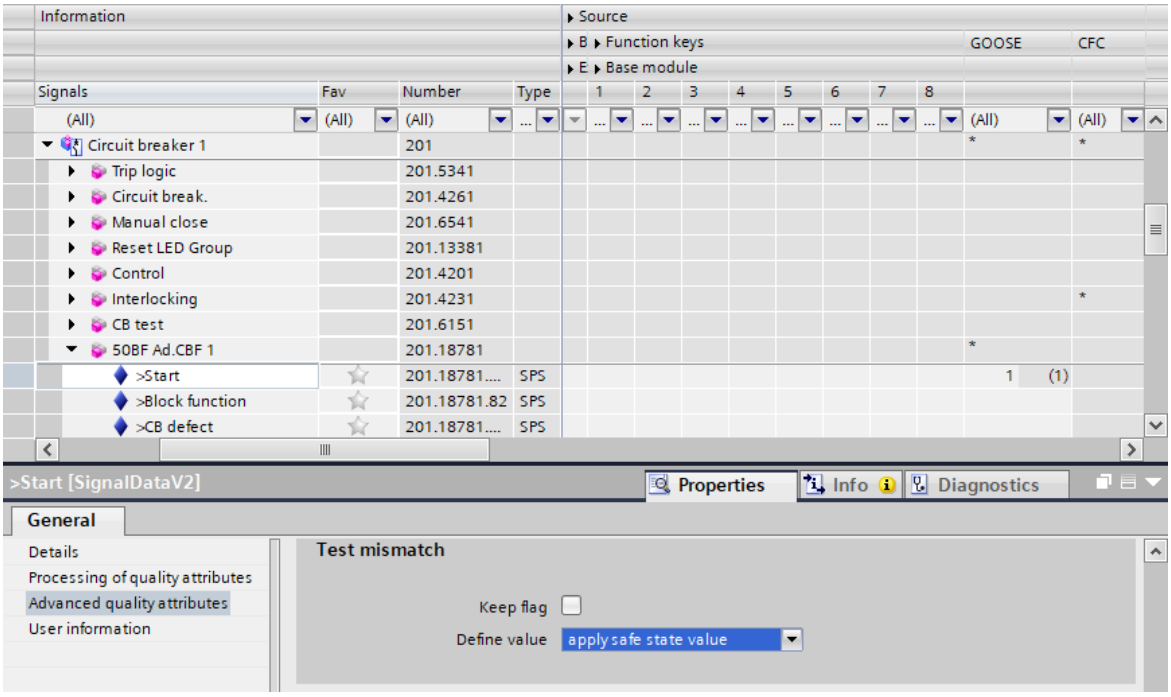


Figure 2-3 Processing of Quality Attributes – Common Settings



[sc_GOOSE_Configuration_2, 2, en_US]

Figure 2-4 Advanced Quality Attributes - Test Mismatch

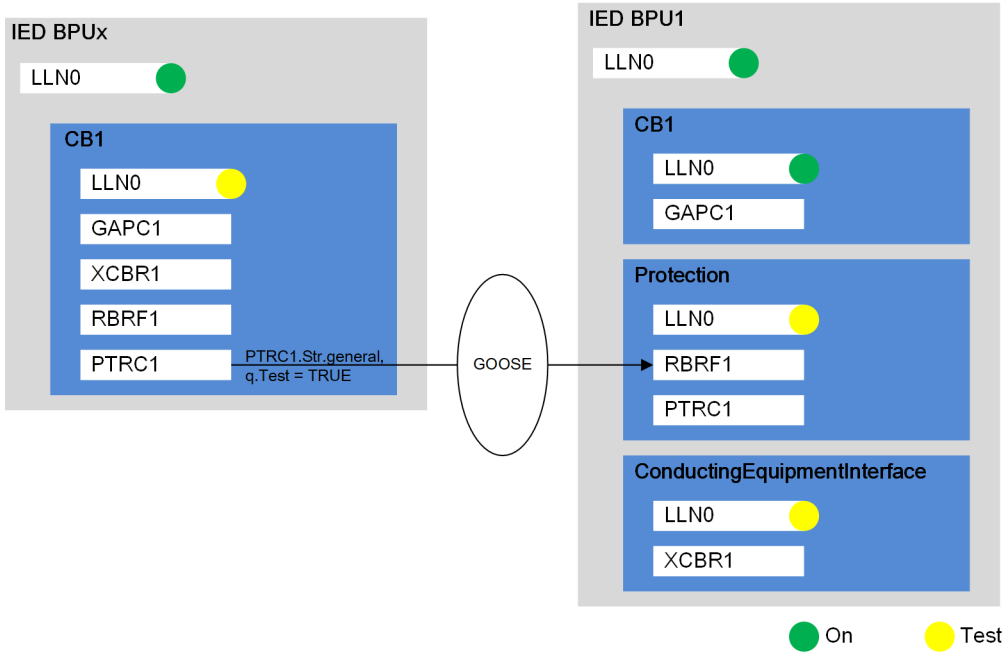
Use Case: Testing the Circuit-Breaker Failure in the Flexible-Engineering Scenario

If the default IEC 61850 structure does not meet your requirements, you can use flexible engineering to adjust the structure.

In this case, flexible engineering is used to move the Logical Nodes from the circuit-breaker Logical Device to user-defined Logical Devices (similar to *Figure 3-4*):

- The instances **PTRC1** and **RBRF1** are moved from the native Logical Device **CB1** to a user-defined Logical Device **Protection**.
- The instance **XCBR1** is moved from the native Logical Device **CB1** to a user-defined Logical Device **ConductingEquipmentInterface**.

The following figure shows the circuit-breaker failure test scenario in this case.



[lo_GOOSE test CB failure_flexible_engineering, 1, en_US]

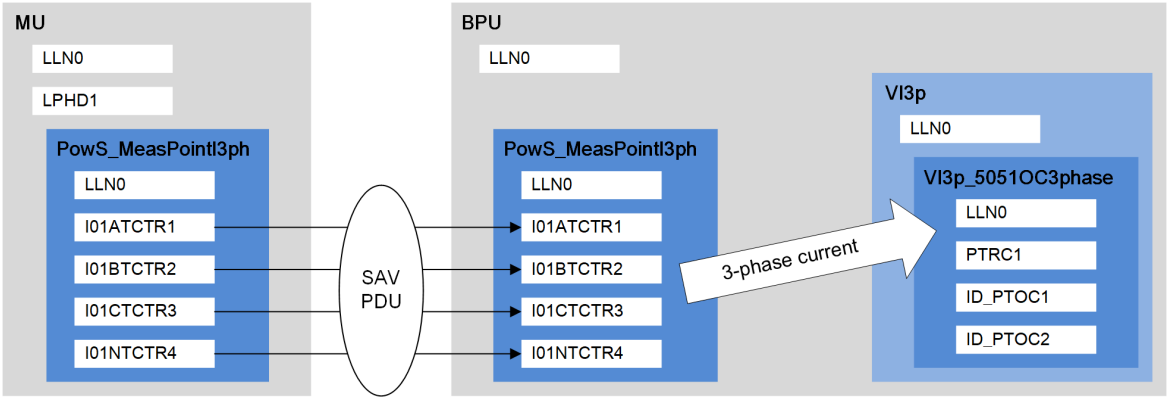
Figure 2-5 Testing the Circuit-Breaker Failure in the Flexible-Engineering Scenario

During the test, both the user-defined Logical Devices **Protection** and **ConductingEquipmentInterface** must be turned into the behavior **Test** to achieve the same system behavior as that in *Figure 2-1*.

2.2 Behavior of SMV Processing during Test of Process-Bus Connections

2.2.1 Input Measuring Points with Process-Bus Connection

Currents or voltages are gathered and distributed via measuring points within the SIPROTEC 5 device. The following figure shows an example of a current measuring point of a Merging Unit (MU) connected to an input-current measuring point of a Bay Protection Unit (BPU) via the process-bus interface.



[lo_pb_connections, 1, en_US]

Figure 2-6 Process-Bus Connection

SAV	Sampled Analog Value
PDU	Protocol Data Unit

The measuring points within the BPU then distribute the 3-phase currents to the connected functions.

The principles for the 3-phase currents and 3-phase voltages are the same. The following cases use a 3-phase current as an example.

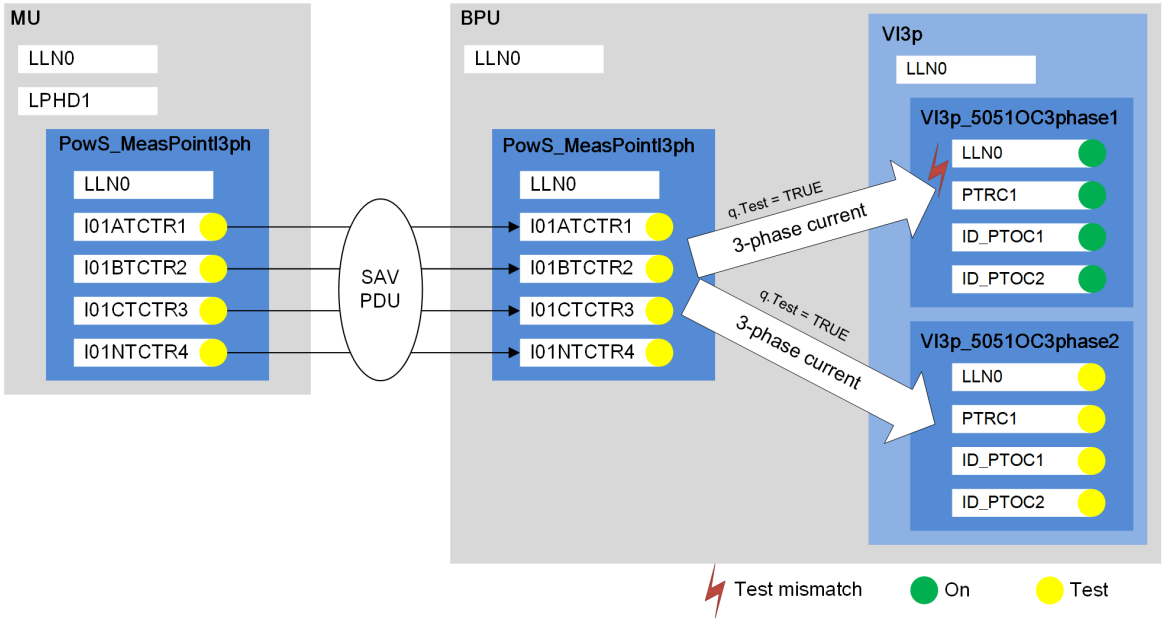
2.2.2 Use Cases for IEC 61850 Compliant Testing

The following use cases illustrate how SIPROTEC 5 handles symmetrical[2] and asymmetrical[3] behavior (**Beh**) values of the transformer Logical Nodes that are associated to a measuring point. Nevertheless, Siemens recommends setting coherent values of **Beh** for all Logical Nodes that are associated to a measuring point in a test scenario.

NOTE

If the value **Beh** of only one of the current (TCTR) or voltage (TVTR) transformer Logical Nodes of a current or voltage measuring point is set to ***Test*** while the value **Beh** of other Logical Nodes is set to ***On*** (incoherent values of **Beh**), the internal forwarded current or voltage sample is considered as the test sample. Use case 5 illustrates a case of incoherent values of **Beh**.

Use Case 1: Symmetrical Behavior Values



[lo_pb_test_sample, 3, en_US]

Figure 2-7 Use Case 1: Symmetrical Behavior Values

The Logical Nodes of transformers in the MU are in the behavior ***Test*** (or ***Test/Relay blk.***).

The SAV PDU transmits the test sample.

The Logical Nodes of input transformers in the BPU are in the behavior ***Test*** (or ***Test/Relay blk.***) and accept the test sample with the **Health** set to ***Ok***.

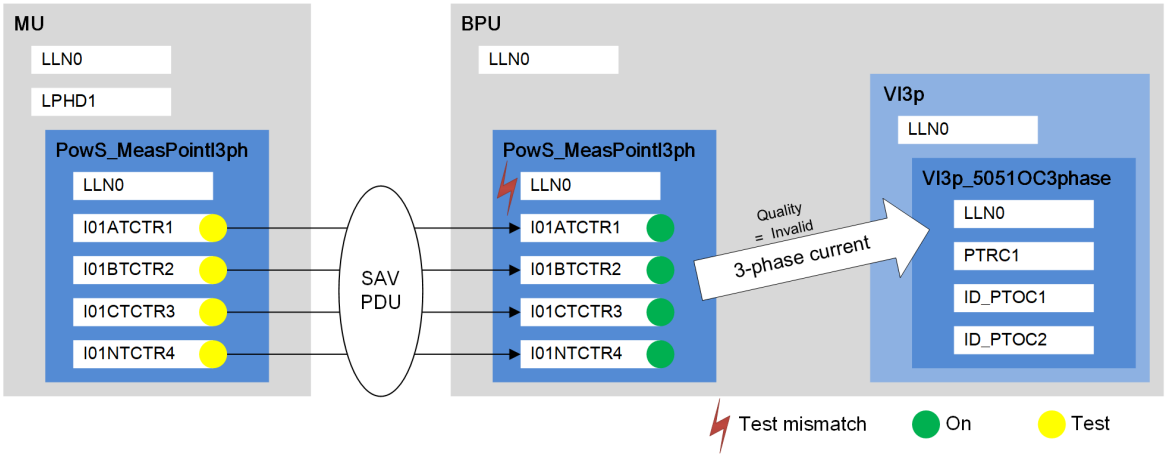
The internal 3-phase current is forwarded as the test sample.

2 symmetrical: The transformer Logical Nodes of 3 phases always have the same behavior value.

3 asymmetrical: The transformer Logical Nodes of 3 phases may have different behavior values.

- Any functions connected to the 3-phase current with their behavior set to ***Test*** (or ***Test/Relay blk.***) use the test sample.
- Any functions connected to the 3-phase current with their behavior set to ***On*** (or ***Relay blocked***) indicate the test mismatch with the **Health** set to ***Alarm*** and are inactive.

Use Case 2: Symmetrical Behavior Values

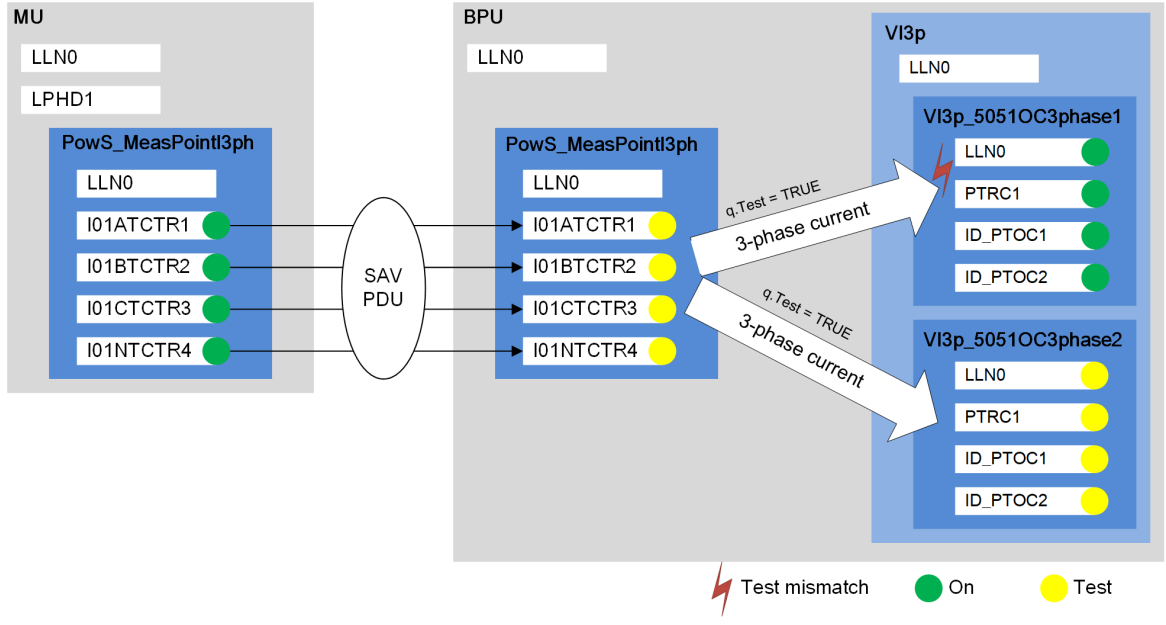


[lo_pb_test_mismatch, 2, en_US]

Figure 2-8 Use Case 2: Symmetrical Behavior Values

The Logical Nodes of transformers in the MU are in the behavior ***Test*** (or ***Test/Relay blk.***).
The SAV PDU transmits the test sample.
The Logical Nodes of input transformers in the BPU are in the behavior ***On*** (or ***Relay blocked***) and indicate the test mismatch with the **Health** set to ***Alarm***.
The internal 3-phase current is forwarded as the invalid sample.
Any functions connected to the 3-phase current are inactive.

Use Case 3: Symmetrical Behavior Values



[lo_pb_test_part 1, 3, en_US]

Figure 2-9 Use Case 3: Symmetrical Behavior Values

The Logical Nodes of transformers in the MU are in the behavior ***On*** (or ***Relay blocked***).

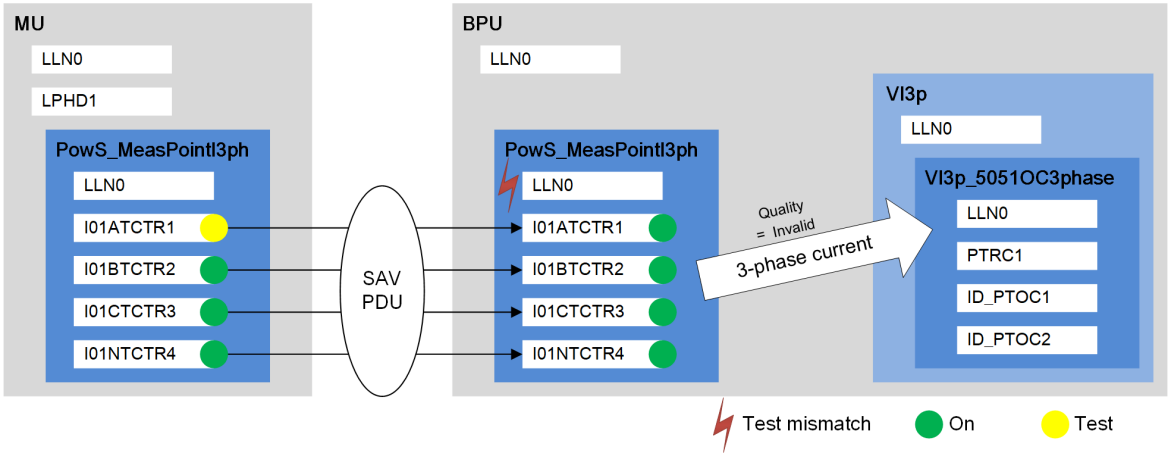
The SAV PDU transmits the operating sample.

The Logical Nodes of input transformers in the BPU are in the behavior ***Test*** (or ***Test/Relay blk.***) and accept the operating sample with the **Health** set to ***Ok***.

The internal 3-phase current is forwarded as the test sample.

- Any functions connected to the 3-phase current with their behavior set to ***Test*** (or ***Test/Relay blk.***) use the test sample.
- Any functions connected to the 3-phase current with their behavior set to ***On*** (or ***Relay blocked***) indicate the test mismatch with the **Health** set to ***Alarm*** and are inactive.

Use Case 4: Asymmetrical Behavior Values



[lo_pb_test_part 3, 2, en_US]

Figure 2-10 Use Case 4: Asymmetrical Behavior Values

Only one Logical Node of the transformer in the MU is in the behavior ***Test*** (or ***Test/Relay blk.***).

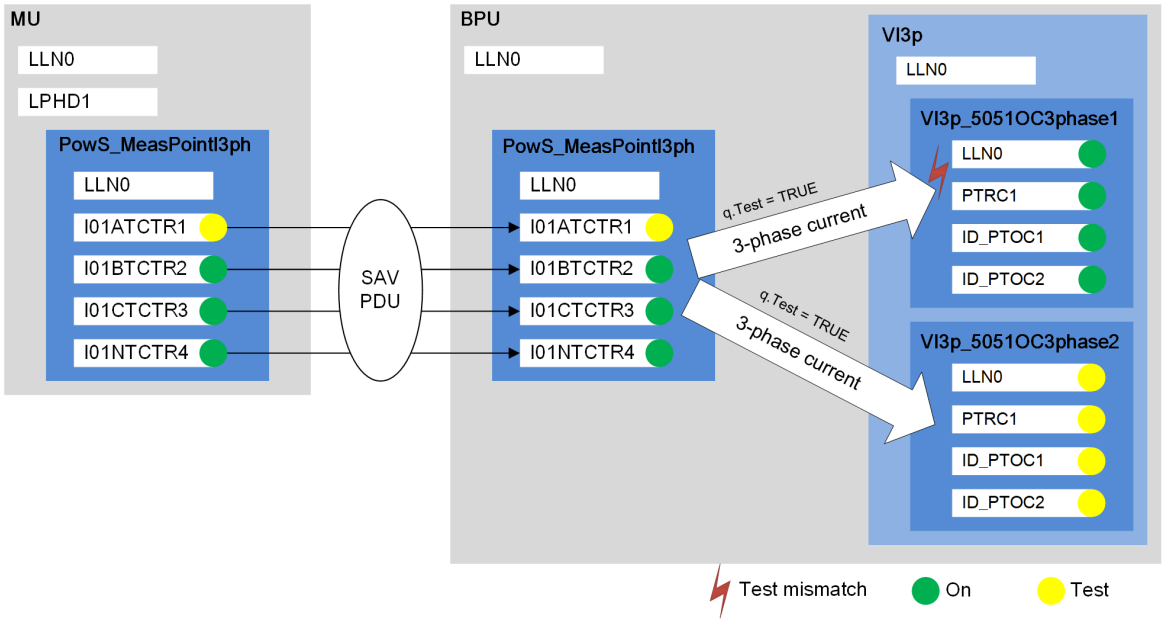
The SAV PDU transmits the test sample.

The Logical Nodes of input transformers in the BPU are in the behavior ***On*** (or ***Relay blocked***) and indicate the test mismatch with the **Health** set to ***Alarm***.

The internal 3-phase current is forwarded as the invalid sample, as long as none of the Logical Nodes of the input transformer in the BPU is in the behavior ***Test*** (or ***Test/Relay blk.***).

Any functions connected to the 3-phase current are inactive.

Use Case 5: Asymmetrical Behavior Values



[lo_pb_test_part 4, 3, en_US]

Figure 2-11 Use Case 5: Asymmetrical Behavior Values

Only one Logical Node of the transformer in the MU is in the behavior ***Test*** (or ***Test/Relay blk.***).
The SAV PDU transmits the test sample.
Only one Logical Node of the input transformer in the BPU is in the behavior ***Test*** (or ***Test/Relay blk.***).
The Logical Nodes accept the test sample with the **Health** set to ***Ok***.
The internal 3-phase current is forwarded as the test sample.

- Any functions connected to the 3-phase current with their behavior set to ***Test*** (or ***Test/Relay blk.***) use the test sample.
- Any functions connected to the 3-phase current with their behavior set to ***On*** (or ***Relay blocked***) indicate the test mismatch with the **Health** set to ***Alarm*** and are inactive.

NOTE

If at least one Logical Node **TCTR** in the BPU is set to ***Test***, the internal 3-phase current is forwarded as the test sample.

NOTE

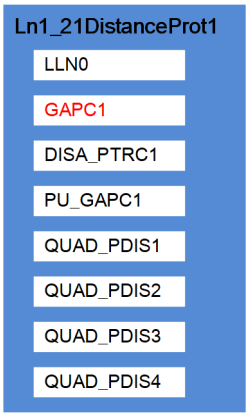
If the asymmetric behavior values of the MU and of the BPU are not in the same phase, the device reacts differently. For example, if only the Logical Node **I01ATCTR1** (phase A) of the MU and only the Logical Node **I01BTCTR2** (phase B) of the BPU are in the behavior ***Test***, then the Logical Node **I01ATCTR1** (phase A) of the BPU indicates the test mismatch with the **Health** set to ***Alarm***, and the internal 3-phase current is forwarded as the invalid sample.

2.2.3 Impact of the Location of a Grouped Functionality during Testing with Test Samples

Especially when the process-bus connection is used, the location of the Logical Node that represents a grouped functionality in the Logical-Device hierarchy impacts the test execution.

Concept of a Grouped Functionality

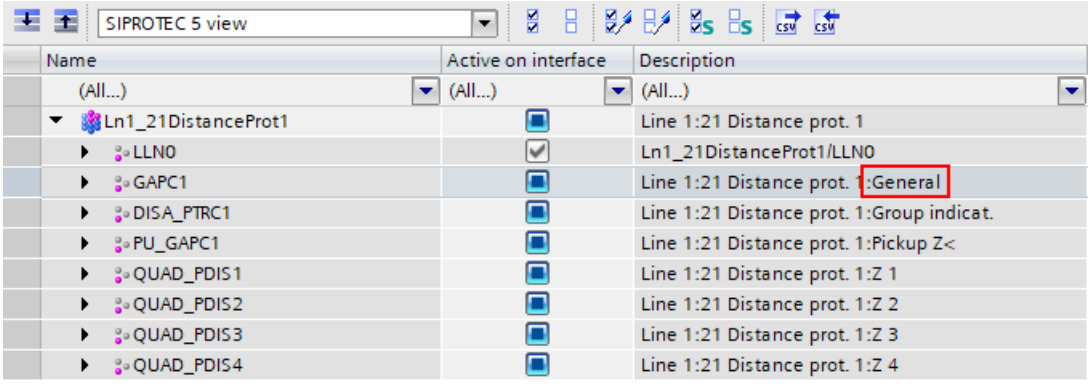
Each group, such as a bay, a function group, or a function, usually requires the implementation of a dedicated function responsible for some grouping aspects. This grouped functionality is represented by a Logical-Node instance of the class **GAPC** (generic automatic process control), as shown in the following figure.



[dw_GAPC for grouped functionality, 1, —_–]

Figure 2-12 Usage of **GAPC** for a Grouped Functionality

The instance of **GAPC** is characterized by a visible description ending with **:General** in the Editor **IEC 61850 structure** in DIGSI 5, as shown in *Figure 2-13*. Most of the time, the **GAPC** is the preparing or preprocessing information that must be shared with multiple function elements within a bay, a function group, or a function. The behavior (**Beh**) and readiness of the **GAPC** are probably of relevance when you use the IEC 61850 testing functionality on the Logical-Device level, especially when you move function elements that belong to a given function within the IEC 61850 structure. Therefore, it is important to understand the role and position of the **GAPC** in the chain of information. In the test scenario, if the Logical Nodes in a grouped functionality use the data from the **GAPC**, the **GAPC** must have the behavior ***Test*** to perform the test.

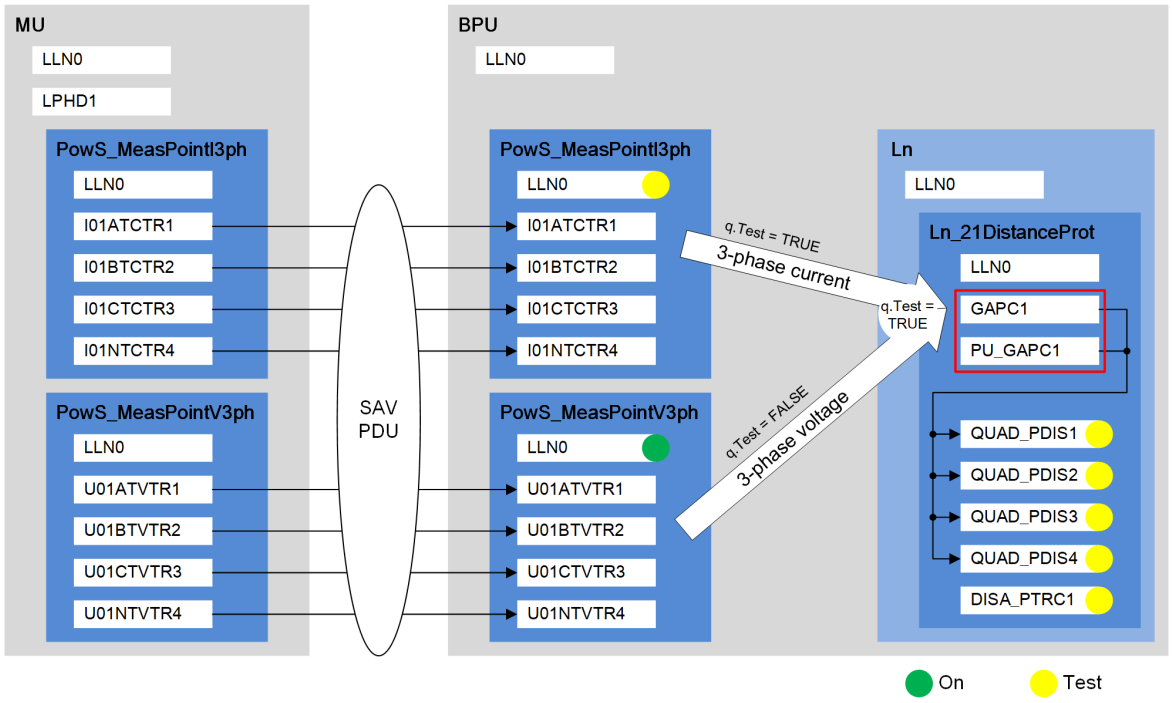


[sc_GAPC, 1, en_US]

Figure 2-13 Example of a Grouped Functionality

Impact of the Location of the Grouped Functionality when Operating a Function under Test

For example, the input-current measuring point distributes test samples and the input-voltage measuring point distributes operating samples. As the impedance calculation requires both voltage and current values, the function **Distance protection** must have the behavior ***Test*** (or ***Test/Relay blk.***) to accept the current test samples.



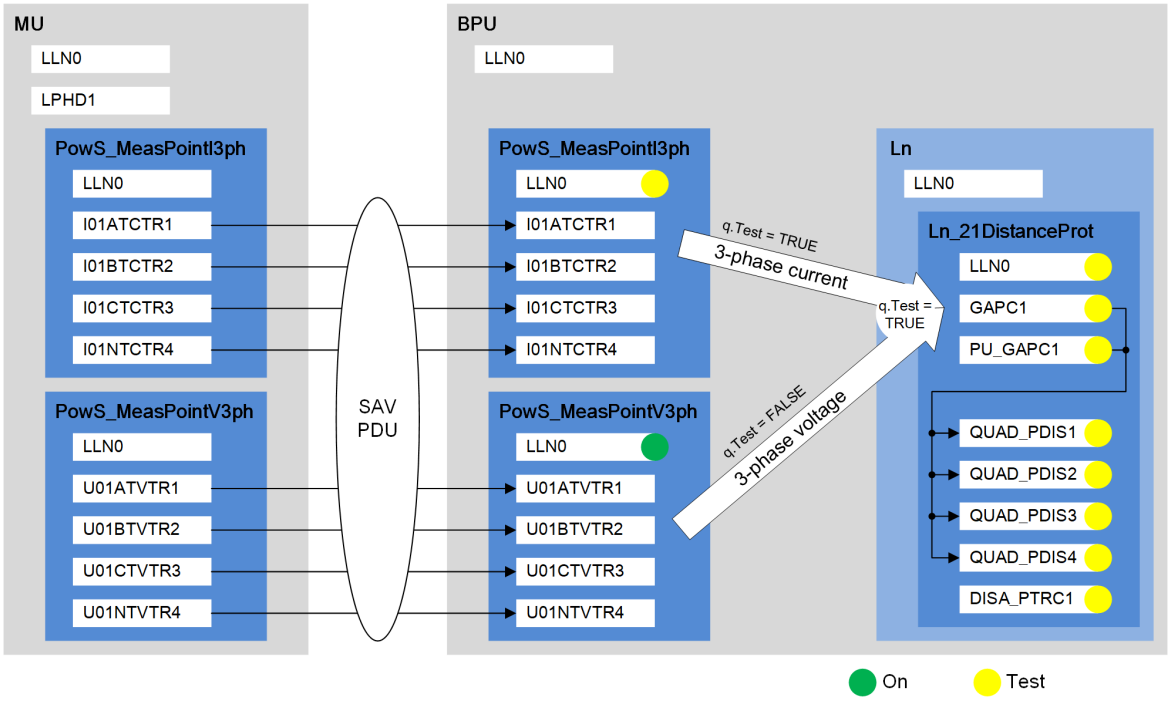
[lo_impact of grouped functionality, 2, en_US]

Figure 2-14 Impact of the Grouped-Functionality Behavior when Turning a Function into Test

As shown in the preceding figure, the group-functionality (**GAPC1** and **PU_GAPC1**) behavior must match the test scenario. As the individual zone (for example, **QUAD_PDIS1**) also needs the shared preprocessing results delivered by the grouped functionality, it is not sufficient to turn the Logical Node of each individual zone into the behavior ***Test*** (or ***Test/Relay blk.***).

Figure 2-15 and *Figure 2-16* illustrate 2 use cases of turning the function **Distance protection** into the behavior ***Test*** (or ***Test/Relay blk.***).

Use Case 1: Function Elements Located in the Native Logical Device



[lo_function consuming test samples, 1, en_US]

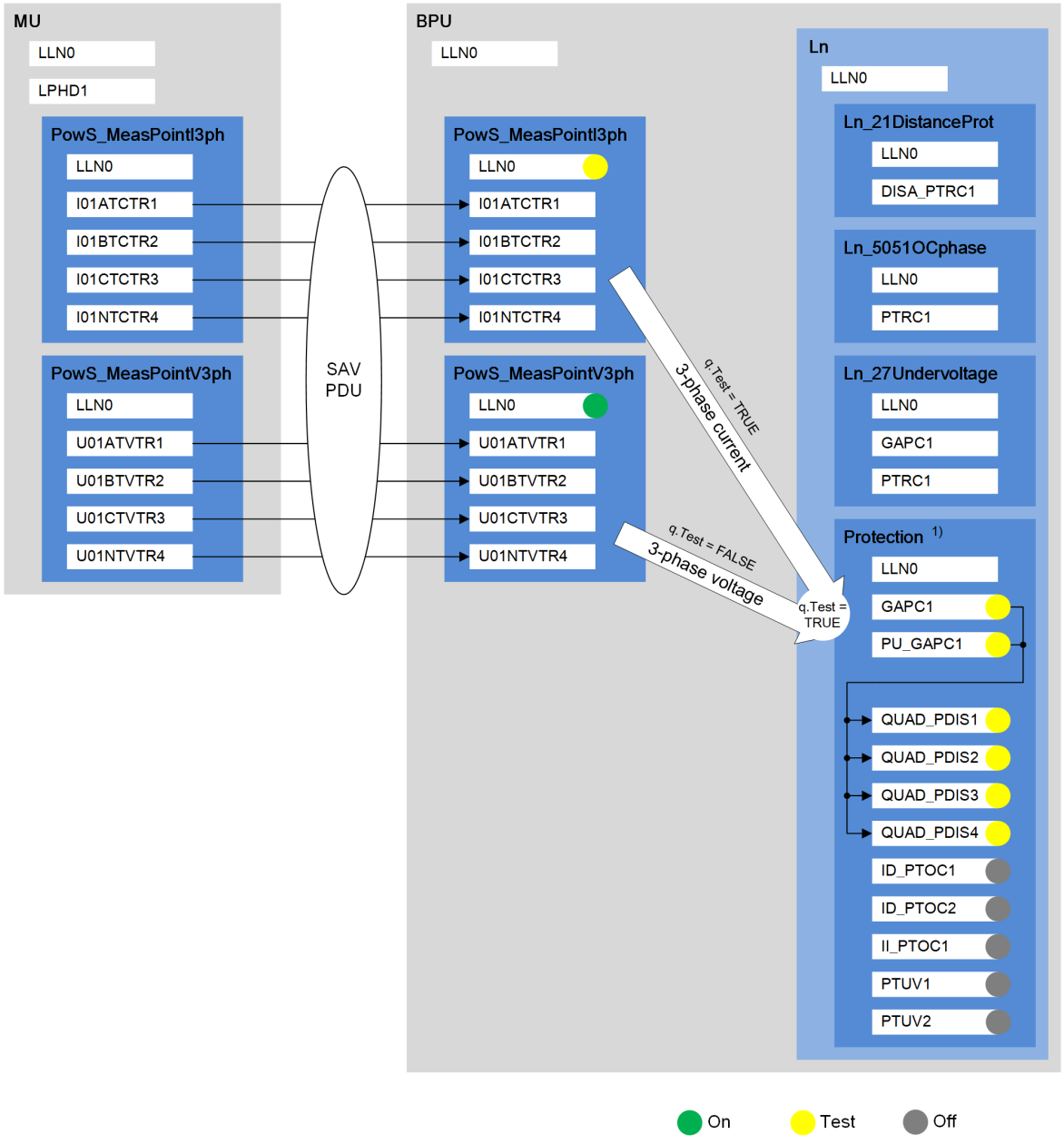
Figure 2-15 Use Case 1: Function Elements in the Native Logical Device **Ln_21DistanceProt**

In use case 1, the IEC 61850 representation is the native SIPROTEC 5 hierarchical representation. The native Logical Device **Ln_21DistanceProt** can be turned into the behavior ***Test*** (or ***Test/Relay blk.***) via **Ln_21DistanceProt/LLN0.Mod**.

NOTE

If the Logical Node **GAPC** representing the grouped functionality is hidden (refer to *3.1 Hiding a Grouped Functionality*), the **Ln_21DistanceProt/LLN0.Mod** can be used to set the entire Logical Device to ***Test*** (or ***Test/Relay blk.***). Then the function **Distance protection** has the behavior ***Test*** (or ***Test/Relay blk.***) to accept the test samples.

Use Case 2: Function Elements Located in a User-Defined Logical Device



[lo_function consuming test samples_LD protection, 1, en_US]

Figure 2-16 Use Case 2: Function Elements Moved to the User-Defined Logical Device **Protection**

(1) User-defined Logical Device

In use case 2, all the function elements are moved from the native Logical Device **Ln_21DistanceProt** to the user-defined Logical Device **Protection**.

To accept the test samples, all the function elements belonging to the **Ln_21DistanceProt** must have the behavior ***Test*** (or ***Test/Relay blk.***):

- The entire Logical Device **Protection** can be turned into the behavior ***Test*** (or ***Test/Relay blk.***).
  Siemens recommends changing the behavior of the entire Logical Device. In this case, if some of the Logical Nodes are hidden, the entire Logical Device **Protection** must be turned into the behavior ***Test*** (or ***Test/Relay blk.***) using **Protection/LLN0.Mod** to ensure all the function elements have the behavior ***Test*** (or ***Test/Relay blk.***).

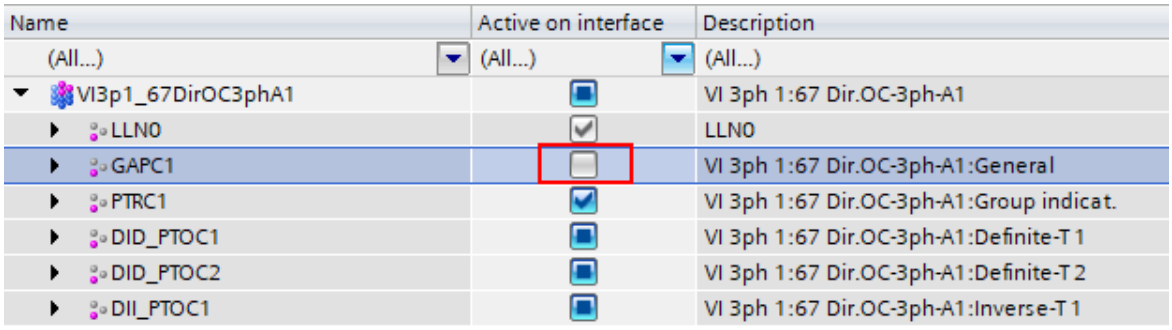
- If all the Logical Nodes are visible, these Logical Nodes can be turned individually into the behavior ***Test*** (or ***Test/Relay blk.***).
- With the behavior of the Logical Device **Protection** set to ***Test*** via **Protection/LLN0.Mod**, all functions that are hierarchically arranged below the Logical Device **Protection** are set to ***Test*** by means of inheritance. If the influence of other protection functions (such as **Overcurrent protection** and **Undervoltage protection**) is to be excluded during testing, they can be individually set to the behavior **Off**.

3 Behaviors in Flexible Engineering

SIPROTEC 5 supports flexible engineering of the IEC 61850 structure. This chapter describes the mode/behavior management of the Logical-Device hierarchy with flexible engineering.

3.1 Hiding a Grouped Functionality

If the native Logical Device of a function is kept or visible in the IEC 61850 structure, you can hide the Logical Node **GAPC** of the grouped functionality from the IEC 61850 communication interface (deselected **GAPC**), as shown in *Figure 3-1*. Then the behavior of the grouped functionality follows the behavior of the function, which is determined by the Logical Node **LLN0** of the native Logical Device of the function (in this example, **VI3p1_67DirOC3phA1/LLN0**).



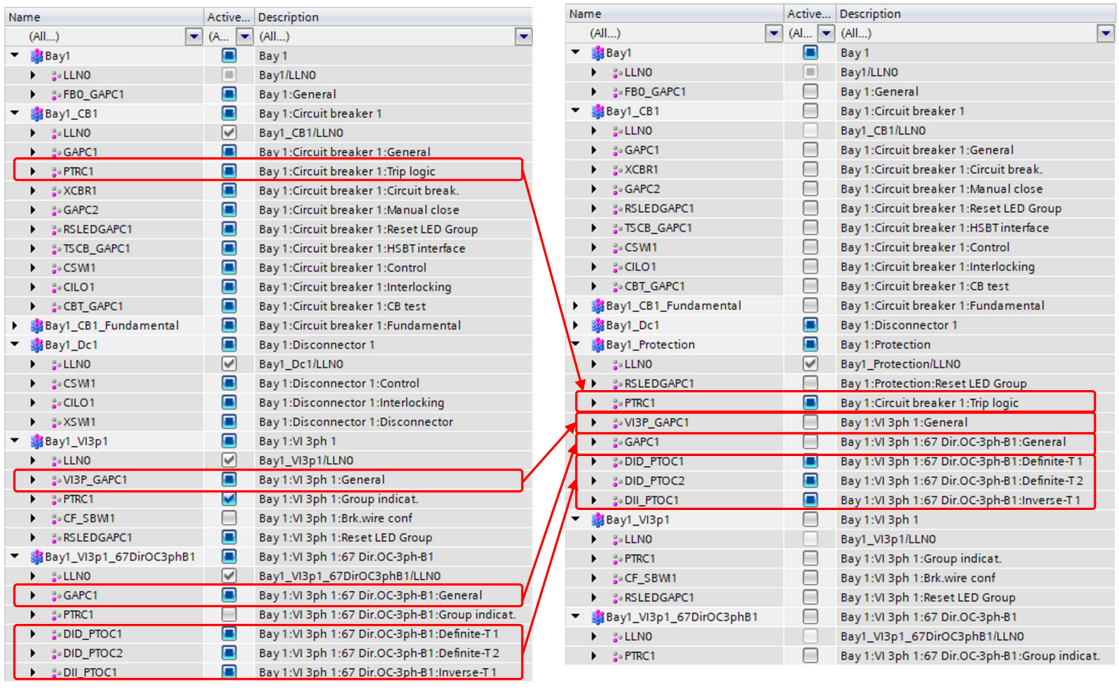
[sc_Hide_group_functionality, 1, en_US]

Figure 3-1 Example of Hiding a Grouped Functionality

3.2 Moving Logical Nodes

Via flexible engineering, the protection-stage Logical Nodes can be moved from the native Logical Device to a user-defined Logical Device. The following options are available for the grouped functionality (Logical Node **GAPC**) during flexible engineering:

- Option 1 (preferred): The grouped functionality is moved together with the protection-stage Logical Nodes that are members of the function to the same Logical Device.



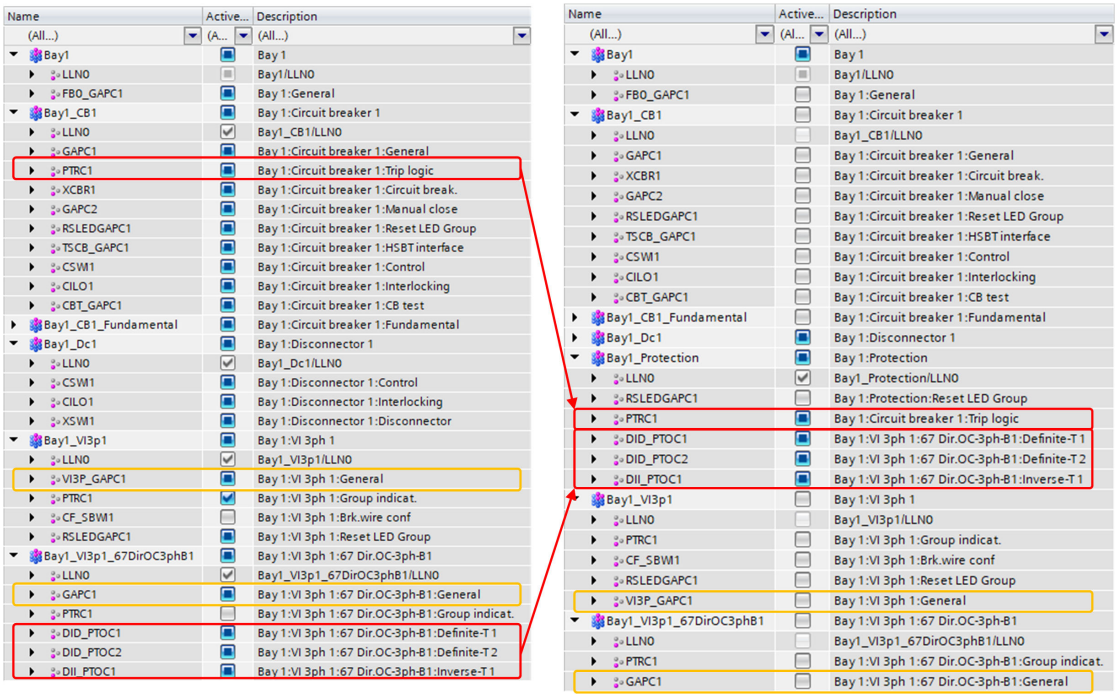
[le_stage_moved_option1, 1, en_US]

Figure 3-2 Option 1: Moving the Grouped Functionality Together with the Protection Stages

As shown in the preceding figure, the Logical Nodes **VI3P_GAPC1** and **GAPC1** representing the grouped functionalities are moved together with the protection stages to the same Logical Device **Bay1_Protection**. The grouped functionalities are hidden, that is, not shown via the IEC 61850 communication interface.

In this option, the behavior of the grouped functionalities follows the behavior of the hosting function **Bay1_Protection** (determined by the **Bay1_Protection/LLN0**) and therefore is consistent with the behavior of the individual protection stages.

- Option 2: The grouped functionality remains in the native Logical Device representing the function.



[le_stage_moved_option2, 1, en_US]

Figure 3-3 Option 2: Remaining the Grouped Functionality when Moving Protection Stages

As shown in the preceding figure, the grouped functionality remains in the native Logical Devices of the functions **Bay1_VI3p1** and **Bay1_VI3p1_67DirOC3phB1**. Both the native Logical Device and the grouped functionality are hidden, that is, not shown via the IEC 61850 communication interface.

In this option, the behavior of the grouped functionality (**VI3P_GAPC1** and **GAPC1**) follows the behavior of the upper-level function.

In both options, the grouped functionality is not shown via the IEC 61850 communication interface. However, if the grouped functionality is shown via the IEC 61850 communication interface, the **Mod** of the grouped functionality must be controlled carefully. Different behaviors (**Beh**) between the grouped functionality and the individual stages probably lead to an unrealistic situation: For example, the grouped functionality in the behavior ***Test*** delivers test samples to stages in the behavior ***on***.

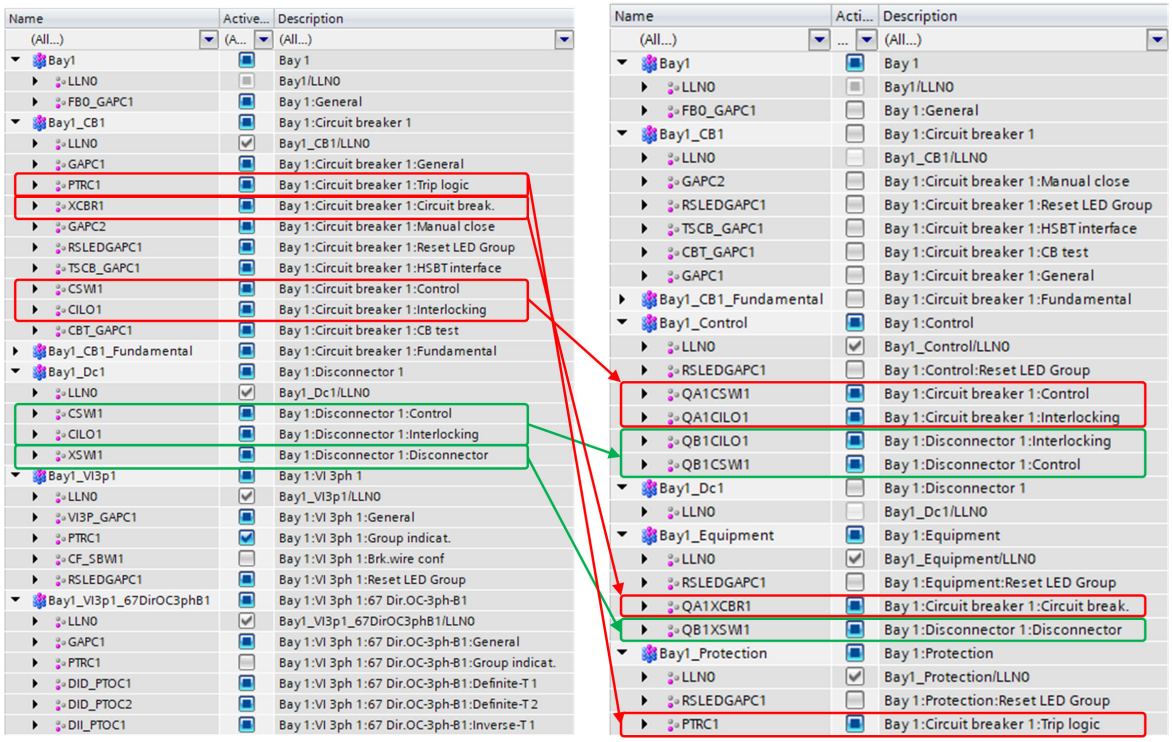


Figure 3-4 Example of Moving Native Function Elements to Different Logical Devices

The preceding figure shows how to split up the circuit-breaker and disconnector functionalities into different Logical Devices: **Control**, **Equipment**, and **Protection**.

First of all, create the user-defined Logical Devices **Control**, **Equipment**, and **Protection** in the appropriate hierarchical level (for example, under **Bay1**).

The red arrows show the relocation of the function elements from the native Logical Device **Bay1_CB1** (circuit breaker) to the Logical Devices **Bay1_Control** (**QA1CSWI1** and **QA1CILO1**), **Bay1_Equipment** (**QA1XCBR1**), and **Bay1_Protection** (**PTRC1**). The grouped functionality (**GAPC**) is not moved and the native Logical Device **Bay1_CB1** is hidden.

The green arrows show the relocation of the function elements from the native Logical Device **Bay1_Dc1** (disconnector) to the Logical Devices **Bay1_Control** (**QB1CSWI1** and **QB1CILO1**) and **Bay1_Equipment** (**QB1XSWI1**). The native Logical Device **Bay1_Dc1** is hidden.

3.3 Creating User-Defined Data Objects That Are Output of a CFC Logic

CFC charts that generate user-defined data objects must include the behavior (**Beh**) of the Logical Node that hosts the data objects via the block **CHART_STATE**, as shown in the following figure.

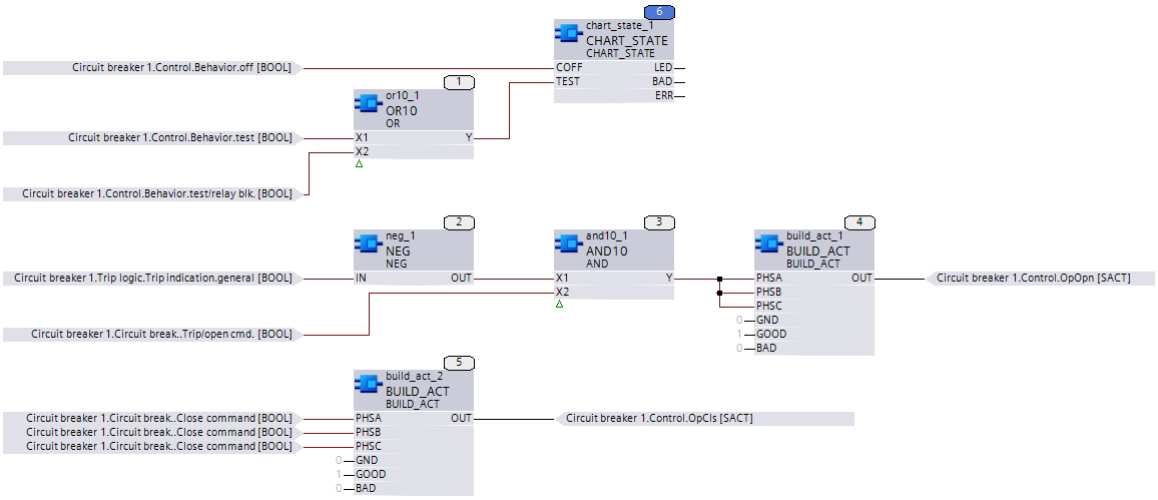


Figure 3-5 User-Defined Data Objects from Logic

The logic shown in the preceding figure illustrates the user-defined data objects *OpOpn* and *OpCls* extending the native Logical Node **Circuit-Breaker Control (CSWI)**.

The block **CHART_STATE** considers the **CSWI.Beh** (***Test***, ***Test/Relay blk.***, ***Off***) to set the quality attributes of *OpOpn* and *OpCls* adequately.

A chart can contain only one block **CHART_STATE** and therefore a chart can deal with the data-object value generation of only one Logical Node.

If the parameter **CFC chart quality handling** is set to ***automatic*** and the block **CHART_STATE** is correctly connected, a signal with **q.Test** = ***TRUE*** does not execute a CFC chart which is operating in normal state as per its logic block **CHART_STATE**. This is analogous to a quality processing, which retains the last valid value (***keep last valid value***) for a test-mismatch scenario.

3.4 Moving Data Objects

When moving data objects from one Logical Node to another Logical Node, note that the moved data objects are still under the influence of the behavior (**Beh**) of the native Logical Node that hosts the data objects.

Use Case 1: The Native Logical Nodes and the Target Logical Nodes Are Located in the Same Logical Device.

If the test scenarios can be executed using solely the **LLN0.Mod** for changing the **Beh** of part of the tree, then no further configuration efforts are needed: the native Logical Node and the target Logical Nodes share the same **Beh**, which is determined by the **LLN0.Beh**.

The following figure illustrates the use case of moving the measurement Logical Nodes together in a user-defined Logical Device named **Measurements** and splitting native data objects **MMXU** (measurement) in several Logical Nodes **MMXU**.

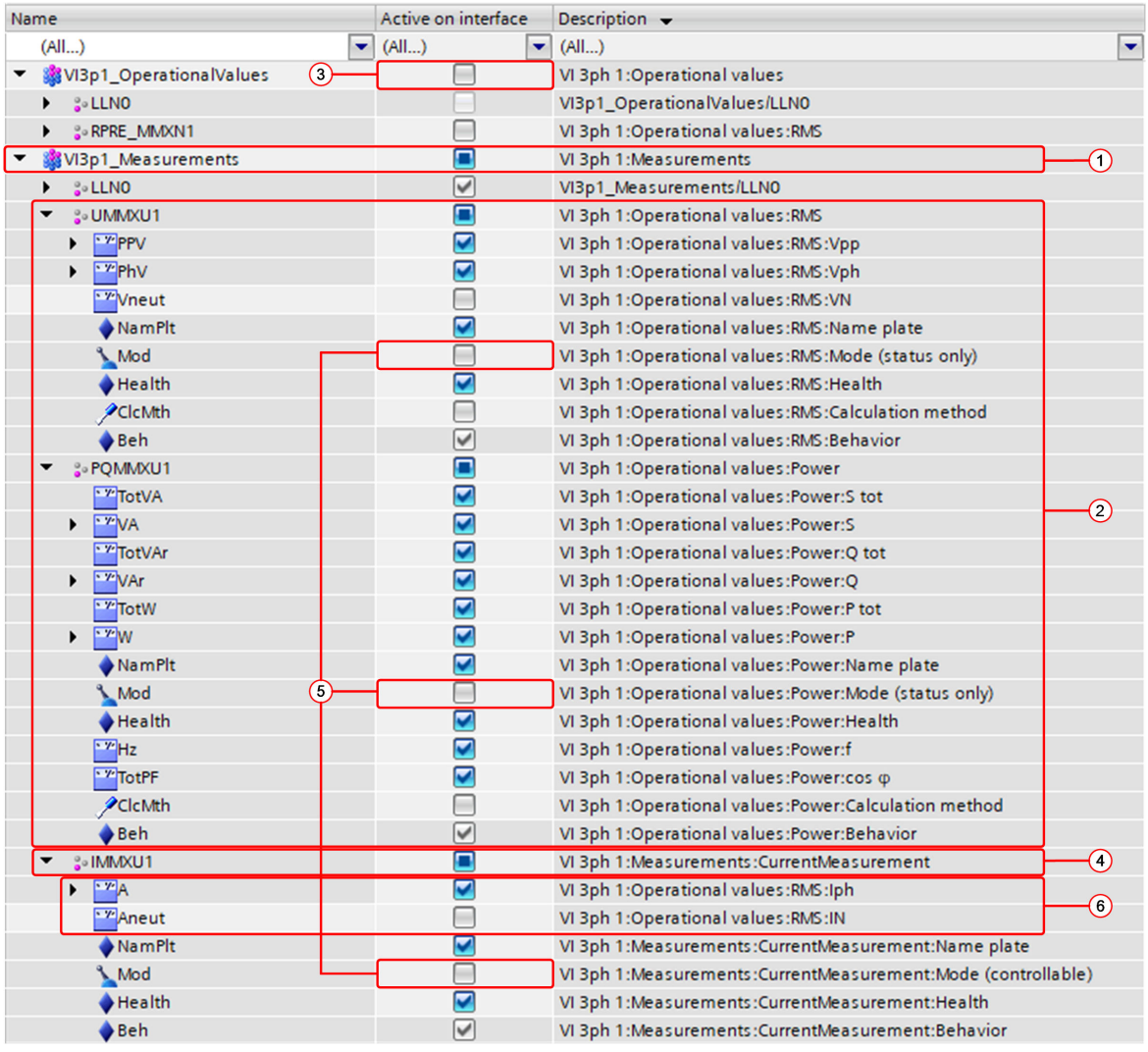


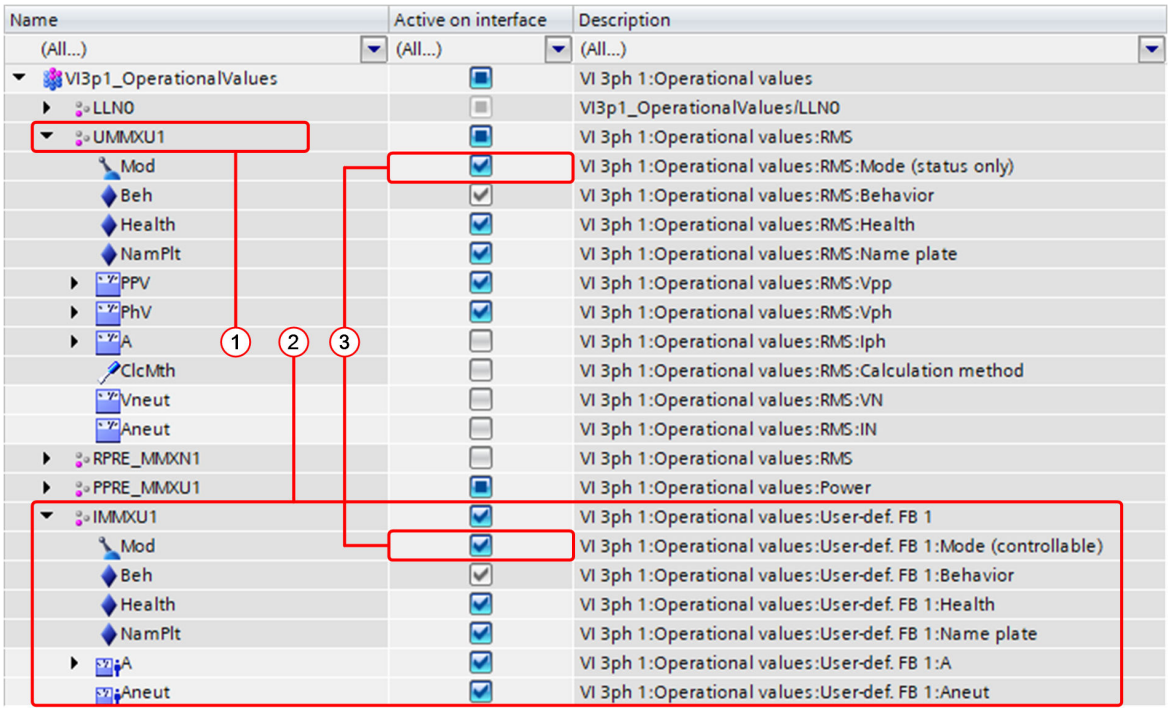
Figure 3-6 Moving Data Objects – Measurements

- ✧ Create a user-defined Logical Device **VI3p1_Measurements**. (❶)
- ✧ Move the native Logical Nodes **RPRE_MMXU1** and **PPRE_MMXU1** from the native Logical Device **VI3p1_OperationalValues** to the user-defined Logical Device **VI3p1_Measurements**, and rename them to **UMMXU1** and **PQMMXU1**, respectively. (❷)
- ✧ Hide the native Logical Device **VI3p1_OperationalValues** as it is not used anymore. (❸)
- ✧ Create a user-defined Logical Node **IMMXU1**. (❹)
- ✧ Hide the data objects **Mod** of the Logical Nodes **UMMXU1**, **PQMMXU1**, and **IMMXU1**. (❺)
  Then, the **Beh** of the 3 Logical Nodes is controlled by the **VI3p1_Measurements/LLN0.Mod**.
- ✧ Move the current data objects **A** and **Aneut** from **UMMXU1** to **IMMXU1**. (❻)

Use Case 2: The Native Logical Nodes and the Target Logical Nodes Shall Support Different Beh Values during a Test Execution.

In this use case, you must not move the data objects, but must create user-defined sibling data objects (data objects with the same CDC type) in the target Logical Node. In addition, you must create a CFC logic chart to connect the source and target data objects. The CFC logic must consider the **Beh** of the target Logical Node via the block **CHART_STATE** as described in *3.3 Creating User-Defined Data Objects That Are Output of a CFC Logic*.

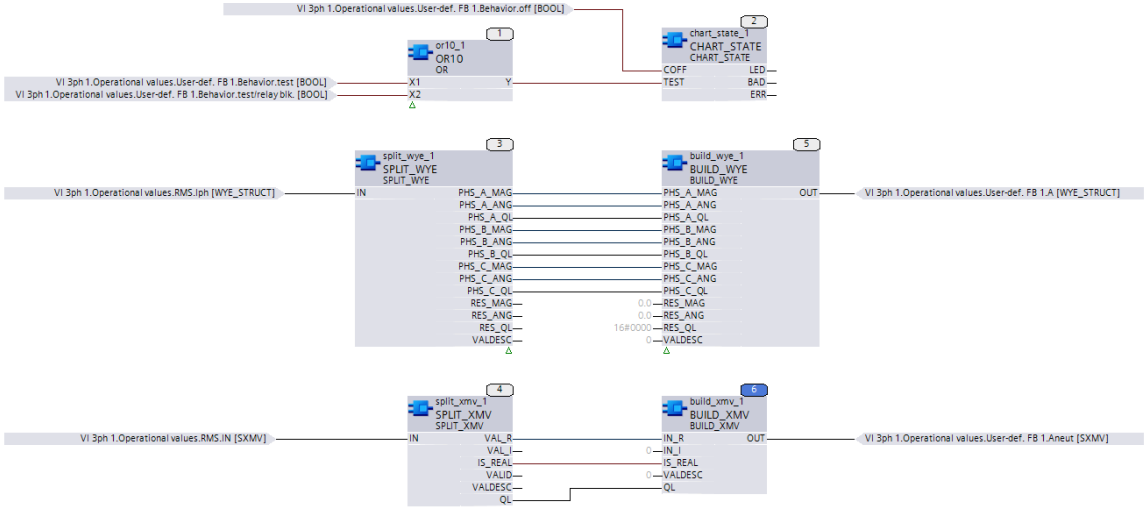
The following figure illustrates the use case where **MMXU** needs to be split: 1 **MMXU** for the voltage (**UMMXU**) and 1 **MMXU** for the current (**IMMXU**), and the user-defined **IMMXU.Mod** must be controllable.



[le_sibling DO, 1, en_US]

Figure 3-7 Creating Sibling Data Objects

- ✧ Rename the native Logical Node **RPRE_MMXU1** to **UMMXU1**. (❶)
- ✧ Create a user-defined Logical Node **IMMXU1** and the data objects **A** and **Aneut**. (❷)
- ✧ Select the check box in the column **Active on interface** of the individual data objects **Mod** so that they are visible via the IEC 61850 communication interface and **IMMXU1.Mod** is controllable. (❸)
- ✧ Create a CFC chart (see the following figure) to generate the data objects **A** and **Aneut** and to generate the adequate quality of the data objects following the **Beh** of the hosting Logical Node **IMMXU**.



[sc_CFC_connecting sibling DataObjects, 1, en_US]

Figure 3-8 Connecting Sibling Data Objects

Those adjustments made in Use Case 2 allow controlling the operation modes of the Logical Nodes **UMMXU1** and **IMMXU1** independently:

- The operation mode of **UMMXU1** is inherited from **VI3p1_OperationalValues/LLN0.Mod** or is controlled via **VI3p1_OperationalValues/UMMXU1.Mod**. With the latter, the operation mode can be set different from the operation mode of **IMMXU1**
- The operation mode of **IMMXU1** is inherited from **VI3p1_OperationalValues/LLN0.Mod** or is controlled via **VI3p1_OperationalValues/IMMXU1.Mod**. With the latter, the operation mode can be set different from the operation mode of **UMMXU1**.