



December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA

CIM/SMI-S Provider Generator Tools

Martine Wedlake

&

Todd Bates

IBM



Agenda

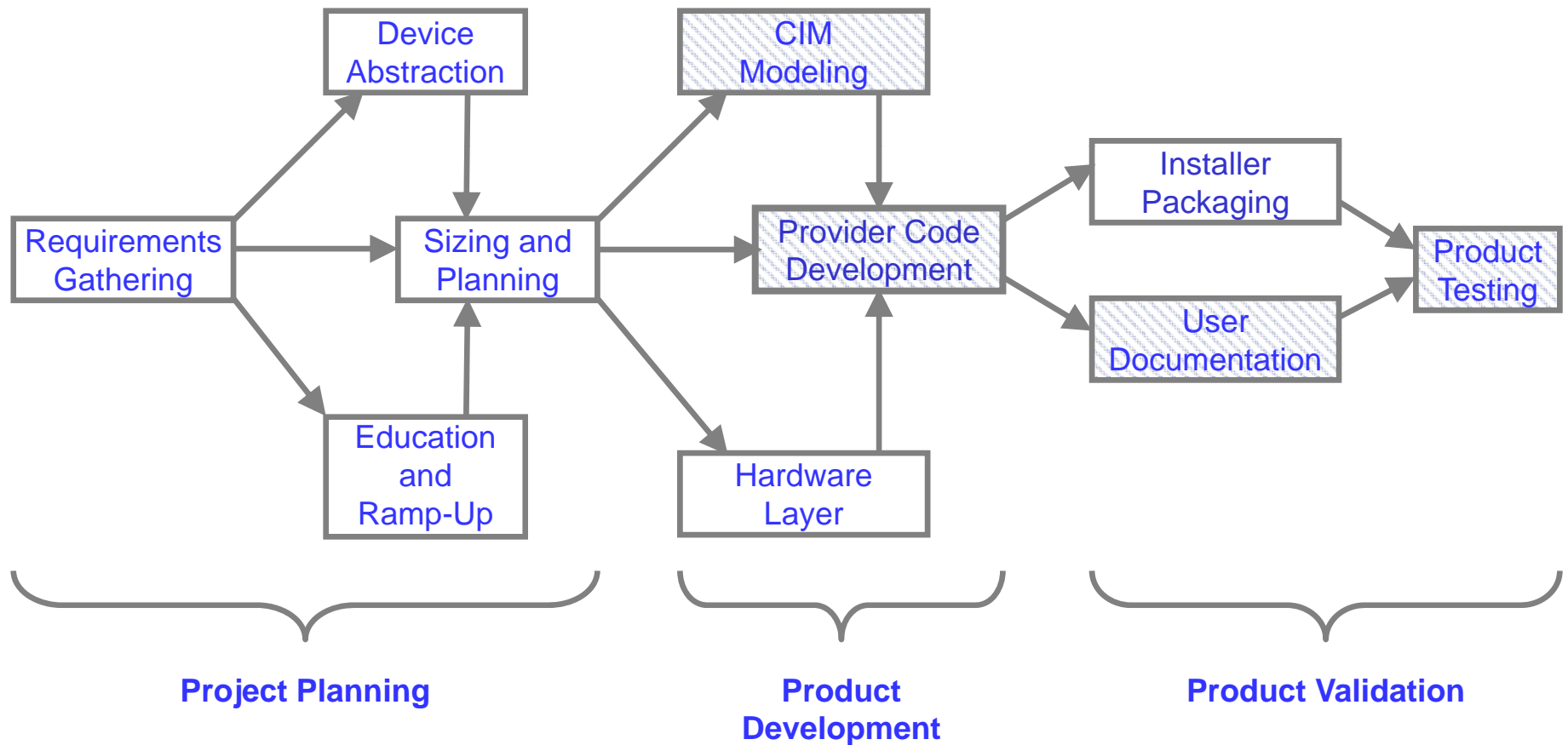
- Motivation
- Provider Development Workflow
 - UML Modeling & Product Validation
 - Provider Code and Documentation Generation
- Questions




Motivation

- Because CIM/SMI-S is very complex...
 - Complex object models (DMTF, SMI-S),
 - Complex APIs (CMPI, NPI),
 - Complex development environments (Client-Server, Indications)
- ...we see higher development costs (specialized CIM knowledge required) and more product defects.
- *Our motivation* is to increase product quality with lower costs by using tools that simplify the development environment and require less specialized CIM knowledge

Provider Development Workflow



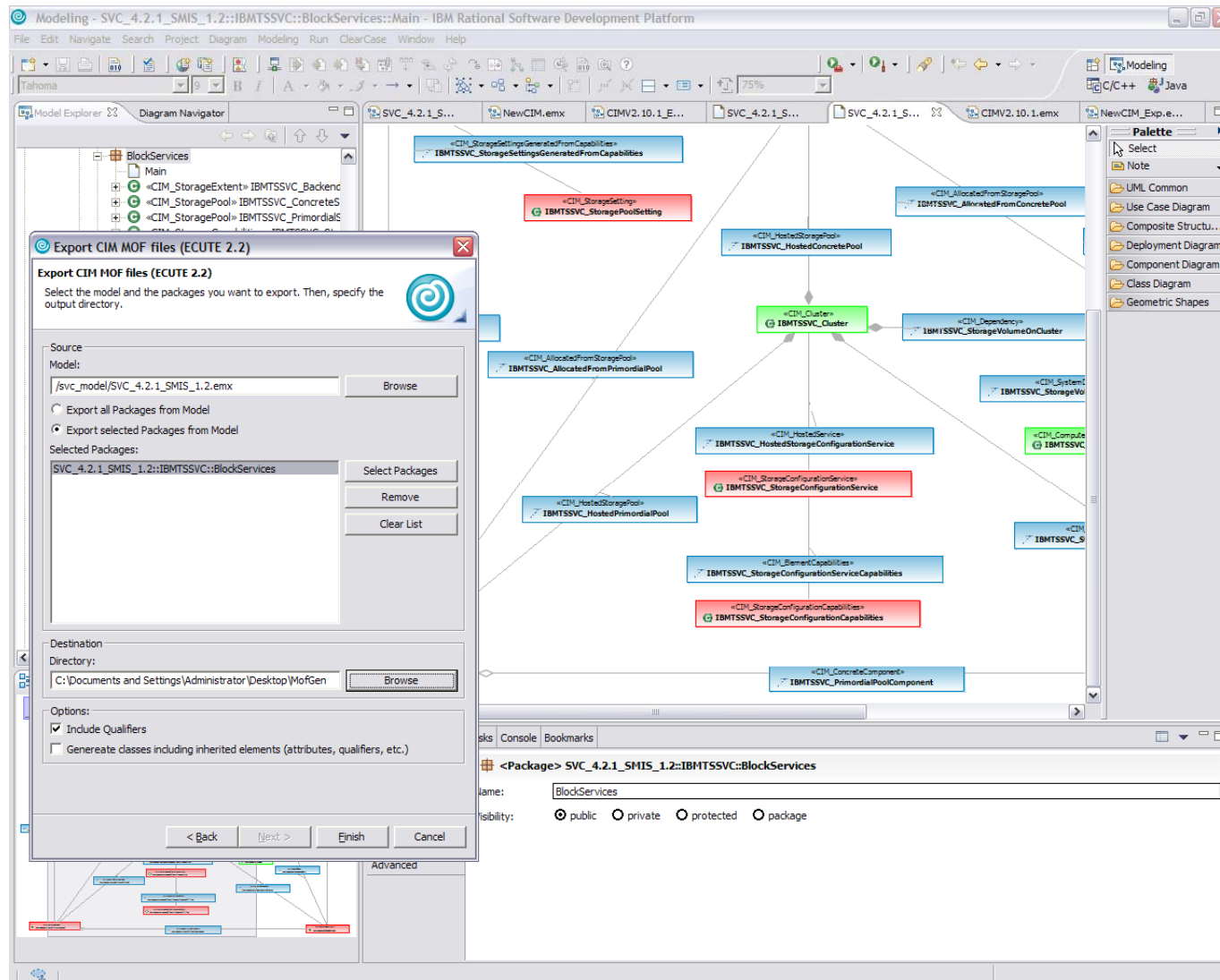
 Shaded items are discussed within talk



CIM Modeling & Product Validation (1/4)

- **ECUTE** (Extensible CIM and UML Tooling Environment)
 - <http://sblim.wiki.sourceforge.net/Ecute>
 - Suite of tools
- **ECUTE Graphical UML Editor**
 - Imports and exports MOF
 - Version 2.2 requires Rational Software Architect 6
 - Version 3.0 re-written for Eclipse RCP framework

CIM Modeling & Product Validation (2/4)



The screenshot displays the IBM Rational Software Development Platform interface. The main window shows a UML class diagram for a CIM model. The diagram includes several classes and their relationships:

- IBMTSSVC_StorageSettingsGeneratedFromCapabilities** (blue box)
- IBMTSSVC_StoragePoolSetting** (red box)
- IBMTSSVC_AllocatedFromStoragePool** (blue box)
- IBMTSSVC_AllocatedFromPrimordialPool** (blue box)
- IBMTSSVC_HostedStoragePool** (blue box)
- IBMTSSVC_HostedConcretePool** (blue box)
- IBMTSSVC_Cluster** (green box)
- IBMTSSVC_StorageVolumeOnCluster** (blue box)
- IBMTSSVC_System** (blue box)
- IBMTSSVC_StorageVolume** (blue box)
- IBMTSSVC_HostedStorageConfigurationService** (blue box)
- IBMTSSVC_StorageConfigurationService** (red box)
- IBMTSSVC_StorageConfigurationServiceCapabilities** (blue box)
- IBMTSSVC_StorageConfigurationCapabilities** (red box)
- IBMTSSVC_ConcreteComponent** (blue box)
- IBMTSSVC_PrimordialPoolComponent** (blue box)

An "Export CIM MOF files (ECUTE 2.2)" dialog box is open in the foreground. The dialog contains the following fields and options:

- Source Model:** /svc_model/SVC_4.2.1_SMIS_1.2.emx
- Export Options:**
 - Export all Packages from Model
 - Export selected Packages from Model
- Selected Packages:** SVC_4.2.1_SMIS_1.2::IBMTSSVC::BlockServices
- Destination Directory:** C:\Documents and Settings\Administrator\Desktop\MofGen
- Options:**
 - Include Qualifiers
 - Generate classes including inherited elements (attributes, qualifiers, etc.)

The bottom of the screenshot shows the Package Explorer with the package **SVC_4.2.1_SMIS_1.2::IBMTSSVC::BlockServices** selected, and the Package Properties dialog box showing the package name and visibility set to **public**.



CIM Modeling & Product Validation (3/4)

- ECUTE CIM-XML Analyzer
 - Sits between CIM Client and CIMOM and records incoming/outgoing messages
 - Supports HTTP and HTTPS
 - Basic validation of CIM-XML
 - Can record traffic into data file
- ECUTE CIM Explorer
 - Navigates multiple CIMOMs and namespaces in parallel
 - Discovers CIMOMs via SLP or by manual entry
 - Indication listener provided to subscribe and receive indications which can be recorded to a file



CIM Modeling & Product Validation (4/4)

The screenshot displays the ECUTE (Enterprise Configuration User Tool) interface, which is used for modeling and validating CIM (Common Information Model) data. The interface is divided into several panes:

- Discovery View:** A tree view on the left showing a hierarchy of CIM classes and instances, such as `IBMTSDS_StorageSystem.ConformToProfile`, `IBMTSDS_StorageSystem.SoftwareIdentity`, and `IBMTSDS_Volume`.
- Details:** A central pane showing the details of a selected CIM instance. It includes a table of key attributes and a list of associations. The table below shows the key attributes for `IBMTSDS_Volume`.
- Request/Response View:** A pane on the right showing the raw HTTP request and response for a specific operation. The response is an XML document representing the CIM instance data.

Key Attributes	Name	Type	Value
Access	Access	uint16	
AdditionalAvailability	AdditionalAvailability	uint16	
Availability	Availability	uint16	712140
BlockSize	BlockSize	uint64	
Caption	Caption	string	
ConsumableBlocks	ConsumableBlocks	uint64	2226
DataOrganization	DataOrganization	uint16	4
DataRedundancy	DataRedundancy	uint16	1
DeltaReservation	DeltaReservation	uint8	100
Description	Description	string	
ElementName	ElementName	string	V2
EnabledByDefault	EnabledByDefault	uint16	2
EnabledState	EnabledState	uint16	5
ErrorCleared	ErrorCleared	boolean	
ErrorDescription	ErrorDescription	string	
ErrorMethodology	ErrorMethodology	string	
EnterInStatus	EnterInStatus	uint16	2
HealthState	HealthState	uint16	5
IdentifyingDescriptions	IdentifyingDescriptions	string	
InstantDate	InstantDate	datetime	
IsBasedOnUnderlyingRedundancy	IsBasedOnUnderlyingRedundancy	boolean	false
LastErrorCode	LastErrorCode	uint32	
LSS	LSS	string	11
MaxQueueTime	MaxQueueTime	uint64	
Name	Name	string	IBM.1750-1100VC3-110
NameFormat	NameFormat	uint16	1
NameMapping	NameMapping	uint16	0
NoSinglePointOfFailure	NoSinglePointOfFailure	boolean	true
NumberOfBlocks	NumberOfBlocks	uint64	2226
OperationalStatus	OperationalStatus	uint16	2
OtherEnabledState	OtherEnabledState	string	
OtherIdentifyingInfo	OtherIdentifyingInfo	string	
OtherNameFormat	OtherNameFormat	string	
OtherNameNamespace	OtherNameNamespace	string	
PackageRedundancy	PackageRedundancy	uint16	1
PowerManagementCapabilities	PowerManagementCapabilities	uint16	
PowerManagementSupported	PowerManagementSupported	boolean	
PowerOnHours	PowerOnHours	uint64	
Proprietary	Proprietary	boolean	false
Purpose	Purpose	string	
RequestedState	RequestedState	uint16	12
SequentialAccess	SequentialAccess	boolean	

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset="utf-8"
Transfer-Encoding: chunked
Content-Operation: MethodResponse
Trailer: Content-Operation, Content-Language

<?xml version="1.0" encoding="utf-8" ?>
<CIM-CLASS-NAME="IBMTSDS_Volume" >
  <MESSAGE ID="303774" PROTOCOLVERSION="1.0" >
    <PARAMETERS >
      <PARAMETER NAME="Associators" >
        <OBJECTPATH >
          <INSTANCES >
            <INSTANCE CLASSNAME="IBMTSDS_StorageSystem" >
              <KEYVALUE NAME="CreationClassName" >IBMTSDS_StorageSystem</KEYVALUE >
              <KEYVALUE NAME="Name" >IBM.1750-1100VC3-110</KEYVALUE >
              <INSTANCES >
                <INSTANCE CLASSNAME="IBMTSDS_Volume" >
                  <KEYVALUE NAME="CreationClassName" >IBMTSDS_Volume</KEYVALUE >
                  <KEYVALUE NAME="Name" >IBM.1750-1100VC3-110</KEYVALUE >
                  <INSTANCES >
                    <INSTANCE CLASSNAME="IBMTSDS_StorageSystem" >
                      <KEYVALUE NAME="CreationClassName" >IBMTSDS_StorageSystem</KEYVALUE >
                      <KEYVALUE NAME="Name" >IBM.1750-1100VC3-110</KEYVALUE >
                      <KEYVALUE NAME="OperationalStatus" >2</KEYVALUE >
                      <KEYVALUE NAME="PowerManagementCapabilities" >0</KEYVALUE >
                      <KEYVALUE NAME="PowerManagementSupported" >false</KEYVALUE >
                      <KEYVALUE NAME="PowerOnHours" >0</KEYVALUE >
                      <KEYVALUE NAME="Proprietary" >false</KEYVALUE >
                      <KEYVALUE NAME="Purpose" ></KEYVALUE >
                      <KEYVALUE NAME="RequestedState" >12</KEYVALUE >
                      <KEYVALUE NAME="SequentialAccess" >false</KEYVALUE >
                    </INSTANCE >
                  </INSTANCES >
                </INSTANCE >
              </INSTANCES >
            </INSTANCE >
          </OBJECTPATH >
        </PARAMETER >
      </PARAMETERS >
    </MESSAGE >
  </CIM-CLASS-NAME >
```

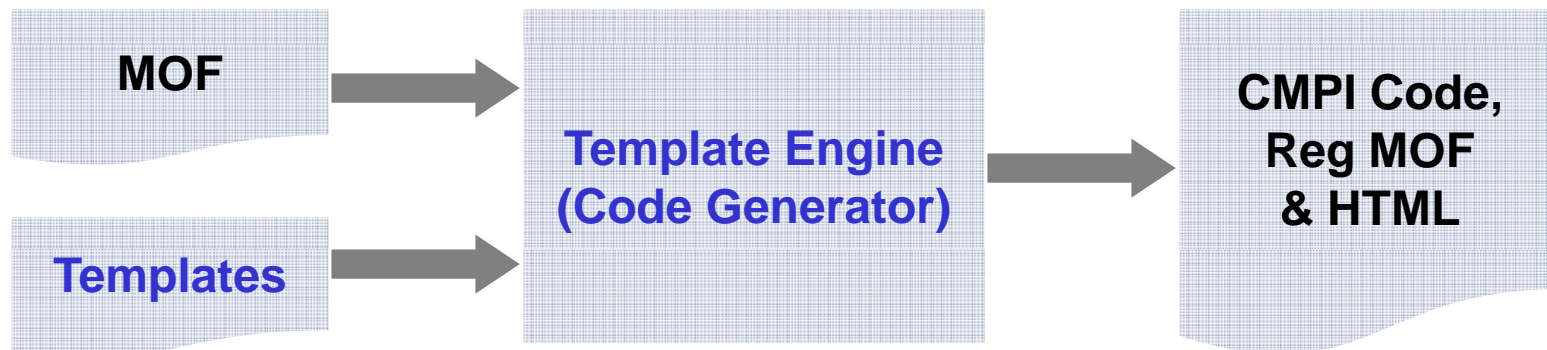


Provider Code/Doc Generation – Overview

- Purpose
 - Generates CMPI Compatible Code for:
 - Instance query (enum, get, names, etc)
 - Extrinsic method Invocation with concrete parameters
 - Association intrinsics (associators, references, names)
 - Pegasus registration MOF
 - Provider stubs
 - HTML documentation
 - Logging and tracing
- Rationale
 - Low memory usage via writer (streaming) interface
 - Simplify first, then optimise
 - Type check enforcement improves code quality
 - Central place for performance improvement

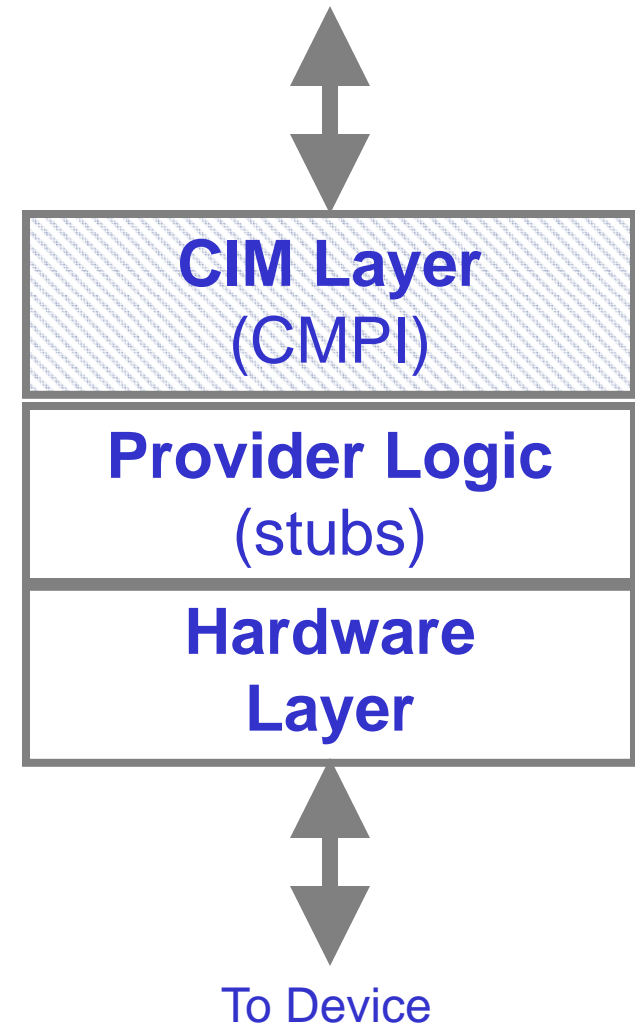
Provider Code/Doc Generation – Design (1/7)

- MOF Compiler/Parser
 - Reads the MOF and makes it available to the Velocity template engine
- Template Engine
 - Drives the auto-generation of code
- Utility Classes
 - Supports the auto-generated code; e.g., data-type conversions, logging/tracing, etc.



Provider Code/Doc Generation – Design (2/7)

- Development Model
 - **CIM Layer** – Communicates with CMPI
 - Automatically generated
 - **Provider Logic** – Main Body of Provider Function
 - Generally developed by hand
 - Automatically generated stubs are edited by developers
 - **Hardware Layer** – Communicates with Device
 - Generally project specific





Provider Code/Doc Generation – Design (3/7)

- **CIM Layer**
 - Maps between CMPI and Provider Logic interfaces
 - Error checking
 - Datatype checking
 - Extrinsic method parameter names
 - Required/Optional extrinsic method parameter
 - Logging and tracing
 - Creates helper functions for Provider Logic
 - Writer Interface



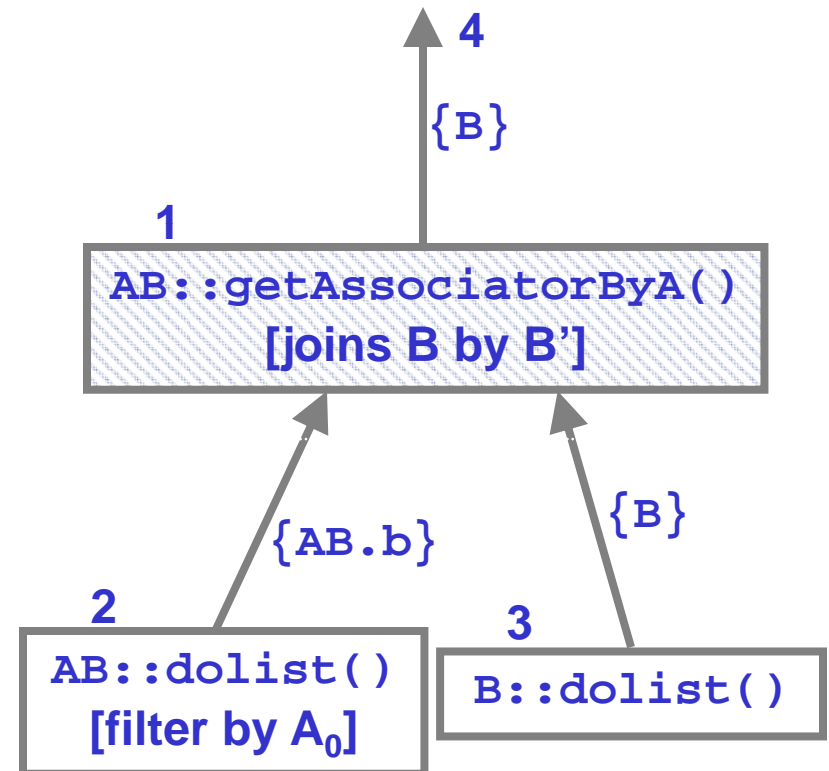
Provider Code/Doc Generation – Design (4/7)

- **Provider Logic**
 - Writer (streaming) Interface
 - Dataflow model
 - Provider Logic writes instances into stream
 - Stream has filters and join operations to perform required actions (e.g., property list filtering, `getInstance`, `getInstanceNames`, etc.)
 - Drive filters down to provider logic to improve performance – however not required, they act as hints

Provider Code/Doc Generation – Design (5/7)

- **Dataflow Example:**

- Class A and B
- Association AB
w/references A::a &
B::b
- Request:
Associators(A₀)





Provider Code/Doc Generation – Design (6/7)

- **Concretization**

- Method signatures

- Keys are pulled from the `CMPIObjectPath` and passed into the Provider Logic
 - Input parameters are pulled from `inArgs` in CMPI and passed into Provider Logic
 - Output parameters are pushed into `outArgs` in CMPI and passed into Provider Logic as `&outParamName`

- Data types (arrays to vectors, etc)

- All CIM datatypes are represented as C++ datatypes (including arrays)

- References

- `CMPIObjectPath` objects are represented as `Ref` objects with typed keys.



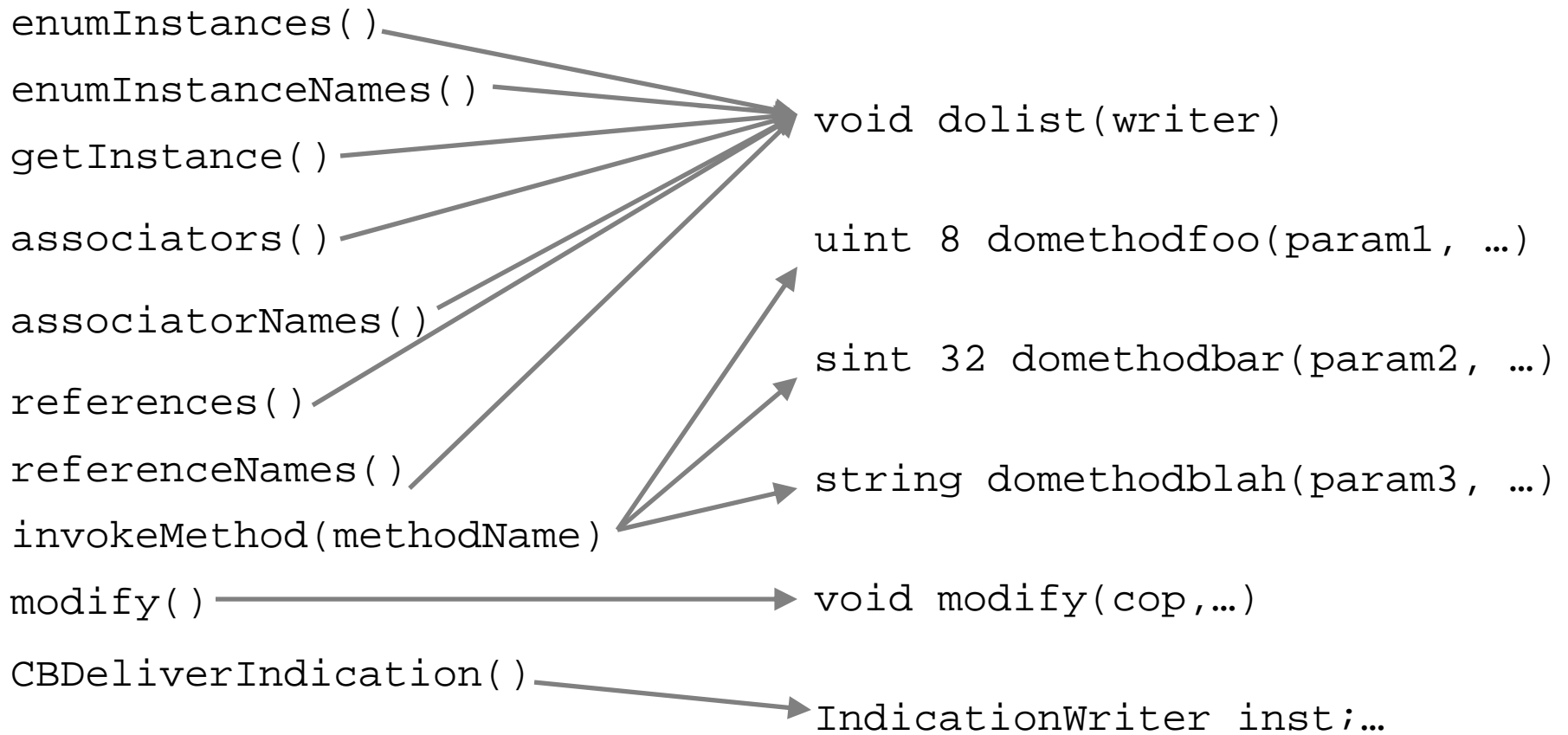
Provider Code/Doc Generation – Design (7/7)

CIM Layer

```
enumInstances()  
enumInstanceNames()  
getInstance()  
associators()  
associatorNames()  
references()  
referenceNames()  
invokeMethod(methodName)  
modify()  
CBDeliverIndication()
```

Provider Logic Layer

```
void doList(writer)  
  
uint 8 doMethodfoo(param1, ...)  
sint 32 doMethodbar(param2, ...)  
string doMethodblah(param3, ...)  
  
void modify(cop,...)  
  
IndicationWriter inst;...
```





Provider Code/Doc Generation – Example

MOF

```
class Sample_Test
{
    [key] uint8 key;
    uint8     foo;
    string    bar
    uint8     testMethod([in] uint8 Arg1, [out] uint8 Arg2);
};
```

Provider Logic Implementation

```
void provider::dolist(Provider::ListWriter &out) {
    for (uint8 i = 0; i < 100; i++) {
        Ref ref = Ref(i);
        Provider::ListWriter::Item item(out,ref);
        item.writeFoo(data[i])
        item.writeBar("Hello, World!");
    }
}

uint8 Provider::dotestMethod(uint8 keyKey, uint8 inArg1, uint8 &outArg1){
    outArg1 = inArg1+1;
    return inArg1;
}
```



Provider Code/Doc Generation – Optimizations

- Glue Code Optimizations
 - Provider Logic Filtering
 - `ListWriter.needsMoreItems()` used to determine if the listwriter needs more instances
 - `ListWriter.getFilter()` returns a concrete class with filter settings; allows provider logic layer to prepare for a hwlayer query with the right structure
 - `Item.is<property>Required()` checks if a given property is required; similar to the `getFilter()` approach, but for a given item
 - `Item.isDone()` used to see if any more properties are required for this given item
 - Association Override
 - The developer can override the standard association implementation by providing a “GetBy” function
 - Get One Override
 - Sometimes the hardware layer has a quick mechanism to access specific instances; so instead of enumerating all instances and returning when it’s found, the CIM layer (glue code) can call into the Provider Logic with additional interface to retrieve the specific instance



Provider Code/Doc Generation – Miscellaneous

- User Templates
 - Simple way to augment CIM Layer code
 - `headers.vm` is a velocity template that is inserted at the top of all glue code
 - good for copyright notices, etc.
 - `preamble.vm` and `postamble.vm` are velocity templates inserted at the top and bottom of each glue function
 - Can be used to establish pre- and post-conditions for the provider logic (e.g., sockets, etc.)
 - Exception handling



Questions?



Backup Slides

Example 1/4

- Indication Delivery

```
unit_instcreation::Provider::IndicationWriter writer;  
writer.start(unit_instcreation::Ref());  
unit_datatypes::Ref ref(keyVal);  
unit_datatypes::EmbeddedObjectItem  
    embedditem(writer.embeddedSourceInstance, ref);  
embedditem.writeValue( value );  
  
...  
writer.writeSourceInstanceModelPath(ref.toString());  
writer.send();
```

■ Association Override

```
void Provider::doAssocByA(const unit_assoca::Ref &inA,
    Provider::ListWriter &out){
    if(unit_assoca::Ref("A1") == inA){
        Provider::ListWriter::Item item(out,Ref(inA,
            unit_assocb::Ref("B1")));
        item.writeCaption("Hello, World!");
        item.writeC(unit_a::Ref(1));
    } else if(unit_assoca::Ref("A2") == inA){
        Provider::ListWriter::Item item(out,Ref(inA,
            unit_assocb::Ref("B2")));
        item.writeCaption("Hello, World!");
        item.writeC(unit_b::Ref("b1"));
    }
}
```

Example 3/4

- Get One Override

```
void Provider::getOne(const Ref &ref,ListWriter &out) {  
    if(ref.getKeyId() < MAX_ID)  
        Provider::ListWriter::Item item(out,ref);  
    ...  
}
```



Example 4/4

■ Modify Instance

```
void Provider::modify(const Ref &cop,  
    const cpa::string_optional &inValueString) {  
    if(!inValueString.isNull())  
        valueString = inValueString.getValue();  
}
```