White Paper

DSP2011 ***Status: Informational***

Standard Messages Whitepaper 1.0 2/1/2006

Abstract

The DMTF Common Information Model (CIM) is a conceptual information model that describes computing and business entities in Internet, enterprise, and service provider environments. It provides a consistent definition and structure of data, using object-oriented techniques. The CIM Schema establishes a common conceptual framework that describes the managed environment.

The design of event reporting in CIM and WBEM must be more interoperable. A management client must determine how to respond, programmatically, to an Event reported to it by the managed element in order to automate management even in the face of adverse conditions. This whitepaper introduces the concept of Standard Messages as a framework for a solution to this problem.

Table of Contents

1. Introduction

The Standard Message Whitepaper describes the standardization of messages that can be used to report errors and other events of significance. It is necessary to standardize the messages so that consumers of the DMTF and DMTF inspired management models can take meaningful action on the message reported.

1.1 Overview

This white paper provides background information about why the general CIM/WBEM design and implementation has been amended with Standard Messages. Standard Message is a design that uses existing DMTF protocol and model technologies to express registered messages in the event of an error and other significant Events.

The objective of this paper is to:

- Explain why an additional approach to reporting Events was produced
- Provide a use case for when this new mechanism should be employed
- Introduce the concept of a Standard Message registry
- Provide some implementation considerations when using this design to report Events
- Distinguish between the use of Errors and Alerts to convey Standard Messages

1.2 Background Reference Material

There are no known overlapping standards or specifications.

1.3 Terminology

This section defines the terminology that is used within the white paper and indication model. Readers of this document should be familiar with CIM and the existing models, and have a general familiarity with what is required to implement a CIM client.

Term	Definition
Event	A change in the system
Error condition	An Event that results from the invocation of a CIM operation (method), adverse prevailing conditions of a managed element, or adverse prevailing conditions of a CIM Server or related infrastructural components
Standard Message	The expression of particular Event that includes information about how this message conveys the Event context
Error	The CIM_Error instance that conveys a Standard Message
Alert	The CIM_AlertIndication instance that conveys a Standard Message

2. The WIP Standard Message Model

This section describes the issues with current reporting mechanisms and the rationale for a Standard Message model.

2.1 Background and Assumptions

Until the publication of CIM 2.7.[1] and *CIM Operations of HTTP* DSP 200 version 1.2.[2], there were only two mechanisms for reporting errors that result from the invocation of intrinsic and extrinsic methods.

The first pre-existing error reporting mechanism reports CIM Status Codes[3]. These status codes represent only general categories of the types of errors that a CIM Operation might produce. In addition, these status codes are difficult to extend without backward compatibility concerns and do not provide the additional information a CIM Client might need. For example, the CIM_ERR_INVALID_PARAMETER status code cannot tell a CIM Client which parameter is invalid. In addition, the CIM_ERR_NOT_SUPPORT cannot express whether the extrinsic method is not supported or the related CIM Server does not support the invocation of extrinsic methods.

The second pre-existing error reporting mechanism uses return codes from extrinsic methods. Many extrinsic methods were designed to return the same codes, whether they were valid for that method. Moreover, as in the aforementioned status codes, return codes cannot express context sensitivity. For example, '5' "Invalid Parameter" of the CIM_EnabledLogicalElement.RequestedStateChange() method cannot tell the CIM Client which parameter is invalid or how the unspecified parameter is invalid. For example, the following information cannot be determined:

- Is the parameter the wrong type?
- Does the parameter contain a value out of range?
- Does the parameter contain a value that is too long?
- Is the parameter invalid because it specifies or implies an element that does not exist?

Some types of Events prevent the invocation of control and configuration operations, and even read operations. These conditions are typical for the CIM Object Manager or the device represented within it. The parameter that is passed to an intrinsic or extrinsic method may be correct. For example, a CIM Server might allow only a limited number of concurrent requests. In this case, no method call from a CIM Client will succeed. Because the resulting error is the same whether an intrinsic or extrinsic method is called, the message returned should be the same.

[1] See http://www.dmtf.org/standards/cim/cim_schema_v273
[2] See http://www.dmtf.org/standards/wbem/DSP0200.html
[3] See http://www.dmtf.org/standards/wbem/DSP0200.html#2.3.1.3

As a result of these shortcomings, the existing CIM/WBEM error reporting mechanisms do not provide for automated error and recovery. The management client implementer must retry various combinations until it achieves success or pass an expression of the situation back to the end-user so that he or she can figure it out. The ability of the IT staff to employ more computing resources, devices, and software, and to reach a higher level of productivity is limited by what they can accomplish in a reasonable amount of time. Certainly automation is a solution. However, the ability of the computing resources to be managed by other computing resources is blocked by the inability of a management application (that is, automatons), to parse the error condition, determine the context of the error, and handle the situation.

The solution is not to redesign Event delivery for CIM and WBEM, because existing methods are sufficient for reporting errors. The problem is that a CIM Client cannot always derive enough information from the context in the error expression to take meaningful action, other than write a log record. The CIM Client could investigate the error, but, in many situations, the context in which the error occurs is transient or difficult to determine from the model and can only be determined by the Event expression. The solution is to provide more meaningful Event expressions. In addition, the ability of a management application to respond to Events requires that the design for the model define what Events can occur, how those Events are conveyed to the CIM Client, and what those Events mean in relation to the rest of the model and behavior.

2.2 Conceptual Areas Addressed by the Model

This document will answer the following questions:

1. What are Standard Messages?
2. How do Standard Messages relate to current error reporting techniques?
3. How should Standard Messages be documented?

2.2.1 Understanding the Standard Message Model

Standard Messages are the expression of the Event and its context. Profiles define the conditions that produce a given message and the states that should be expressed in the message. The use of a Standard Message within the profile allows a CIM Client to determine the meaning of the message not only in the context of the profile, but also in the context of the prevailing conditions of the CIM Server and device or application instrumentation.

Events are reported for many reasons. Not all the reasons are directly related to the operation being imposed on the implementation by the client. The client must be able to distinguish between Events that are associated to problems in the formation and invocation of a method, extrinsic or intrinsic, or are related to other conditions. The client application might need to reform the method call, by fixing parameters, or the CIM Client might need to stop the operation it is attempting. At a basic level, the client needs to know whether this operation will succeed, given the prevailing conditions on the managed element. A CIM Client may also need to notify the end-user about the situation that is preventing the client from fulfilling its function.

There are three types of Events:

- Events that communicate problems using the method invocation

  For example, the method called might not exist, the method name might have a spelling error, or one or more of the parameters may be incorrectly formed, expressed, or otherwise invalid. This type of error informs the CIM Client that the attempted operation is still valid, but the request was faulty. This error informs the CIM Client what is wrong with the method call and allows the method to be invoked again.

- Events that communicate the adverse prevailing conditions in the managed element

  For example, the device or application may be in some type of failure condition that prevents it from honoring this particular method call or the next several calls. This type of error informs the CIM Client that the method being attempted will not be performed. Specifically, the method execution is blocked by the prevailing condition described in the message.

- Events that communicate the adverse prevailing conditions in the CIM Server or related, infrastructural components

  For example, the WBEM Service is a separate architectural element from the managed element. This Service can fail, even though the methods and the managed element are without error. For example, the CIM Server may allow only a limited number of concurrent connections or requests and reject all others. The server may be shutting down or starting up and, as a result, is unable to process any requests. Unlike the previous error type, this type of error is usually transient in nature.

2.2.2. Mapping of Standard Messages to Error Reporting Techniques

Before Standard Messages existed, problems with intrinsic method invocations were reported through CIM Status Codes and problems with extrinsic method invocations were reported through return codes. As previously explained, neither method communicates the context of the problem.

CIM 2.10.1, *CIM Operations of HTTP, DSP 200, version 1.2*, allows the WBEM error response to not only carry a status code and description, but also an Error instance. The Error contains properties very similar to the Alert properties. The Error instance was added to the error response to allow the communication of additional details about the error.

Alert instances report significant conditions about a managed element that is not necessarily related to any changes that a CIM Client may impose. In addition, the problem being reported does not need to manifest the characteristics of the problem within the CIM model being produced by the managed element. Generally, this design is like the use of ‘traps’ in report errors in SNMP.

Both CIM_Error and CIM_AlertIndication were modified in CIM 2.9 to add the following new properties: OwningEntity, MessageID, Message, and MessageArguments. These properties define the scope of the message ID, the contents of a message, and the

dynamic elements of the message, respectively. For more information, see CIM_Error.mof and CIM_AlertIndication.mof.

A CIM Client uses the value of OwningEntity to determine if it has knowledge of the Standard Messages being reported. If the CIM Client has been encoded with knowledge of the registry, then further processing may proceed.

The Standard Message does not contain a reference to a profile, if one is being implemented by the producer of the message, because any given Standard Message may be reused by many profiles. Because the semantics of the message is self-contained, the message does not need additional support from a profile. However, the profile can inform the CIM Client what messages are required to be compliant with the profile and how to address the adverse condition being communicated.

The CIM Client uses the MessageID to determine which Standard Message it received. The MessageID is unique for all Standard Messages for an OwningEntity value (for example, "DMTF"). The MessageID is defined as the combination of a working group or defining group and a sequence ID. For DMTF, each working group manages their own sequence number space.

The Message property contains a string that conveys both the message and the context. The Message contains text that is the same for each instance of the same Standard Message. These elements are known as static elements. The Message may also contain text that is different for each instance of the same Standard Message. These elements are known as dynamic elements. The static elements are declared as part of the message. The context of a Standard Message instance is conveyed by the dynamic elements. Each dynamic element in the array may have its own meaning. The Message property value can be localized because it is for the end-user.

The MessageArguments property defines the context of the message. The value of this property is the dynamic elements of the Message. The MessageArguments property is not localized because the CIM client application is looking the same values regardless of the locale.

The Standard Messages, whether from DMTF or some other organization, define what Events may be reported regardless of the method invoked.

The CIM-XML protocol, as defined in DSP 201[4], can return an error response or success response because the Standard Message is conveyed within a CIM_Error instance included in the error response rather than the success response[5]. Extrinsic methods generally return codes. These return codes determine the success or failure of the method. However, return codes are returned with the success response and not the error response. With the production of Errors rather than return codes, extrinsic methods may defined to return nothing (void) and still report errors.

The CIM_Error that is returned when a method fails is similar to an Exception in other object-oriented languages. In fact, the CIM_Error class contains the Exception qualifier.

[4] See http://www.dmtf.org/standards/documents/WBEM/DSP201.html
[5] See http://www.dmtf.org/standards/documents/WBEM/DSP201.html#SecERROR

2.2.3 Documenting Standard Messages

Errors and Alerts should be normatively documented as the model is documented. Many cases occur where the same error message can be used across many classes. These messages should be documented separately from CIM so that they are documented only once and can be reused where appropriate. A message registry is the published collection of the definitions of Standards Messages. Appropriate management of these registries ensures that the same Standard Message is not designed more than once.

Each message in the message registry describes the possible content for an instance of CIM_Error or CIM_AlertIndication. A message does not define new classes or new class properties and does not change the semantics of existing classes or properties. The registry defines the format and semantics of the contents of these four properties, OwningEntity, MessageID, Message, and MessageArguments. The Common Information Model defines Standard Message properties and their meaning. The registry extends this definition by narrowly defining specific content of properties for each Standard Message. For example, the MessageArguments property is defined as "An array containing the dynamic content of the message." As a result, the registry defines what each element of that array contains for a particular message.

For a CIM Client, the message registry defines how to interpret the contents of the aforementioned properties contained in a CIM_Error or CIM_AlertIndication instance. The registry also informs the implementer of the CIM Providers[6] how to convey the Standard Messages. Standard Messages allow automatons to be created that can respond to messages in ways that are different from merely displaying the Message to the end-user.

[6] See CIM_Provider introduced in CIM 2.8.1. for definition.

3. WIP Standard Message Model Use Case

This section presents a use case for the CIM Standard Message model. A use case provides an example that illustrates how the model is used for management purposes.

The MessageFormatString row in Table 1 defines how the Message is constructed from the message arguments, the dynamic components, and the static text components. The value of the MessageFormatString property is defined by appending the static and dynamic elements that are defined for that message, as they are listed from top to bottom in the registry.

Table 1 Example Standard Message Declaration

Message Property	**Value**
OwningEntity	DMTF
MessageID	WIP14
Name	Parameter Error
Description	A parameter error message is produced when the parameters for an intrinsic or extrinsic method are incorrect. This message informs the client about which parameters are problematic and why.
MessageFormatString	Parameter <*Position*> of the <*Method Type*> method, <*Method Name*>, is invalid, producing <*Status Code*> . <*Additional Status*>. Please fix the parameter at position <*Position*> and retry.
MessageArguments	*Position*: The position in which the errant argument appears in the declaration of the method, from left to right. *Method Type*: The intrinsic or extrinsic Method Name *Status Code*: CIM Status Code <status code> *Additional Status*: Additional circumstances describing the error (for example, Parameter out of range).

Using the following method declaration:

```
uint32 RequestStateChange(
     [IN, Description (“...”),
      ValueMap { "2", "3", "4", "5", "6",
"7..32767",
      "32768..65535" },
      Values { "Start", "Suspend", "Terminate",
"Kill", "Service",
      "DMTF Reserved", "Vendor Reserved" }]
    uint16 RequestedState,
      [IN, Description ("...")]
    datetime TimeoutPeriod);
```

A client makes the following call:

```
RequestedStateChange("1", null);
```

"1" is an invalid RequestedState. As a result, the target of the CIM Operation will produce an error. Table 2 shows the message properties for this example.

Table 2 Example CIM_Error Instance, Including Only the Standard Message Properties

Message Property	Value
Owning Entity	DMTF
MessageID	WIP14
Message	Parameter 0 of the extrinsic method, RequestStateChange, is invalid, producing CIM_ERR_INVALID_PARAMETER CIM Error. Parameter out of range.
MessageArguments	“0” "Extrinsic" "RequestedStateChange" “CIM_ERR_INVALID_PARAMETER” “Parameter out of range”

4. Future Work

The Standard Message must not only express the problem, but must also suggest what course of action to attempt. The drawback of the CIM_AlertIndication.RecommendedAction[] property is that it contains free-form strings. To provide programmatic processing for the Standard Message, a mechanism must be added to this class and the CIM_Error class to communicate the recommended action so that it may be processed. The design should not require the expression of every possible recommended action. It is likely that, like the message itself, there will be more static elements and more dynamic elements. A better design would express well-known verbs and object types, so that some simple language processing could be encoded into a CIM Client. As a result, semantics recommended actions could be read.

Appendix A – Change History

Version .01	October 2005	Initial Draft
.02	October 2005	1st review
.03	November 2005	2nd review
.04	December 2005	Revised 1st Technical Committee ballot
1.0	January 2006	Revised 2nd Technical Committee ballot
1.0	January 2006	cPubs revisions and corrections

Appendix B – References

[1] Common Information Model (CIM) Specification, V2.10.1, October 3rd, 2005, at http://www.dmtf.org/standards/cim/cim_schema_v2101 /

[2] CIM Operations over HTTP, DSP 200, V1.2 December 9th, 2004, at http://www.dmtf.org/standards/wbem/

[3] Specification for the Representation of CIM in XML, V2.1, May 02, 2002, at http://www.dmtf.org/standards/wbem/

Appendix C – Extending the Model

Vendors or organizations other than DMTF may create their own registries. These organizations must ensure that the MessageID is unique across all Standard Messages they develop. However, it is in the interests of the entire community that these organizations create additional Standard Messages only when the DMTF charter does not extend into particular areas of the computer system design domain, or if there are messages that would not match those of other vendors of the same types of products with similar functionality. In other words, we are all interested in having a single message defined by one organization, preferably DMTF, to define commonly applicable messages.