Specification

Using CIM for Support Activities:

Solution Exchange and Service Incidents

(Draft)

February 23, 2000

Abstract

This white paper describes the Common Instrumentation Model (CIM) object and transaction models for the exchange of knowledge related to support activities (Solutions) and the processing of Service Incidents. This document, along with the related UML and MOF, are intended to provide enough information for developers to create interoperable implementations. These models are referred to as the Problem Resolution Standard or PRS. All classes defined by this specification begin with the letters “PRS_” as opposed to the use of “CIM_” for other CIM-related object models. This prefix is due to historical reasons and all classes within PRS are to be treated as standard CIM extensions.

Change History

Version	Date	Description
0.91	February 23, 2000	Incorporated final comment from members and additional technical editing
0.9	February 17, 2000	Incorporated further comment from members and additional technical editing
0.8	February 15, 2000	Incorporated comment from members and performed technical editing
0.7	February 9, 2000	Initial Draft

Editors

Casey Bahr, Intel Corporation
John Chmaj, Microsoft Corporation
David Lawrence, Smart Technology Enablers, Inc.
For the DMTF Support Working Group

Table of Contents

1. Introduction

1.1. Purpose

This document presents an overview of the CIM object and transaction models for the exchange of knowledge related to support activities (Solutions) and the processing of Service Incidents. This document, in conjunction with the Support MOF and UML published by the DMTF, is intended to provide enough information to allow Producers and Consumers from different organizations to exchange Solutions using the DMTF Common Information Model ("CIM"). This document is also intended to provide enough information for Requesters and Providers developed by unrelated organizations to process Service Incidents.

1.2. Related Documents

Examples of Solution and Service Incident instantiations are available on the Exchange Standards pages of the Customer Support Consortium web site at www.customersupport.org.

The CIM schema and associated XML meta-schema adopted by the DMTF are available from the DMTF web site at www.dmtf.org.

1.3. Terms and Abbreviations

Term	Description
CIM	Acronym for the DMTF Common Information Model. CIM defines object models to promote interoperable management and configuration of computer systems and their hardware and software components.
Class	The definition of a related set of properties.
Consumer	The entity that consumes knowledge created by a Producer.
DMTF	Acronym for Distributed Management Task Force, formerly known as the Desktop Management Task Force.
Expression	Multiple statements used with Boolean logic to describe a Problem, Reference and/or Resolution.
Instance	An instantiation of a class with values for each property in the class definition.
Object	A convenient means of referring to both classes and instances with a single word.
Problem	A set of objects describing a problem that is part of a Solution.
Producer	The entity that creates knowledge.
Provider	The entity providing support to a Requester.
PRS	Acronym for Problem Resolution Standard.
Reference	A set of objects providing reference information that is part of a Solution.
Requester	The entity requested support from a Provider.

Term	Description
Resolution	A set of objects describing the resolution of a problem.
Service Incident	A single request for support by a Requester to a Provider.
SES	Acronym for Solution Exchange Standard, an earlier standard defining solution exchange, now incorporated in the Problem Resolution Standard.
SIS	Acronym for Service Incident Exchange Standard, an earlier standard defining Service Incident exchange, now incorporated in the Problem Resolution Standard.
Solution	A set of objects describing a single Solution. This may include one or more References, Problems and/or Resolutions.
Statement	Information used to describe Problems, References and/or Resolutions.

2. Background

In the past decade, there has been exponential growth in both the complexity and interdependence of products in the computing industry. This is due to rapid advances and growth of technology as well as the increased openness between products. A fundamental principle of system design in the development of modular, “plug-and-play” approaches, such as client/server in the computing industry, is to allow diverse products to work together. Customers increasingly require and expect products and the companies supporting them to work together to provide a total solution to their needs. These trends have created a demand in many industries for support providers to access support information about related products.

In response to this demand, many support providers have attempted to publish their information to each other on a more intensive basis, and to engage in partnerships that allow support analysts to collaborate on multi-vendor issues. However, without any standardized way to represent and communicate information, the process of gathering, publishing and interpreting the immense variety of support information remains costly, inconsistent—and largely ineffective. A solution exchange standard has broad applicability in the customer support domain, and has the potential to promote richer communication and collaboration between two or more support partners, both in solving specific problems and in evolving a more effective overall relationship.

Several factors are driving the support industry to adopt a standard method for exchanging Service Incidents. These factors stem from the multi-vendor nature of the computer and software industry. Many different companies create the computer components, peripherals, operating systems, and application software that comprise the personal computer market. Successfully providing customer support in today’s multi-vendor environment requires high levels of cooperation between support organizations. Providing such support has created efficiency and cost challenges to support providers and product vendors. Standardized incident exchange provides a mechanism for support organizations to share incident information effectively and ultimately reduce the cost of supporting computers, thus reducing the total cost of ownership to customers.

The Problem Resolution Standard (PRS) as described in this document and associated MOF and UML is the merger of two prior standards from the Distributed Management Task Force (DMTF) and the Customer Support Consortium (CSC) Support Working Group known as the Solution Exchange Standard (SES) and Service Incident Exchange Standard (SIS). The primary purpose of PRS is to define an open exchange standard that facilitates Solution exchange and Service Incident processing between cooperating parties, both within an organization and across organizational boundaries.

All classes defined by this specification begin with the letters “PRS_” as opposed to the use of “CIM_” for other CIM-related object models. This prefix is due to historical reasons and all classes within PRS are to be treated as standard CIM extensions.

3. Solution Exchange

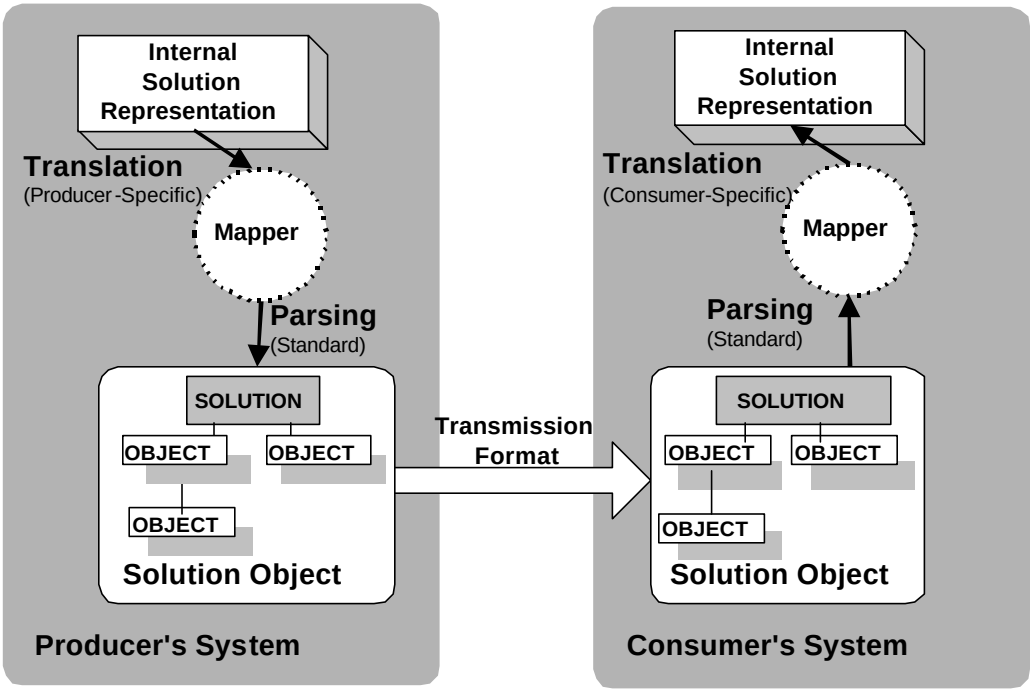
Before Solutions can be exchanged, they must be encoded so that they are consistently created by any compatible Producer and understood by any compatible Consumer. The exchange of Solutions is more than simply exchanging data (bits and bytes) and it is more than the exchange of random information (properties and classes). The exchange of Solutions requires an understanding between the parties of the exchange on the type of information being exchanged and the relationships (associations) between exchanged information to convey complete understanding or knowledge.

Consumers and Producers vary widely on the complexity and detail of the knowledge they process, so exchange must support this variety. Exchange participants “mine” the knowledge to the depth of their ability. Any exchange standard must also address the need to support extensibility so Consumers and Producers may extend the object model for their own unique needs.

It is important to note that this standard is focused on Solution exchange and not Solution storage. It is intended to facilitate the exchange of Solution knowledge without favoring any method of Solution storage.

3.1. Solution Object Model

The basic unit of exchange for PRS knowledge is the Solution. A Solution is a complete representation of a single issue. All of the information within a Solution is directly related to this one central issue. **PRS_SOLUTION** aggregates all of the elements of a Solution.



All non-association PRS classes inherit from **PRS_EXCHANGEELEMENT**. This class provides a common ancestry for the elements of a Solution. **PRS_EXCHANGEELEMENT** contains the key property PRSUniqueName. This property is the unique identifier for each instance of a PRS class making each instance unique within the world. Due to this uniqueness and the associative

nature of the model, any instance within a Solution has the potential to be reused in other Solutions.

3.2. Problem, Reference and Resolution

A Solution is subdivided into Problems, References and/or Resolutions. A Solution always has at least one of these elements and may have more than one. However, in the case of a Solution with multiple Problems, References and/or Resolutions, all of the elements always relate to a single issue.

A Problem is a description of a single issue. It may include symptoms, causes and other descriptive information including product identification. **PRS_PROBLEM** is the aggregating object for a Problem.

The object model allows more than one problem to exist within a single Solution. As an example, multiple 'print quality' Problems can be described by a single 'print quality' Solution. Because there can be multiple Resolutions in a single Solution, there are restrictions regarding the description of multiple Problems and Resolutions in the same Solution. In order to avoid ambiguity regarding which Problems go with which Resolutions, use the following rules for building Solutions:

- A single Problem may have many Resolutions (e.g., if Problem A happens, try Resolution X or Y or Z to fix the Problem).
- Multiple problems may have a single Resolution (e.g., if Problems A, B, or C happen, Resolution X fixes the Problem).
- Multiple Problems may have multiple Resolutions, if the Resolutions are essentially interchangeable with any of the Problems (e.g., if Problems A, B, or C happen, try Resolution X or Y or Z to fix the Problem).
- If particular Problems are tied to particular Resolutions they must be broken into separate Solutions.

A Resolution is used to describe how a Problem may be corrected. When a Solution contains a Problem, it typically contains one or more Resolutions. **PRS_RESOLUTION** is the aggregating object for a Resolution.

A Reference is used for general or non-diagnostic content that does not follow the Problem/Resolution paradigm. Solutions may be composed entirely of one or more References. This structure allows for non-diagnostic information to be exchanged. Examples of such information are: specifications, product documentation, FAQ's, and similar reference material. References may also be used in combination with diagnostic content (Problems and Resolutions) when supplemental material needs to be provided with a Solution. **PRS_REFERENCE** serves as the aggregating object for a Reference.

Problem(s), Reference(s) and/or Resolution(s) may also be categorized as described in the following section.

3.3. Categorization

Solutions, Problems, References or Resolutions may all benefit from classification. For example, a Solution may be related to installation issues, printing or specific functionality of a program. By classifying a Solution, a Producer imparts potentially significant information about

how the Solution was intended to be used and allows searches to be quickly focused on specific areas.

PRS_CATEGORY is used for classification. Solutions are associated to **PRS_CATEGORY** through the **PRS_SOLUTIONCATEGORY** association. The elements of a Solution (Problem, Reference and Resolution) are associated to a **PRS_CATEGORY** through a **PRS_CATEGORIZATION** association.

Hierarchical classification is supported through the use of a **PRS_CATEGORYPARENTCHILD** association between two **PRS_CATEGORY** instances.

3.4. Products and Statements

Problems, References and/or Resolutions are described as a set of related Products (**PRS_PRODUCT**) and/or Statements (**PRS_STATEMENT**).

Specific products are identified by *Vendor*, *Name* and *Version*. To completely describe a product (or a range of products) and their relationship to other products and its environment, **PRS_PRODUCT** may be used with expressions and other associations.

For example, a Product may exist in a hierarchy of relationships with other Products. If the implementer deems these hierarchy relationships are relevant to the current Problem or Resolution, then the position of the Product in a hierarchy is defined using one of two types of relationships: “is a” or “has a”.

An “is a” relationship implies a classification, such as versioning or typing. The **PRS_PRODUCTPARENTCHILD** association is used to define a classification hierarchy.

A “has a” relationship implies a component relationship: one product contains another. The **PRS_PRODUCTCOMPONENT** association is used to describe containment relationships.

The **PRS_PRODUCTASSET** association is be used to define products as assets. An expression and statement(s) are used to state more complex identification than is allowed with the simple *Version* data. For instance, a range of lot numbers or manufacturing dates may be expressed as groups of products associated through **PRS_PRODUCTASSET**.

Products also may be parts of expressions in a Problem or Resolution to describe the exact product context of the situation. For instance, the Problem only occurs when (Product X AND (Product Y OR Product Z)) are active. Since both **PRS_PRODUCT** and **PRS_STATEMENT** derive from **PRS_EXPRESSIONELEMENT**, the **PRS_EXPRESSIONLINK** is used to define expressions between Products and Statements.

Statements may be units of text ranging from brief sentences or phrases to paragraphs or pages. There is no limit to the number of statements that may be provided. More complex relationships are defined through the use of Boolean expressions. For example,

StatementA = “Printer output is blotchy.”

StatementB = “Print jobs per day are less than 50.”

ExpressionA = (StatementA) AND (StatementB)

PRS_EXPRESSION is used when multiple statements define a Problem or Resolution. When using multiple statements, expressions enable you to build a Boolean expression defining the relationships between the statements using the relations: AND, OR, and NOT. The expression acts like a wrapper for a collection of statements and other nested expressions. The relation governs how the statements and nested expressions are to be treated. However, many Problem and Resolution descriptions do not require complex expressions, instead they are a simple list

of Statements grouped together with an implied AND relation, the default relation type for an expression.

In some diagnostic systems specific conditions may be stated more formally, as actual values rather than free-form text. **PRS_FEATURE** is used when specific features and their values may be defined. When used in conjunction with **PRS_STATEMENT**, **PRS_FEATURE** defines the name, type and legal values that may be used in creating a *Feature*/*Feature Value* pair.

Features contain specific values that conform to some index or array. The examples above could be expressed as “Output (feature) is (operator) blotchy (Feature Value)”, or “Print jobs per day (feature) are less than (operator) 50 (Feature Value). While the *Operator* and *Feature Value* (e.g., < 50) used in a particular statement may be specific to a single problem or resolution, the *Feature* (e.g., print jobs per day) is likely to be used across many statements in many problem and resolutions.

The degree of Statement detail used to describe Problems, References and Resolutions within a Solution is identified as part of the Administrative information associated with a Solution.

3.5. Administrative, Revision and Contact Information

Solution exchange requires more than a description of an issue. There is also a need to describe information related to the administration, revision and contacts related to a Solution.

PRS_ADMINISTRATIVE includes copyright, disclaimer and rights statements. There is at least one **PRS_ADMINISTRATIVE** instance associated with the **PRS_SOLUTION** instance of a Solution. This administrative information relates to the entire Solution. If there is administrative information for other instances within the Solution, separate **PRS_ADMINISTRATIVE** instances are associated with those instances. This approach allows for definition of administrative information at any desired level of detail within the model.

In addition to describing the ownership of a Solution, there is a need to associate revision information with a Solution (or portions of a Solution). Revision information includes information such as the current editorial status or when the related instances were last revised. **PRS_REVISION** instances are associated with **PRS_ADMINISTRATIVE** instances for this purpose.

It is also important to identify who is the designated contact for administrative or revision information. Contacts may be person and/or organizations. Every contact must be associated with at least one person or organization. Multiple persons and organizations may be designated as the administrative or revision contact. Each person may have multiple addresses and each address may have an associated location description.

3.6. Attachments

Data that is not a PRS object is associated with a Solution (or a component of a Solution) as an attachment. Examples are files representing video and audio clips or documents. Each attachment requires a **PRS_ATTACHMENT** instance to be associated with a non association PRS object. Attachments are by reference or by value. An attachment by reference is a link to the actual attachment through the use of a string identifier, typically a Universal Resource Locator or URL. An attachment by value means the actual attachment is included within the **PRS_ATTACHMENT** instance.

3.7. Resource Tracking

There may be a cost associated with a Resolution or when using detailed Statements that include Feature/Value pairs. This cost may have several components such as schedule, effort, and materials. **PRS_RESOURCE** instances are used to track these costs.

4. Service Incidents

Service Incident processing builds on the Solution exchange object model. All of the objects defined for Solution exchange may be used in Service Incident processing. Service Incident processing adds five new classes, a transaction model and some new associations to the Solution object model.

4.1. Requester and Provider

The parties in the processing of a Service Incident are known as the Requester and the Provider. Requesters initiate a request for service and Providers respond to those requests.

Service Incident processing is modeled as a series of point-to-point transactions. If a Service Incident needs to be escalated to a third party, the original Provider creates a new Service Incident and becomes a Requester for the new Service Incident. The third party becomes the Provider for the new Service Incident.

4.2. Service Incident

A Service Incident is a set of objects related to one request for service from a Requester to a Provider. A **PRS_SERVICEINCIDENT** instance contains information directly related to a specific Service Incident. A **PRS_SERVICEINCIDENT** instance is associated with a single **PRS_AGREEMENT** that identifies the Agreement between the Requester and Provider for the service requested.

Agreements define the relationship between the two parties including the level of service the Provider will perform, how quickly the Provider commits to responding, and which products are covered. **PRS_AGREEMENT** contains a reference or contract identifier and a brief description of the service contract. The contract identifier may be used to reference a more detailed specification of the contract between Requester and Provider.

A Service Incident is processed by exchanging objects between the parties and invoking methods on local or remote instances of **PRS_SISSERVICE**. Each object exchange between the parties is delimited by a Transaction represented by a **PRS_TRANSACTION**. The objects affected by a Transaction (created, modified or deleted) are noted within a **PRS_ACTIVITY** associated with the **PRS_TRANSACTION**.

Service Incident processing makes small modifications to the usage model for Solution objects by adding new associations. For example, Solution Exchange defines which objects may be associated with other objects and always requires a **PRS_PROBLEM** to be associated with a **PRS_SOLUTION**. Service Incident processing allows **PRS_PROBLEM** to be associated with a **PRS_SERVICEINCIDENT** in order to describe the Problem encountered by the Requester.

Other association modifications to the Solution exchange object model include:

- **PRS_CONTACT** instances are associated with a **PRS_SERVICEINCIDENT** instance to identify the Requester and the Provider.
- **PRS_RESOURCE** instances may be associated to **PRS_ACTIVITY** instances to document the cost of a specific Transaction.

4.3. Transactions

Requesters and Providers step through defined Transactions to process Service Incidents. The following Transactions are defined:

Transaction	Source	Description
AcceptProblem	Provider	Acknowledge the Problem submission and set response guidelines.
ConfirmClose	Requester	Confirm resolution and close Service Incident.
Entitlement	Provider	Acknowledge or deny request for service.
ProblemResolution	Provider	Submit technical resolution to Problem (Solution).
ProblemSubmittal	Requester	Submit Problem and desired response.
ProvideAdminInformation	Requester or Provider	Respond to RequestAdminInformation with Administrative information.
ProvideProblemInformation	Requester or Provider	Respond to QueryIncident or submit unsolicited information about Problem.
QueryIncident	Requester or Provider	Request Administrative (non-Problem) information.
RejectResolution	Requester	Reject Resolution submitted by Provider.
RequestClosure	Requester or Provider	May be used at any time to request the Service Incident be closed.
RequestProblemInformation	Requester or Provider	Request all of the problem data elements associated with the Service Incident
ServiceRequest	Requester	Request service.

4.4. Legal Transactions

At any point in time, the set of Transactions that may be legally performed by the Requester or Provider are based on the shared state of the Service Incident. The table below identifies legal transactions for specific Service Incident states:

State	Legal Transactions
BEGINNING_STATE	***RequestService***
CLOSED	***QueryIncident***
CLOSED_PENDING_CONFIRMATION	***ConfirmClose*** ***RejectResolution*** ***QueryIncident*** ***ProvideAdminInformation*** ***RequestClosure***
ENTITLED	***ProblemSubmittal*** ***QueryIncident*** ***RequestClosure*** ***ProvideAdminInformation***
NOT_ENTITLED	***Entitlement*** ***RequestClosure*** ***ProvideAdminInformation***
OPEN_PROVIDER	***RequestProblemInformation*** ***ProvideProblemInformation*** ***ProvideAdminInformation*** ***ProblemResolution*** ***QueryIncident*** ***RequestClosure***
OPEN_REQUESTER	***RequestProblemInformation*** ***ProvideProblemInformation*** ***ProvideAdminInformation*** ***ProblemResolution*** ***QueryIncident*** ***RequestClosure***
PROBLEM_SUBMITTED	***AcceptProblem*** ***QueryIncident*** ***RequestClosure*** ***ProvideAdminInformation***

All Service Incidents are originated by a Requester. The initial state of a Service Incident is BEGINNING_STATE. The only legal transaction from this state is ***RequestService***. The

Provider responds granting or denying support with the ***Entitlement*** transaction. If the Provider grants support, the Service Incident state is moved to ENTITLED.

The Requester submits a Problem associated with the Service Incident using the ***Problem Submittal*** Transaction. Once the Provider accepts the problem (as indicated by an ***Accept Problem*** Transaction) the Service Incident moves to the PROBLEM__SUBMITTED state.

Subsequent transactions exchange problem or solution data and may be iterative, moving the Service Incident between states of OPEN_PROVIDER and OPEN_REQUESTER, depending on which side, Requester or Provider, is requesting information. At any time after BEGINNING_STATE, the Provider may Request Closure of the Service Incident for any reason, though only the Requester is able to actually close the Service Incident using the ***Confirm Close*** transaction. Both the Requester and Provider may also perform transactions that do not alter the state of the Service Incident at any time, for example, ***Query Incident***, ***Request Admin Information***, and ***Provide Admin Information***. These are used to obtain the current state of the Service Incident by either party or to provide or gather non-technical information, such as contact or resource information.

4.5. Required Objects and Properties

Each Transaction requires specific data elements be passed between the parties. The objects and properties required for each Transaction are listed below. Additional objects and properties may be passed with a Transaction.

Transaction	Required Data Elements
All Transactions	*Requester ID* **PRS_CONTACT** **PRS_ACTIVITY**
AcceptProblem	*Requester ID* *Service Provider ID* *Current State* *Priority*, *Response Time*, *Comment*, *Workflow Status* **PRS_ACTIVITY**
ConfirmClose	*Comment* **PRS_ACTIVITY**
Entitlement	*Entitled* *Comment* (if *Entitled* indicates rejection) **PRS_ACTIVITY**
ProblemResolution	**PRS_RESOLUTION** or **PRS_SOLUTION** *Comment*, *Workflow Status* **PRS_ACTIVITY**
ProblemSubmittal	**PRS_PROBLEM** *Severity*, *Response Time*, *Comment*, *Workflow Status* **PRS_ACTIVITY**
ProvideAdminInformation	*Priority*, *Response Time*, *Comment*, *Workflow Status* **PRS_CONTACT** **PRS_ACTIVITY**
ProvideProblemInformation	**PRS_PROBLEM**: **PRS_STATEMENT** **PRS_ACTIVITY**
QueryIncident	**PRS_ACTIVITY**
RejectResolution	*Comment* **PRS_PROBLEM**: **PRS_STATEMENT** *Workflow Status* **PRS_ACTIVITY**
RequestClosure	*Comment* *Workflow Status* **PRS_ACTIVITY**
RequestProblemInformation	**PRS_PROBLEM**: **PRS_STATEMENT** **PRS_ACTIVITY**
ServiceRequest	**SERVICEREQUESTER** **SERVICEPROVIDER** **PRS_AGREEMENT** *Comment*, *Workflow Status* **PRS_ACTIVITY**

4.6. Activity Tracking

PRS_ACTIVITY instances provide an audit trail over the life of a Service Incident. Each Transaction on a Service Incident creates a **PRS_ACTIVITY** instance.

Each **PRS_ACTIVITY** instance contains three basic properties:

- A list of the objects that were created, deleted, or modified by the associated Transaction.
- A timestamp for the Transaction.
- A string denoting what type of general action was performed with regard to the Service Incident.

Furthermore, additional information related to the associated Transaction may be found through associations to **PRS_RESOURCE** and **PRS_CONTACT** instances.