



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

# Profile Definition Documents (PDD) – Lessons Learned

Bob Blair





# Agenda

- What is PDD?
- Why was it developed and where is it used?
- What are the parts of a PDD?
- How is a PDD created?
- What did we learn in creating and using PDD?
- Call to action



# What is PDD?

- Profile Definition Document
- An XML encoding of DMTF profiles
- *NOT* a DMTF standard or standards development effort
- Includes normative and descriptive elements of profiles
  - Structural elements
    - Name, number, etc.
    - Central and scoping classes
  - CIM Model elements
    - Association graph
    - Requirement (optional, conditional, mandatory)
    - Constraints
    - Conditions



# What parts of a Profile does PDD cover?

- Foreword. **No**
- Introduction. **No**
- 1. Scope. **No**
- 2. Normative References. **No**
- 3. Terms and Definitions. **No**
- 4. Symbols and Abbreviated Terms. **No**
- 5. Synopsis. **Profile Name, Version, Schema Version, Central Class, Scoping Class**
- 6. Description. **Association graph implied by the class diagram**
- 7. Implementation Requirements. **Normative behavior**
- 8. Methods. **Normative behavior**
- 9. Use Cases. **No**
- 10. CIM Elements. **Normative statements**



# Normative Behavior encoded in PDD

- Conditional and Mandatory instance presence
- Cardinality
- Conditional and Mandatory property support
- Conditional and Mandatory property value constraints
- Conditional and mandatory method support
- Method return value constraints
- Method parameter constraints



# Why was PDD developed?

- To describe testable elements of DMTF profiles for OpenTestMan
- There was a short deadline
- NOT with the intent that it was a long term solution
  - An industry solution is required
  - There are other use cases to consider:
    - MAP code generation
    - Client code generation
    - Reference document creation



# Where is PDD used?

- The OpenTestMan open source project on source forge
  - Where you can find it:
    - <http://www.opentestman.org>
    - SVN directory: Opentestman/trunk/etc/PDD
    - Documentation: Creating\_a\_PDD.html, ConditionTests.html
    - PDD XML schema: ProfileDefinition.xsd
    - PDDs: \*.xml
  - OpenTestMan is an open source project consisting of
    - Tests of WS-CIM (CIM binding for WS-Management) protocol implementations
    - Tests of implementations of 15 DMTF profiles (with more profiles to come)
    - Tests for the implementation requirements of DASH 1.0 (and eventually SMASH 2.0)
  - PDD is used as input to drive the profile tests
    - Profile testing code is generic. The actual test steps are created from the data in the PDD.
- At least one company is using PDD to generate CMPI-based code for profile implementation on an embedded MAP.



# Parts of a PDD

```
<PDD>
  <DOC>DSPxxxx</DOC>
  <DATE>xxxx</DATE>
  <VERSION>1.0.0c</VERSION>
  <TITLE>xxxx Profile</TITLE>
  <CENTRAL_CLASS>yyyy</CENTRAL_CLASS>
  <SCOPING_CLASS>zxxx</SCOPING_CLASS>
  <PROFILE_REGISTRATION_STRING>xxxx</PROFILE_REGISTRATION_STRING>
    <REGISTERED_VERSION>1.0.0</REGISTERED_VERSION>
  <ELEMENTS>
    <CLASSES>
      ...
    </CLASSES>
    <METHODS>
      ...
    </METHODS>
    <INDICATIONS>
      ...
    </INDICATIONS>
  </ELEMENTS>
</PDD>
```



# PDD Contents – Classes and Properties

```
<CLASS name="CIM_Fan" req="Mandatory" type="data">
  <ASSOCIATED_CLASS name="CIM_EnabledLogicalElementCapabilities"
    association="CIM_ElementCapabilities"
    thisclassrole="ManagedElement" assocclassrole="Capabilities"
    min="0" max="1"/>
  <ASSOCIATED_CLASS name="CIM_RegisteredProfile"
    association="CIM_ElementConformsToProfile"
    thisclassrole="ManagedElement"
    assocclassrole="ConformantStandard"
    filter="@RegisteredName='Fan' and @RegisteredVersion='1.0.0'"
    min="1" max="1"/>
  ...
  <NOTES>See sections 7.1 and 10.4. </NOTES>
  <PROPERTY name="SystemCreationClassName" req="Mandatory"
    classOrigin="CIM_LogicalDevice" cimtype="string" isarray="no" key="yes"
    maxlen="256">
    <NOTES>Key </NOTES>
  </PROPERTY>
```



# PDD Contents – Extrinsic Methods

```
<CLASS name="CIM_Fan" . . .>
```

```
...
```

```
<METHOD name="SetSpeed"
```

```
  req="Optional"
```

```
  classOrigin="CIM_Fan"
```

```
  cimtype="uint32">
```

```
<NOTES>1.0.0a does not list in Section 10 </NOTES>
```

```
<PARAMETER name="DesiredSpeed"
```

```
  cimtype="uint64"
```

```
  type="IN"
```

```
  isarray="no"/>
```



# PDD Contents – Conditions and Constraints

```
<PROPERTY name="EnabledState" req="Mandatory"
  classOrigin="CIM_EnabledLogicalElement" cimtype="uint16"
  isarray="no" range="0:65535" >
<CONDITIONAL_PROPERTY_VALUE_CONSTRAINT>
  <CONDITION
    test="matchany(associatedInstance('CIM_RedundancySet',
    'CIM_MemberOfCollection')[0]/@TypeOfSet, '4') or match-
    any(associatedInstance('CIM_RedundancySet',
    'CIM_MemberOfCollection')[0]/@TypeOfSet, '5') or match-
    any(associatedInstance('CIM_RedundancySet',
    'CIM_IsSpare')[0]/@TypeOfSet, '4') or match-
    any(associatedInstance('CIM_RedundancySet',
    'CIM_IsSpare')[0]/@TypeOfSet, '5') and
    associationInstance('CIM_RedundancySet',
    'CIM_IsSpare')[0]/@SpareStatus=3" />
  <PROPERTY_VALUE_CONSTRAINT range="3"/>
</CONDITIONAL_PROPERTY_VALUE_CONSTRAINT>
```



# How do you create a PDD?

- Extract class information from the profile Section 10.
  - This is a manual operation but text editors and cut-and-paste can help.

## 10.7 CIM\_NumericSensor

The CIM\_NumericSensor class is defined by the Sensors Profile. The requirements denoted in Table 23 are in addition to those mandated by the Sensors Profile (see section 2).

**Table 23 – Class: CIM\_NumericSensor**

Properties	Requirement	Description
SensorType	Mandatory	SensorType shall be set to 5
BaseUnits	Mandatory	BaseUnits shall be set to 19 (RPM).
RateUnits	Mandatory	RateUnits shall be set to 0 (None).

- Add MOF information
  - There is an xsl transform, addMOFData.xslt, that pulls this out of
- Add Profile Constraints and Conditions from Section 7.
  - This is intensely manual and can take many hours
- Add association graph information (mainly from class diagram in Section 6).



## What did we learn?

- Making profiles machine-readable extends their value.
- PDD is useful and has legs, but it is not the final answer
- The key to machine-readable profiles is transformability.
- Machine-readable profiles are essential and the format and content must be a community decision.



# How machine-readability makes profiles more useful

- The use of PDD for conformance test generation is useful, but just the top of the iceberg:
  - Profiles describe rules and behavior that management developers want to capture for several purposes, all of which become much easier when you have machine-readable profile descriptions:
    - Generation of management server code
    - Generation of management console and other client code
    - Generation of unit test, functional test and system test inputs.
- Text-based profiles are often imprecise.
  - Creating, or starting with, machine readable forms helps drive out errors and ambiguity.
- PDD experience tells us that machine-readable profiles are practical, labor-saving, and conducive to high-quality profile implementation.



# PDD is a useful start, but not the final answer

- PDD versions of 15 DASH profiles and quite a few SMASH profiles are extant. We expect the format to be used and useful for several years.
- PDD has some technical issues that need to be addressed:
  - PDD uses xpath syntax to express conditions. xpath is good for describing the relationships between XML elements, but there is a superior language, OCL, for describing object behavior and relationships. OCL almost certainly should be used.
  - Constraints are imposed both by modelers and by profile writers. PDD imposes a distinction between these kinds of constraints that perhaps does not need to be made.
- PDD has some maturity issues that need to be addressed:
  - Its requirements-gathering and development history has emphasized DASH MAP validation testing. As discussed earlier, there are wider applications, and they probably have their own requirements.
  - Profiles are a moving target. A language that describes profiles today, probably won't fully describe profiles next year. Machine-readable profile extensibility will be an important consideration.
  - Many think that machine-readable profiles should be done at the same time, or even before, text profile specifications documents. PDD development is clearly a separate and a later step.



# CALL TO ACTION

- Go get the PDD documentation and the profiles already encoded:
  - <http://www.opentestman.org>
- Discuss how your organization could use machine-readable profiles.
- Join the discussion on machine-readable profiles in your favorite industry standards organizations.