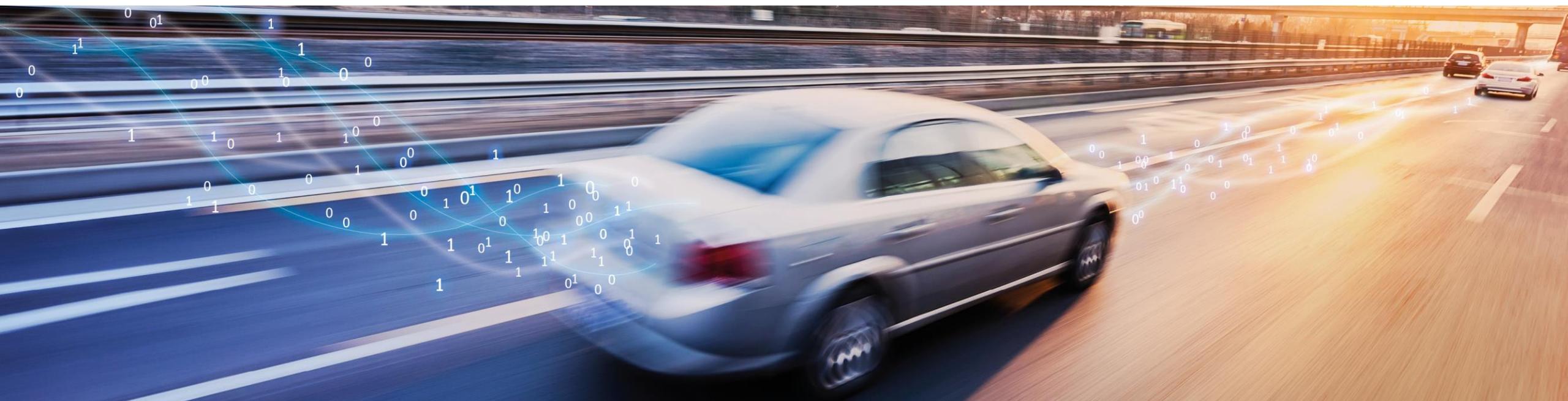


ASAM XIL 2.2

Release Presentation

Christian Sczryba-Neumann
dSPACE GmbH

27. February 2020



Agenda

1 **Introduction**

2 **Motivation for New Release**

3 **New Features**

4 **Other Changes**

5 **Backward-Compatibility**

6 **Deliverables**

Agenda

1 Introduction

2 Motivation for New Release

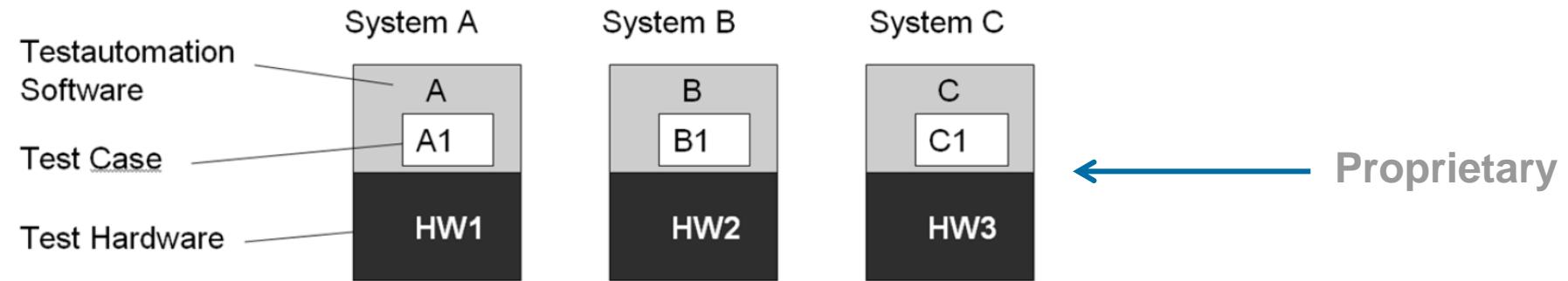
3 New Features

4 Other Changes

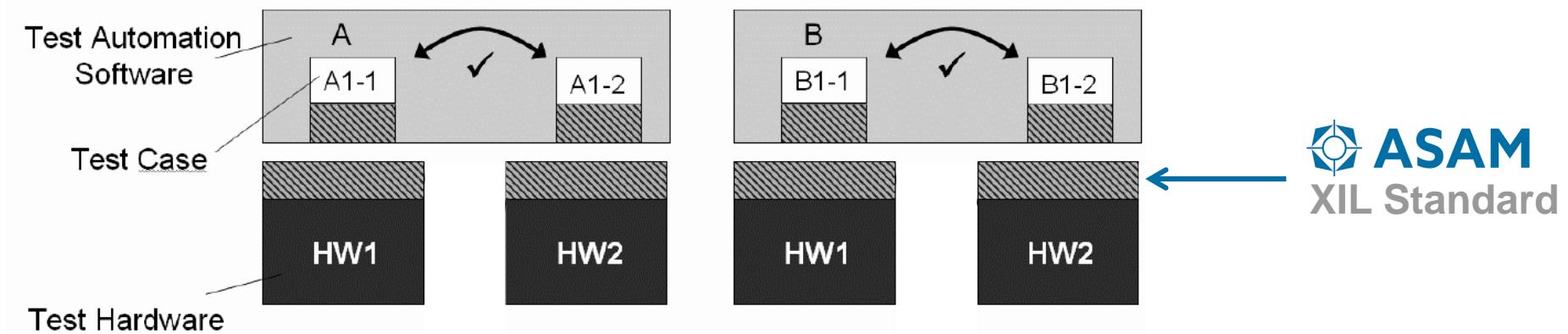
5 Backward-Compatibility

6 Deliverables

Motivation Of The Standard



→ Separation of Test HW and Test SW by means of standardized APIs



Concept Of Ports

MAPort

Model Access port provides access to the simulation model read and write parameters, capture and generate signals.

NetworkPort

provides access to field bus systems such as CAN. E. g. Allows measurement (monitoring) and transmission (single transmit or replay) of bus data.

EESPort

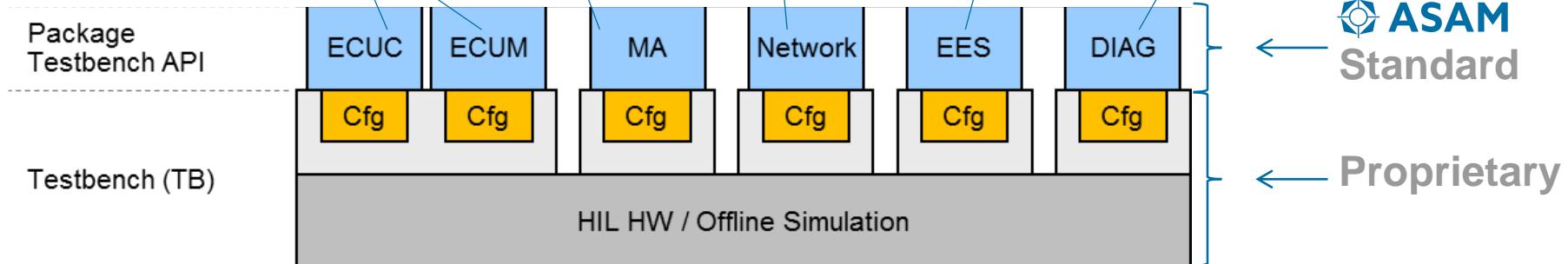
Electrical Error Simulation port controls electrical error simulation hardware. It allows the setup of different types of errors (e. g. short cuts).

ECUPorts (M+C)

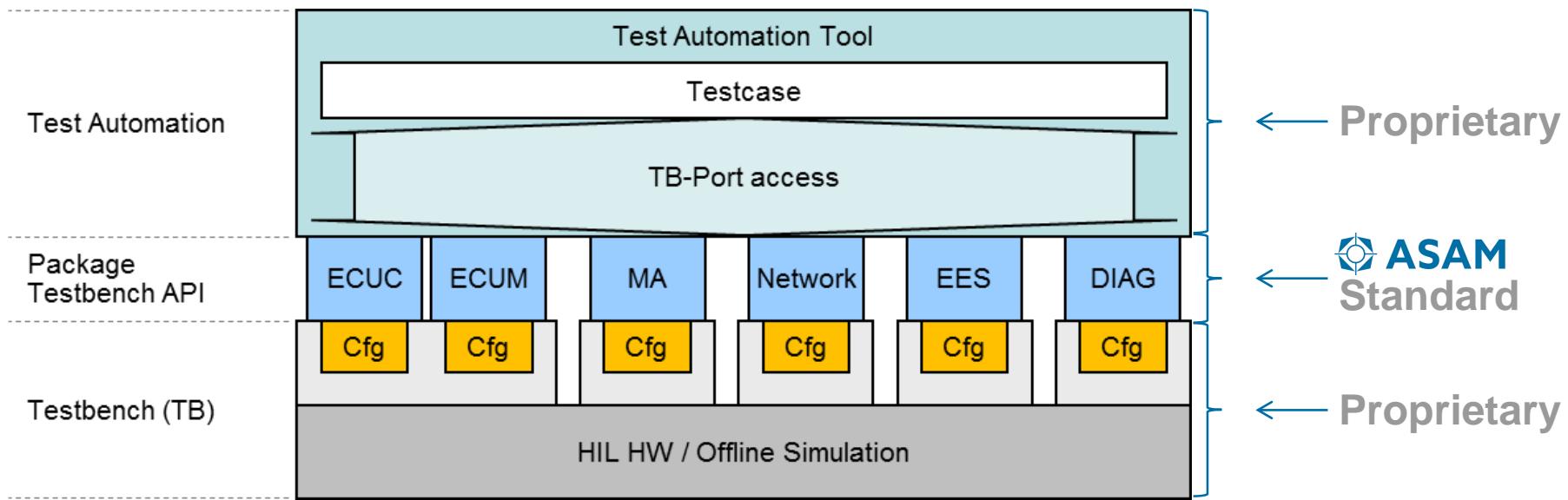
The ECUM port allows capturing and reading of **Measurement** variables. The ECUC port is used for **Calibration**.

DiagPort

Diagnostic port communicates with a diagnostic system, reads data via diagnostic services from an ECU.

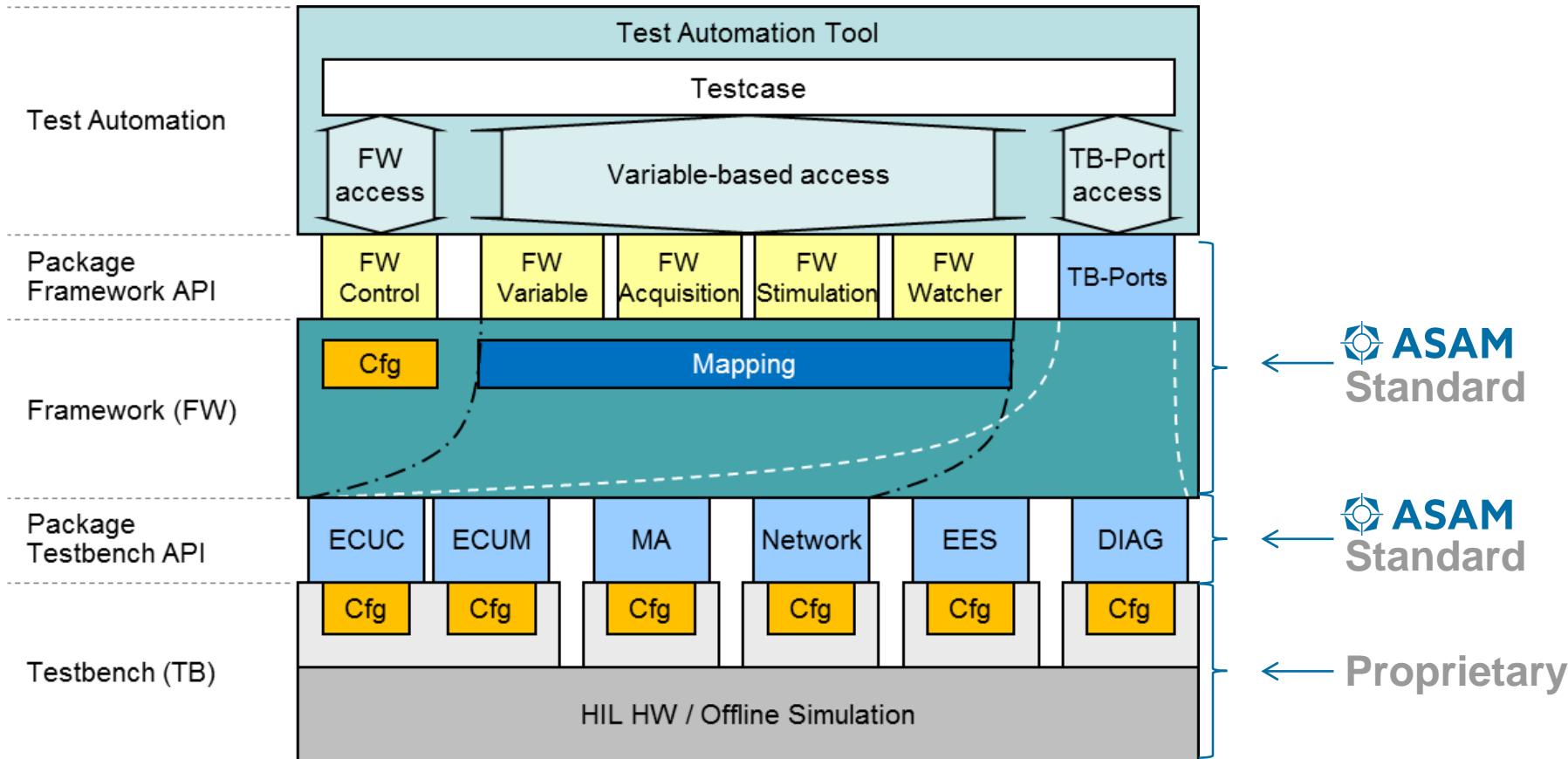


Port-Based Access Of A Test Automation Tool



Variable-Based Access Of A Test Automation Tool

- Port independence of test cases by using an object-oriented access to variables
- Test Developer can use both: Testbench Port access and variable based access by the Framework



Agenda

1 Introduction

2 Motivation for New Release

3 New Features

4 Other Changes

5 Backward-Compatibility

6 Deliverables

Motivation For ASAM XIL 2.2

- Address the feedback from cross tests in 2016 and 2017
- Incorporate the vendor feedback from the ASAM XIL 2.1 implementation
- Remove errors in the standard
- Close gaps in the specification about the expected behavior in certain areas
- Some new features were requested by the users (e.g. capturing support of vector and matrix data types)

Agenda

1 Introduction

2 Motivation for New Release

3 New Features

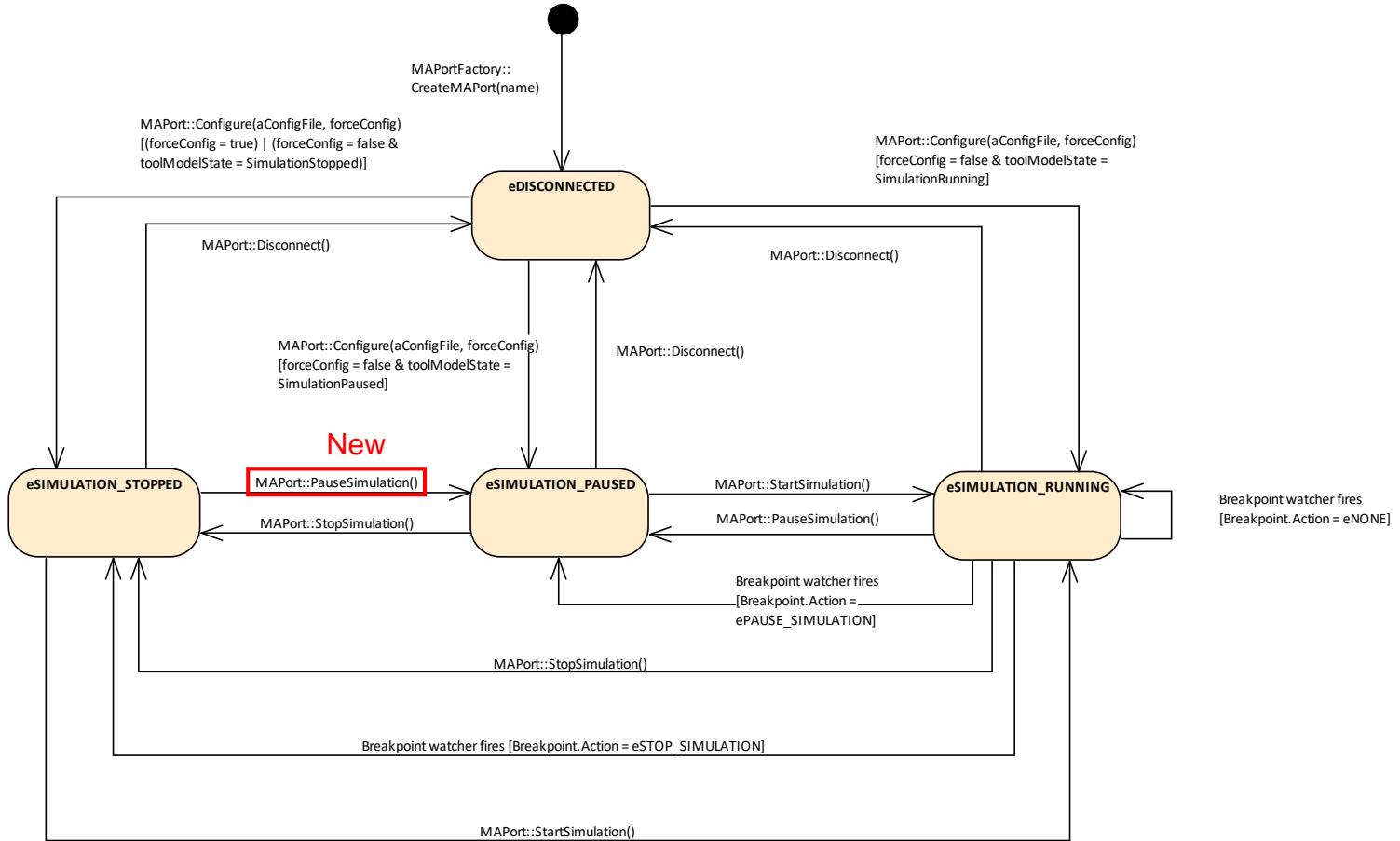
4 Other Changes

5 Backward-Compatibility

6 Deliverables

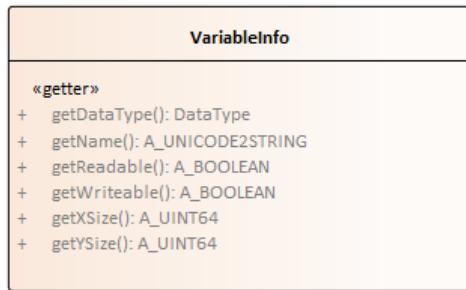
New State Transition Of The MAPort

- Additional state transition allows to pause the simulation at the beginning
- Especially useful for MIL and SIL use-cases

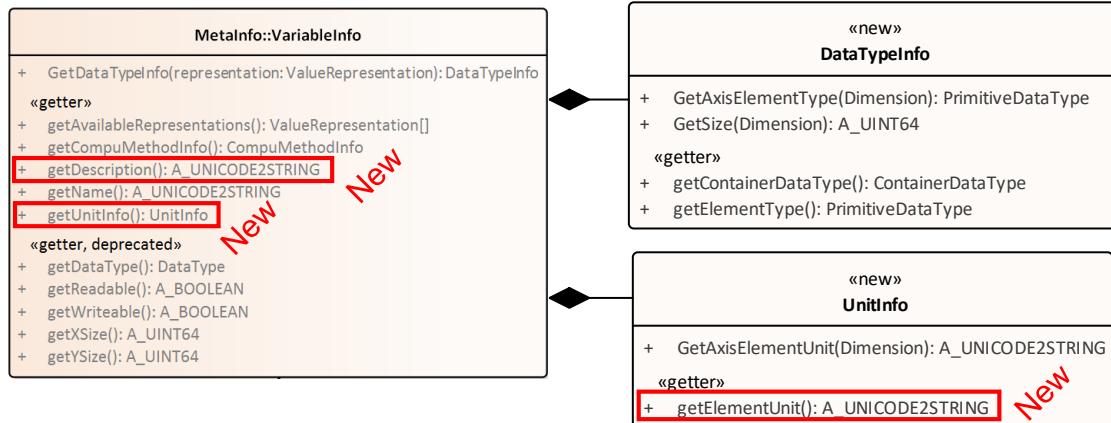


Extension Of The VariableInfo Interface 1/4

ASAM XIL 2.1



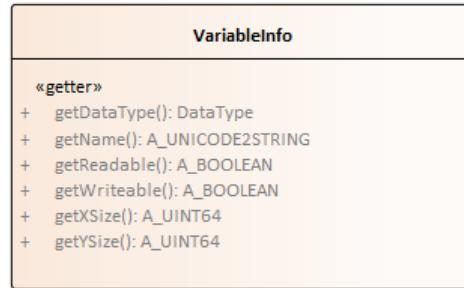
ASAM XIL 2.2



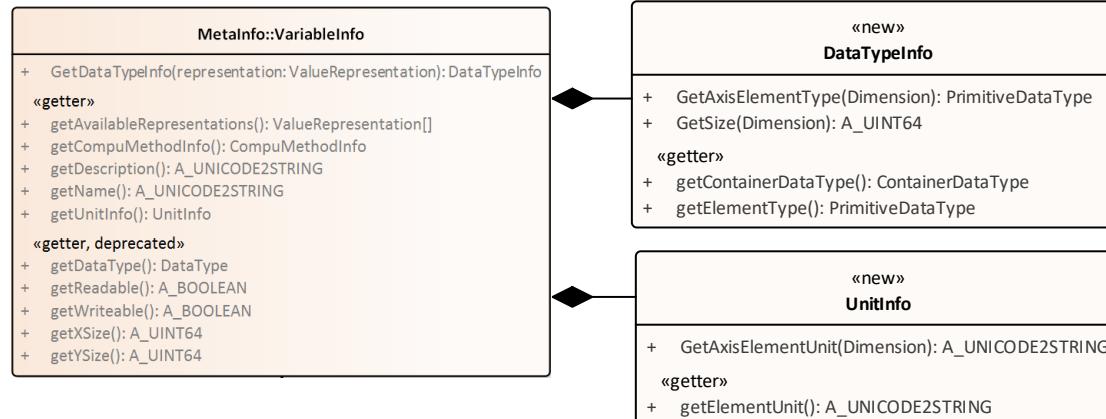
- The VariableInfo interface was not able to return the variable description and physical unit directly
- It tooks multiple function calls to get this information in ASAM XIL 2.1
- This information will now be available directly with the new getDescription() and getUnitInfo() function
- Improves performance and usability at the same time

Extension Of The VariableInfo Interface 2/4

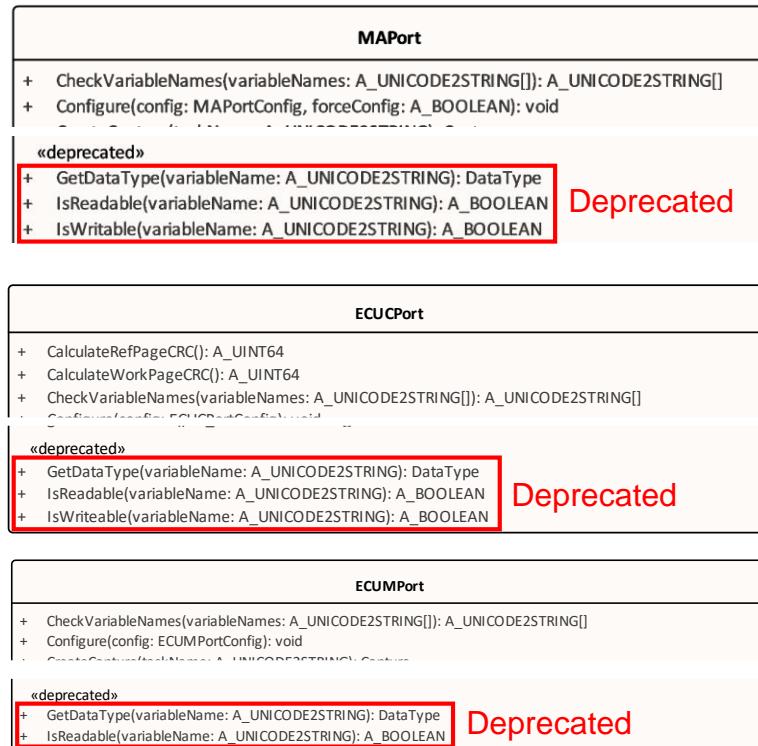
ASAM XIL 2.1



ASAM XIL 2.2

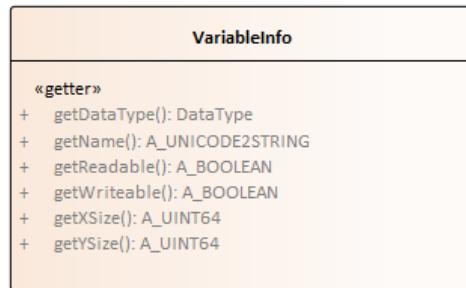


- The VariableInfo interface is now the unified location to get the variables data type and its read and writability.
- The port specific functions are not needed anymore



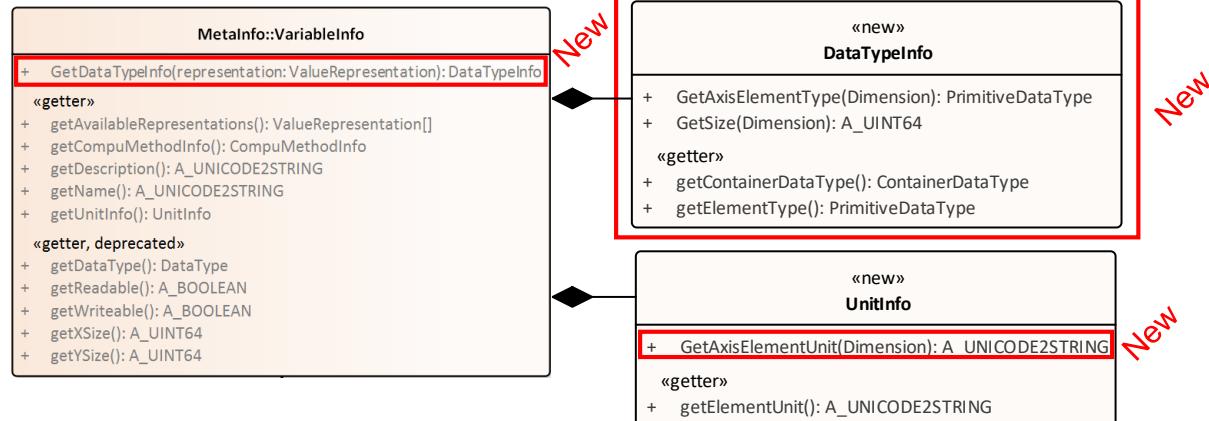
Extension Of The VariableInfo Interface 3/4

ASAM XIL 2.1



- The VariableInfo interface was also not able to retrieve any information on the element and axis types of curve, matrix and map variables
- The new GetTypeInfo() and GetAxisElementUnit() function provides a way to get this information of a complex variable

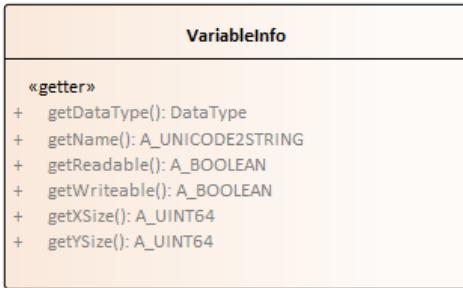
ASAM XIL 2.2



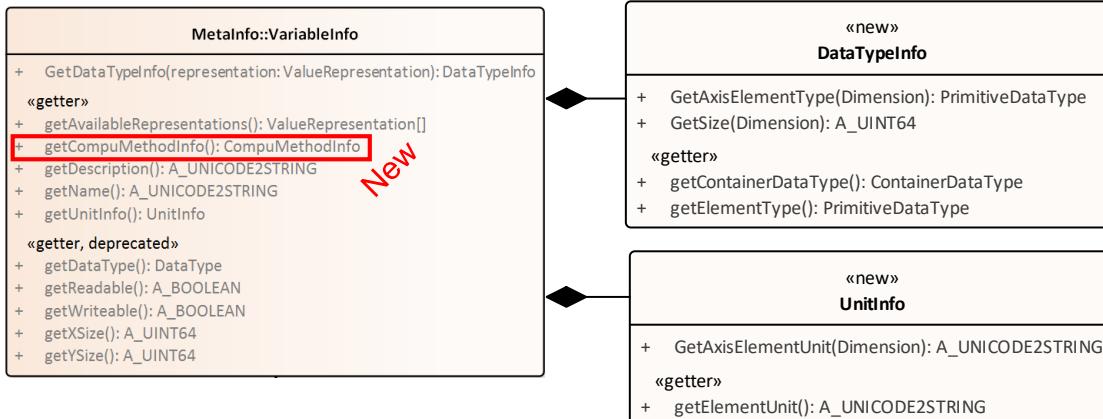
Extension Of The VariableInfo Interface 4/4

New

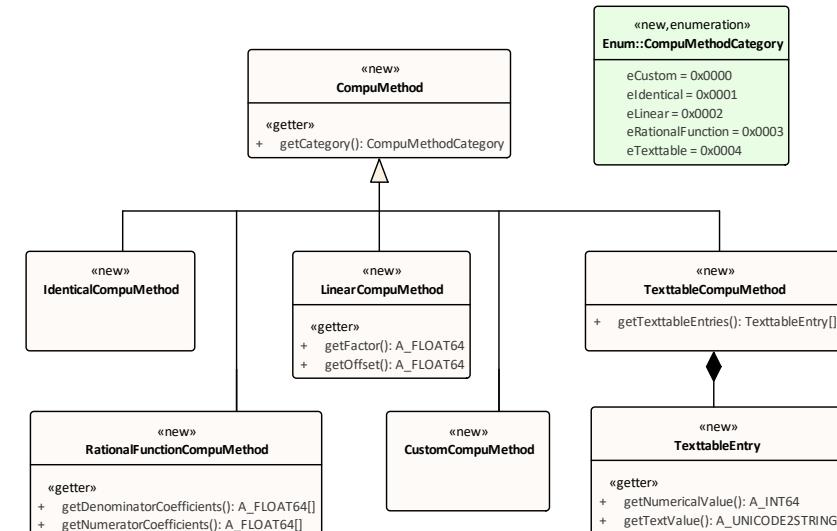
ASAM XIL 2.1



ASAM XIL 2.2



- The VariableInfo interface now distinguishes between physical value representation and raw value representation
 - It is now able to retrieve data type information on the physical value and the raw value of a testbench variable
 - It is now possible to retrieve information about the conversion method from raw value to physical value



As result of this change, the user can now choose whether to read and write data from a testbench variable as raw value or physical value

Discard Processed Capture Data

Capture
<pre>+ ClearConfiguration(): void + Fetch(whenFinished: A_BOOLEAN): CaptureResult + SetStartTrigger(watcher: Watcher, delay: Duration): void + SetStopTrigger(watcher: Watcher, delay: Duration): void + Start(writer: CaptureResultWriter): void + Stop(): void + TriggerClientEvent(eventId: A_UINT64, description: A_UNICODE2STRING): void <>getter</> + getCaptureResult(): CaptureResult + getDiscardFetchedData(): A_BOOLEAN New + getDownsampling(): A_UINT64 + getMinBufferSize(): A_INT64 + getPort(): Port + getRetriggering(): A_INT64 + getStartTriggerCount(): A_INT64 <>setter</> + setDiscardFetchedData(discardFetchedData: A_BOOLEAN): void New + setDownsampling(downSampling: A_UINT64): void + setMinBufferSize(minBufferSize: A_INT64): void + setRetriggering(retriggering: A_INT64): void</pre>

Goal

Let the user decide whether to keep or discard data already retrieved by `Fetch()`

- **DiscardFetchedData ==True**
After calling `Fetch()` the data that is returned is discarded and it is not going to be returned by `getCaptureResult()` after the capturing is done.
- **DiscardFetchedData==False**
Data returned by `Fetch()` is not discarded from memory and it will also be returned by `getCaptureResult()` after the capturing is done.
- This new option will help to avoid memory issues during long capturing sessions

Capture Support For Complex Data Types 1/2

CaptureResult	
+ Open(reader: CaptureResultReader): void	
+ Save(writer: CaptureResultWriter): void	
«deprecated»	
+ ExtractSignalValue(signalGroupName: A_UNICODE2STRING, variableName: A_UNICODE2STRING): SignalValue	Deprecated
+ GetSignalGroupValue(signalGroupName: A_UNICODE2STRING): SignalGroupValue	
+ GetVariableNames(signalGroupName: A_UNICODE2STRING): A_UNICODE2STRING[]	
«getter»	
+ getCaptureStartTime(): A_FLOAT64	
+ getEvents(): CaptureEvent[]	
+ getMetaData(): StringNamedCollection	
+ getSignalGroups(): CaptureSignalGroup[]	New
«deprecated, getter»	
+ getSignalGroupNames(): A_UNICODE2STRING[]	Deprecated
«setter»	
+ setMetaData(metaData: StringNamedCollection): void	

Goal

Support the capturing of complex data types

Capture Support For Complex Data Types 2/2

New

CaptureResult
+ Open(reader: CaptureResultReader): void + Save(writer: CaptureResultWriter): void «deprecated» + ExtractSignalValue(signalGroupName: A_UNICODE2STRING, variableName: A_UNICODE2STRING): SignalValue + GetSignalGroupValue(signalGroupName: A_UNICODE2STRING): SignalGroupValue + GetVariableNames(signalGroupName: A_UNICODE2STRING): A_UNICODE2STRING[] «getter» + getCaptureStartTime(): A_FLOAT64 + getEvents(): CaptureEvent[] + getMetaData(): StringNamedCollection + getSignalGroups(): CaptureSignalGroup[] «deprecated, getter» + getSignalGroupNames(): A_UNICODE2STRING[] «setter» + setMetaData(metaData: StringNamedCollection): void

New

CaptureSignalGroup
+ GetScalarSignalValues(variableRef: VariableRef): VectorValue + GetSignalValues(variableRef: VariableRef): BaseValue[] + GetSliceByIndexRange(startIndex: A_INT64, endIndex: A_INT64): CaptureSignalGroup + GetSliceByTimeRange(startTime: A_FLOAT64, endTime: A_FLOAT64): CaptureSignalGroup + GetTimestampDataType(representation: ValueRepresentation): PrimitiveDataType + GetTimestamps(representation: ValueRepresentation): VectorValue «getter» + getLength(): A_UINT64 + getName(): A_UNICODE2STRING + getSignalInfos(): VariableInfo[] + getTimestampCompuMethod(): CompuMethod + getTimestampUnit(): A_UNICODE2STRING

Goal

Support the capturing of complex data types

- The new `CaptureSignalGroup` interface is now the interface to retrieve data from `CaptureResult`
- The new interface can handle scalar values as well as complex structures like **vector** and **matrix**
- The `getSignalInfos` function in the `CaptureSignalGroup()` interface enables the user also to query metadata like data type, unit, CompuMethod and more
- The `CaptureResult` can return raw values as well as physical values

Signal Generation Support For Complex Data Types 1/2

SignalGenerator	
+ CheckConsistency():	ErrorInfo
+ Load(reader: SignalGeneratorReader):	void
+ Save(writer: SignalGeneratorWriter):	void
«getter»	
+ getAliasDefinitions():	VariableRefNamedCollection
+ getAssignments2():	StringByVariableRefCollection
+ getCustomProperties():	StringNamedCollection
+ getElapsedTime():	A_FLOAT64
+ getSignalDescriptionSet():	SignalDescriptionSet
«getter, deprecated»	
+ getAssignments():	StringNamedCollection
+ getState():	SignalGeneratorState
«setter»	
+ setAliasDefinitions(aliasDefinitions: VariableRefNamedCollection):	void
+ setAssignments2(assignments: StringByVariableRefCollection):	void
+ setCustomProperties(properties: StringNamedCollection):	void
+ setSignalDescriptionSet(value: SignalDescriptionSet):	void
«setter, deprecated»	
+ setAssignments(assignments: StringNamedCollection):	void
Deprecated	

- The `SignalGenerator` interface is now able to assign a `SignalDescription` of a `SignalDescriptionSet` to more than one variable

- ASAM XIL 2.1**

`SignalDescriptorSet1`

- `SignalDescription_1` → `Variable_1`
- `SignalDescription_2` → `Variable_2`
- `SignalDescription_3` → `Variable_3`

- ASAM XIL 2.2**

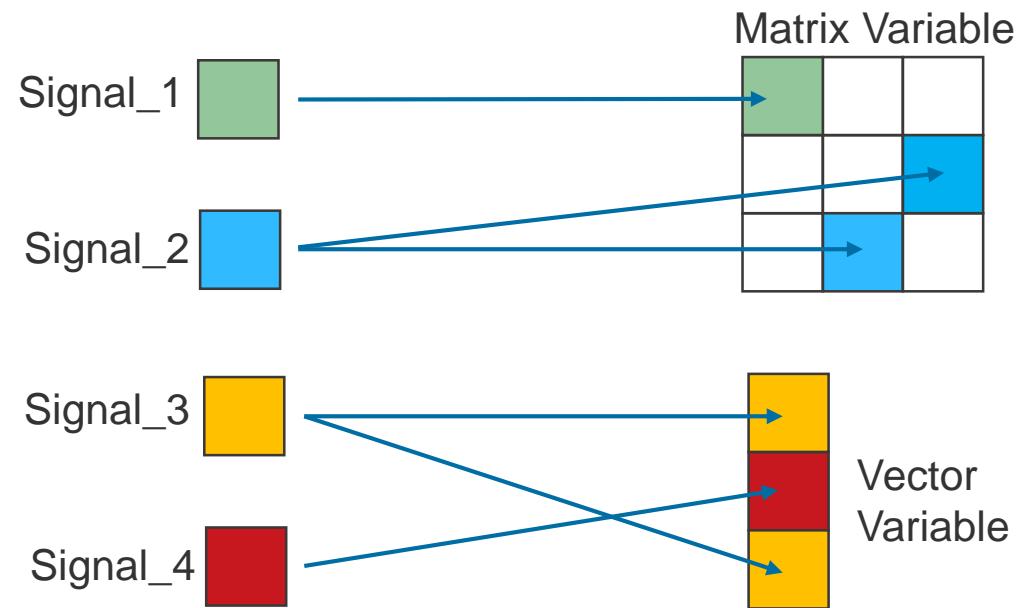
`SignalDescriptorSet1`

- `SignalDescription_1` → `Variable_1`
- `SignalDescription_2` → `Variable_2`
- `SignalDescription_3` → `Variable_3`
- `SignalDescription_3` → `Variable_4`
- `SignalDescription_3` → `Variable_5`

Signal Generation Support For Complex Data Types 2/2

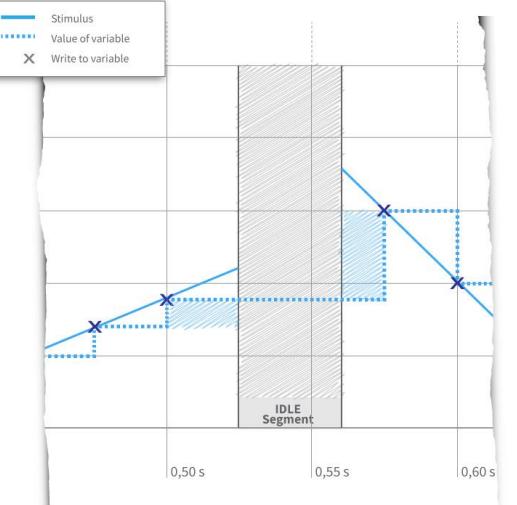
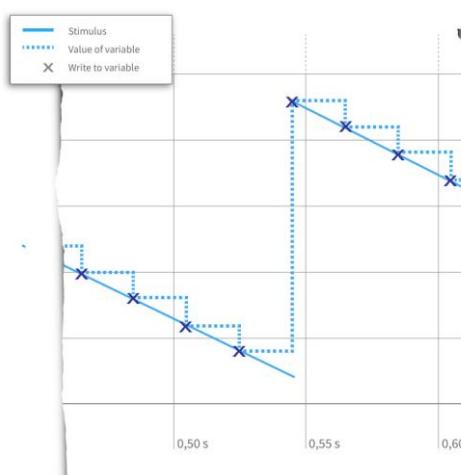
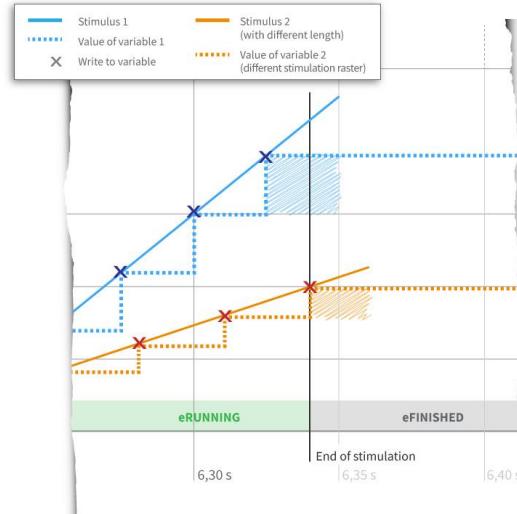
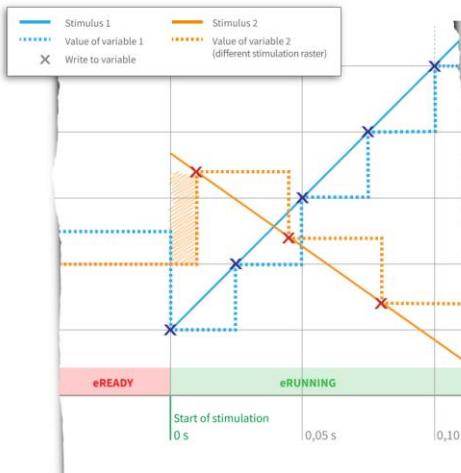
SignalGenerator	
+ CheckConsistency(): ErrorInfo	
+ Load(reader: SignalGeneratorReader): void	
+ Save(writer: SignalGeneratorWriter): void	
«getter»	
+ getAliasDefinitions(): VariableRefNamedCollection	New
+ getAssignments2(): StringByVariableRefCollection	New
+ getCustomProperties(): StringNamedCollection	
+ getElapsed Time(): A_FLOAT64	
+ getSignalDescriptionSet(): SignalDescriptionSet	
«getter, deprecated»	
+ getAssignments(): StringNamedCollection	Deprecated
+ getState(): SignalGeneratorState	Deprecated
«setter»	
+ setAliasDefinitions(aliasDefinitions: VariableRefNamedCollection): void	
+ setAssignments2(assignments: StringByVariableRefCollection): void	New
+ setCustomProperties(properties: StringNamedCollection): void	
+ setSignalDescriptionSet(value: SignalDescriptionSet): void	
«setter, deprecated»	
+ setAssignments(assignments: StringNamedCollection): void	Deprecated

- It's now also possible to stimulate single vector/matrix elements by signals in a SignalGenerator



- The SignalGenerator interface can now also stimulate variables with raw values or physical values

Clarification Of Signal Descriptions / Stimulation Behavior



- Extension of the ASAM XIL 2.2 Programmers Guide to clarify:
 - Stimulation behavior at the boundary between consecutive segments of a stimulus signal
 - Stimulation behavior of idle segments
 - Stimulation behavior at the end of the stimulus signal

Get All Installed Frameworks

New

FrameworkFactory

```
+ CreateVendorSpecificFramework(vendorName, productVersion): Framework  
+ CreateVendorSpecificFramework2(vendorName, productVersion, xilVersion): Framework  
+ GetAvailableFrameworks(): FrameworkInfo[] New
```

- It can be helpful to know all installed ASAM XIL Frameworks on a system
- The new `GetAvailableFrameworks()` function provides a list of all available ASAM XIL Frameworks

New

FrameworkInfo

```
«getter»  
+ getProductVersion(): A_UNICODE2STRING  
+ getSortableProductVersion(): A_UINT16[]  
+ getVendorName(): A_UNICODE2STRING  
+ getXILVersion(): A_UNICODE2STRING
```

Testbench/Framework With Explicitly Specified XIL Version

TestbenchFactory
+ CreateVendorSpecificTestbench(vendorName, productName, productVersion): Testbench
+ CreateVendorSpecificTestbench2(vendorName, productName, productVersion, xilVersion): Testbench
+ GetAvailableTestbenches(): TestbenchInfo[]

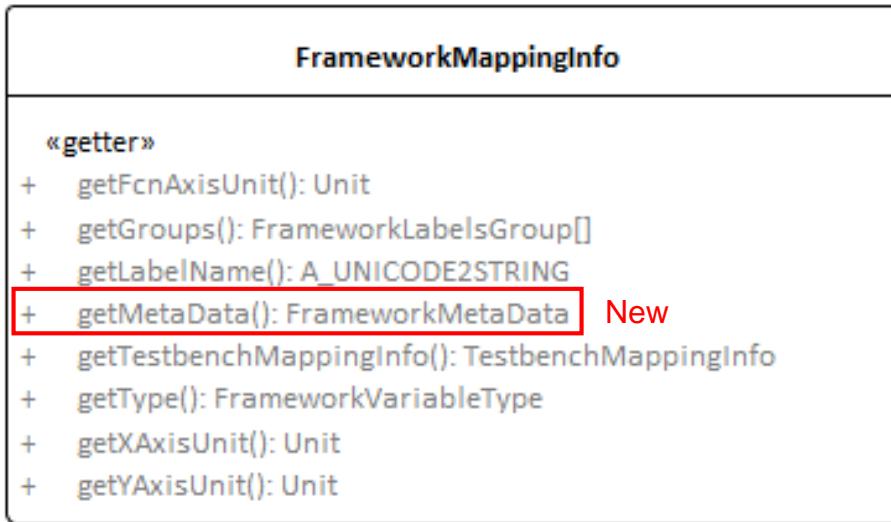
New

- The existing implementations of `CreateVendorSpecificTestbench()` and `CreateVendorSpecificFramework()` will choose the implementation whose XIL version equals the `TestbenchFactory`'s or `FrameworkFactory`'s XIL version
- New functions provide an extra parameter that allows the client to explicitly specify the XIL version of the `Testbench`/`Framework` implementation to be instantiated
- The already existing functions will continue to exist in parallel and are not obsolete
- The new functions are only intended for special use-cases

FrameworkFactory
+ CreateVendorSpecificFramework(vendorName, productVersion): Framework
+ CreateVendorSpecificFramework2(vendorName, productName, productVersion, xilVersion): Framework
+ GetAvailableFrameworks(): FrameworkInfo[]

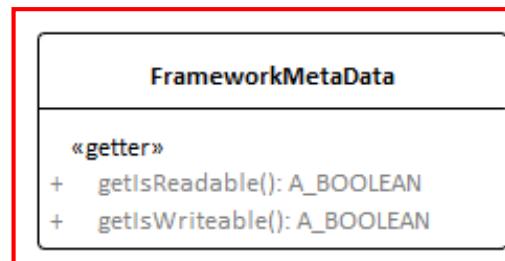
New

Expose Framework Metadata Information



- The `FrameworkMappingInfo` interface didn't provided the metadata information of the framework mapping
- The new `getMetaData()` function will now provide access to this data

New



Extension Of The Framework Configuration File 1/2

PortDefinition type	Permitted target states	Parameters
MAPortDefinition	eDISCONNECTED	---
	eSIMULATION_STOPPED	PortConfigurationFile
	eSIMULATION_RUNNING	New ForceConfig
DiagPortDefinition	eDISCONNECTED	---
	eCONNECTED	PortConfigurationFile
EESPortDefinition	eDISCONNECTED	---
	eCONNECTED	PortConfigurationFile
	eDOWNLOADED	PortConfigurationFile
	eACTIVATED	New ErrorConfigurationFile
ECUCPortDefinition	eDISCONNECTED	---
	eOFFLINE	PortConfigurationFile
	eONLINE	PortConfigurationFile New LoadingType
ECUMPortDefinition	eDISCONNECTED	---
	eMEASUREMENT_STOPPED	PortConfigurationFile
	eMEASUREMENT_RUNNING	
NetworkPortDefinition	eDISCONNECTED	---
	eSTOPPED	PortConfigurationFile
	eRUNNING	

- Modification of the framework configuration file to add new parameters
- New parameters enable the framework to drive all XIL ports to all states without user interaction or default values
- New Parameters
 - ForceConfig
 - ErrorConfigurationFile
 - LoadingType

Extension Of The Framework Configuration File 2/2

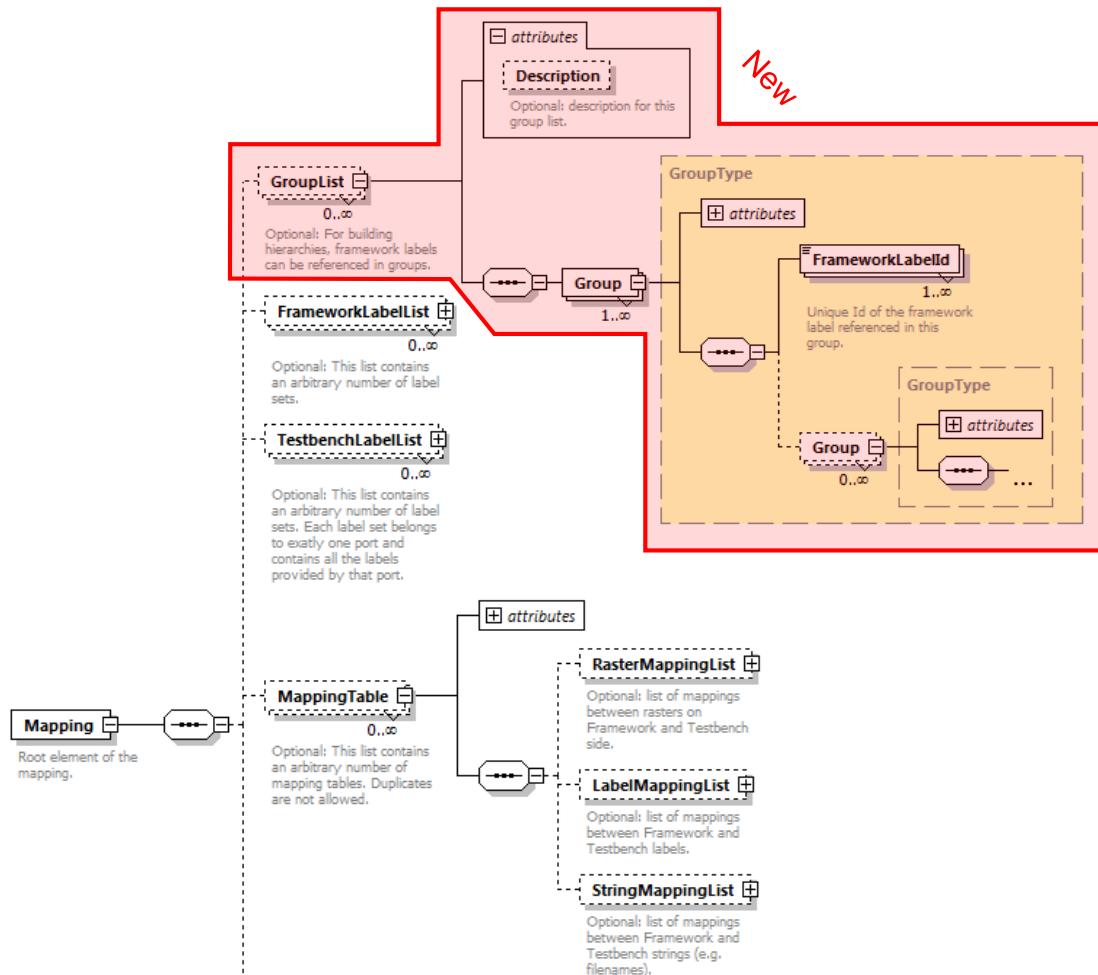
ASAM XIL 2.1

```
<MAPortDefinition InstanceName="MaPortRunning" InitOrder="2" ShutdownOrder="2" TargetState="eSIMULATION_RUNNING">
    <VendorName>VendorXy</VendorName>
    <ProductName>ProductAb</ProductName>
    <ProductVersion>20198-B.2</ProductVersion>
    <PortConfigurationFile>MAPortConfig.xml</PortConfigurationFile>
</MAPortDefinition>
```

ASAM XIL 2.2

```
<MAPortDefinition InstanceName="MaPortRunning" InitOrder="2" ShutdownOrder="2">
    <VendorName>VendorXy</VendorName>
    <ProductName>ProductAb</ProductName>
    <ProductVersion>2019-B.2</ProductVersion>
    <TargetState>
        <eSIMULATION_RUNNING>
            <PortConfigurationFile>MAPortConfig.xml</PortConfigurationFile>
            <ForceConfig>false</ForceConfig>
        </eSIMULATION_RUNNING>
    </TargetState>
</MAPortDefinition>
```

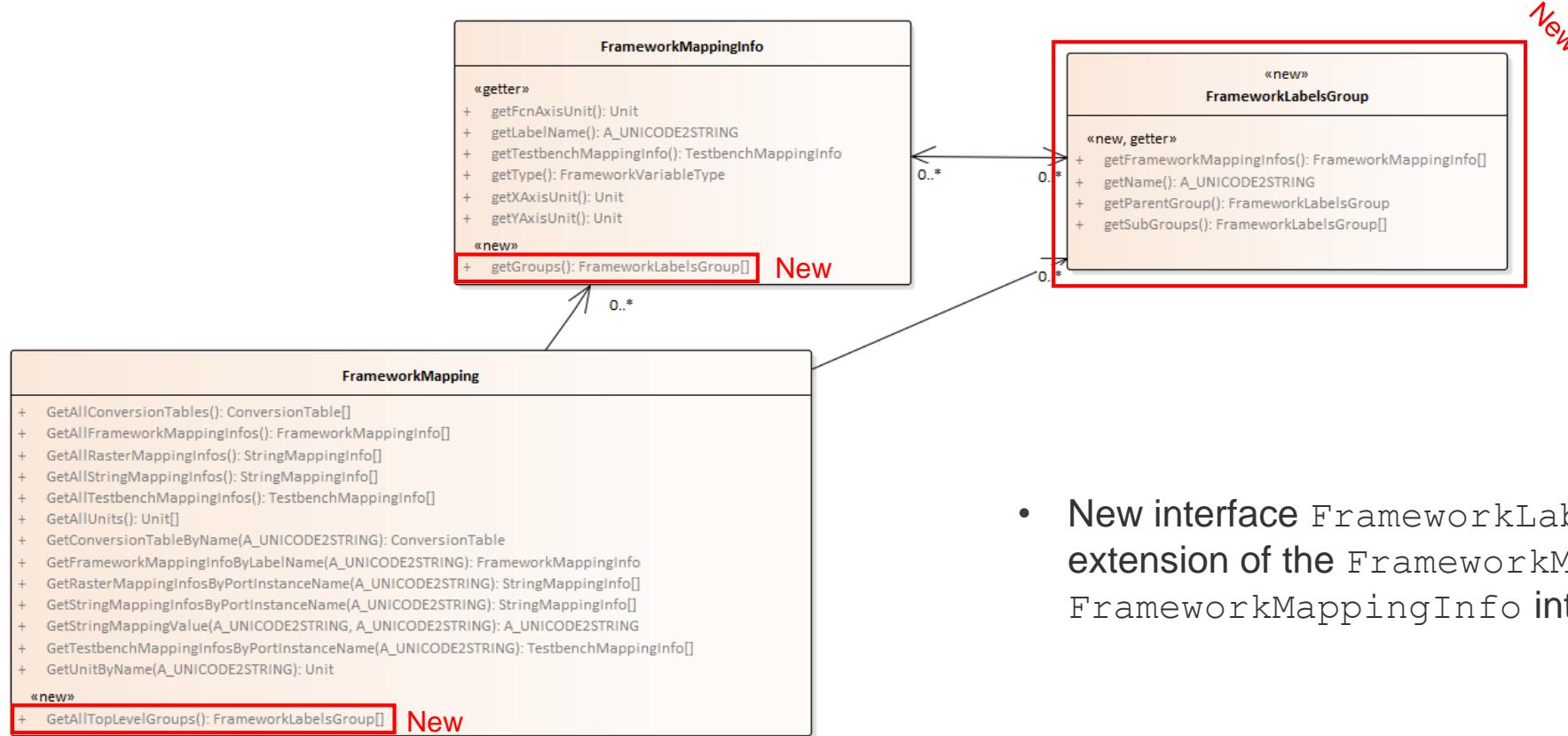
Grouping Of Framework Labels 1/2



- ASAM XIL Framework labels can now be grouped
- Offers a way to hierarchical structure framework labels
- Groups can contain a collection of subgroups
- No artificial limitations how many groups/subgroups and framework labels are used

Grouping Of Framework Labels 2/2

New



- New interface **FrameworkLabelsGroup** and extension of the **FrameworkMapping** and **FrameworkMappingInfo** interface for grouping

Agenda

1 Introduction

2 Motivation for New Release

3 New Features

4 Other Changes

5 Backward-Compatibility

6 Deliverables

Deprecation Marking In Source Code Files

ASAM XIL 2.1

```
/// <summary>
/// Creates a MAPortBreakpoint object.
/// </summary>
/// <param name="watcher">Breakpoint condition (regarding elapsed simulation time
/// or certain values of simulation variables) in form of a Watcher.</param>
/// <param name="action">Action to be automatically performed when simulation
/// reaches the breakpoint, i.e when the breakpoint condition is met.</param>
int CreateMAPortBreakpoint(IWatcher watcher, BreakpointAction action);
```

- Deprecated functions are now also marked in the source code files as deprecated

ASAM XIL 2.2

```
/// <summary>
/// Creates a MAPortBreakpoint object.
/// </summary>
/// <param name="watcher">Breakpoint condition (regarding elapsed simulation time
/// or certain values of simulation variables) in form of a Watcher.</param>
/// <param name="action">Action to be automatically performed when simulation
/// reaches the breakpoint, i.e when the breakpoint condition is met.</param>
[Obsolete("This method is deprecated and might be removed in a future version of
the standard. So XIL clients are strongly recommended to use the replacement
specified in the API documentation and the Programmer's Guide Appendix.", false)]
int CreateMAPortBreakpoint(IWatcher watcher, BreakpointAction action);
```

Agenda

1 Introduction

2 Motivation for New Release

3 New Features

4 Other Changes

5 Backward-Compatibility

6 Deliverables

Backward Compatibility

Affected Component	Deprecated Element	Replacement
Attributes (Package Testbench.Common.Value Container)	Complete interface	Interface VariableInfo and its Port specific derivations (e.g. MAPortVariableInfo from package Testbench.MAPort) for predefined Attributes "Name", "Description" and "Unit". No replacement for user defined Attributes.
BaseValue (Package Testbench.Common.Value Container)	Property Attributes	Method GetVariableInfo of the Port interfaces for predefined Attributes "Name", "Description" and "Unit". No replacement for user defined Attributes.
	Property Type	Properties ContainerType and ElementType
Capture (Package Testbench.Common.Capturing)	Property DurationUnit	Implicitly set by passing a TimeSpanDuration or CycleNumberDuration object (from package Testbench.Common.Duration) to SetStartTrigger , SetStopTrigger and the DurationWatcher factory methods.
	Method SetStartTriggerCondition	Method SetStartTrigger
	Method SetStopTriggerCondition	Method SetStopTrigger
	Property Variables	Property Variables2
CaptureResult (Package Testbench.Common.CaptureResult)	Method ExtractSignalValue	Methods GetTimestamps , GetSignalValues and GetScalarSignalValues of the CaptureSignalGroup interface obtainable via the SignalGroups property
	Method GetSignalGroupValue	Property SignalGroups

- Deprecated elements are listed in Appendix F. – Deprecated Elements of the ASAM XIL 2.2 Programmers Guide
- The API elements listed in the second column of the table are deprecated and might be removed in a future version of the standard.
- So it is recommended to use the replacement depicted in the third column.

Agenda

1 Introduction

2 Motivation for New Release

3 New Features

4 Other Changes

5 Backward-Compatibility

6 Deliverables

Deliverable Content

Documents

- Content Overview
- Generic UML Model (Enterprise Architect)
- Programmers Guide

Supplementary Files

- Schemas (incl. examples)
 - EES_Configuration
 - FrameworkConfiguration
 - ImplementationManifest
 - Mapping
 - StimulusSignalDescription
- Technology Reference for C#
 - C# Interfaces
 - C# Examples
 - **C# Test Suite**
- Technology Reference for Python
 - Python Interfaces
- XIL Standard Assemblies (MSI Installer)

ASAM TestSuite for EESPort und MAPort

New