CIMPortal Plus

Developer's Guide

Version: 2.2.1.301
Date: 11/09/2014

Table of Contents

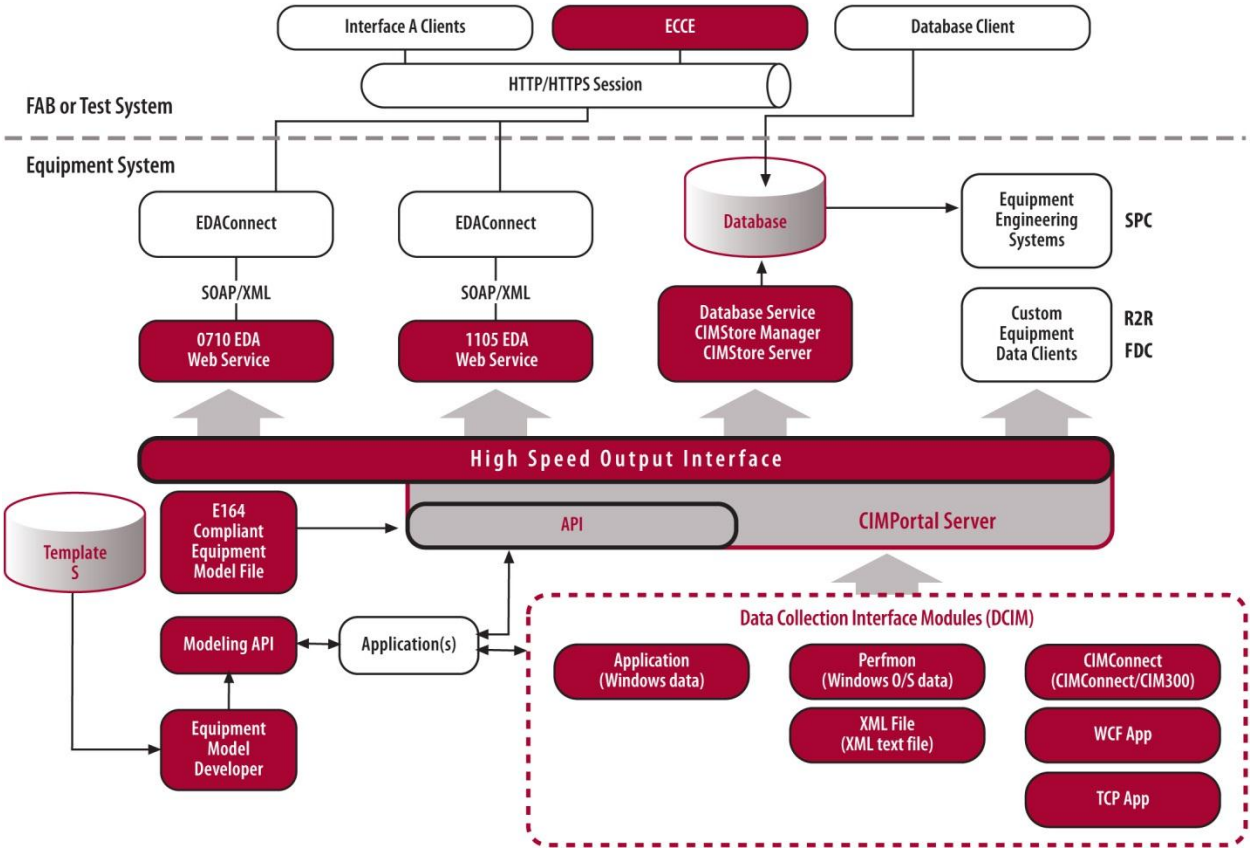
1. Welcome to CIMPortal Plus

CIMPortal Plus is a toolkit for implementing the SEMI Equipment Data Acquisition (EDA) standards on manufacturing equipment. The EDA standards include E120, E125, E128, E132, E134, E138 and E164 also known as Interface A. CIMPortal Plus is also a toolkit for storing data into a database on manufacturing equipment. MS SQL, MySQL and Oracle are supported.

1.1 Architecture

The CIMPortal Plus Architecture is built using Microsoft technology including a combination of C++, COM, C#, WCF and Windows Services. CIMPortal Plus and most of its modules must run on Windows computers. The modules can run all on the same computer or be distributed across multiple computers. Nevertheless, CIMPortal Plus can be used in combination with non-Windows computers. The TCP/IP Application DCIM can be compiled and run on most non-Windows computers.

The following figure illustrates the overall CIMPortal Plus Product architecture.



The figure shows the following modules:

Module	Description
CIMPortal Server	The central module of the CIMPortal Plus product. It exposes a number of data collection services to clients. It is responsible for using the contents of the Equipment Model Package(s) and communicating with DCIMs as data collection sources in order to build and collect Data Collection Reports for clients. Often this is also called the CIMPortal Plus Service since it typically runs as a Windows Service. CIMPortal Server has API functions for collection data (called the High Speed Output Interface), functions for access the DCIM interfaces, modifying the equipment model and for configuration CIMPortal Plus's behavior. The API are protected by Client Security where most API functions require a client name

	and password. Model Security allows private data in the equipment model that is restricted to particular clients.
XML Equipment Model File	In order to describe the data available for gathering, developers must create an equipment model file which includes a hierarchy of the relevant hardware components, all of the available parameters, events, exceptions and other metadata. The Cimetrix Equipment Model File is an XML file that complies with a Cimetrix Equipment Model schema. It also maps the available data to DCIM data sources.
Equipment Model Developer	The Equipment Model Developer GUI application is responsible for building and creating equipment models packages. The packages are then used by the CIMPortal Server runtime engine.
Modeling API	In some situations it is more convenient to create the Equipment Model File programmatically. This can be done by using the Modeling API functions.
1105/0710 EDA Webservice	This web service, called CIMWeb, implements the Interface A standards, including E132, E134, E120, and E125 to web clients. CIMWeb is a client of CIMPortal Server that requests data. CIMWeb1105 is the name of the CIMWeb web service that implements the November 2005 version of the SEMI standards.
Database Service	There are three major components unique to CIMPortal Plus DB, CIMStore, CIMStore Manager and the database. CIMStore is the process that collects data from CIMPortal Plus Server and feeds the formatted data into the database. CIMStore Manager is a GUI to monitor and configure CIMStore's data collection. CIMStore supports MS SQL, MySQL and Oracle.
Data Collection Interface Modules (DCIMs)	DCIMs are used by CIMPortal Plus at runtime to collect data include parameters, events and exceptions. Each DCIM collects data from a particular data source. This data is collated by CIMPortal Server and reported to the client who requested it such as CIMWeb or CIMStore. DCIM Packages include the Application DCIM, CIMConnect DCIM, CIMConnect SR13+ DCIM, TCP/IP App DCIM, WCF App DCIM, Perfmon DCIM and XML DCIM. Each DCIM Package can be used one or many times within the same equipment model to collect data from different sources. The configuration for each DCIM Package's usage is called a DCIM Instance.
DCP Manager	An Interface A client also called Equipment Client Connection Emulator (ECCE) developed by Cimetrix as a reference client to validate Interface A (EDA) implementations. It is the only reference client endorsed by ISMI and therefore it is used by companies around the world. Using ECCE it is very easy to create data collection plans, activate them and view the data. ECCE has a separate installation and should be installed and run on a separate computer and then used to check Interface A before shipping to any end users.
Templates	CIMPortal Plus includes a series of Equipment Modeling template files to accelerate and facilitate equipment model development. In particular, these templates make it simple to build EFEMs, process chambers, and to add all of the GEM 300 capabilities. Developers can use the templates and create custom templates.
Equipment Control Applications	Software applications can use the CIMPortal Plus API to control and monitor CIMPortal Plus. Software applications can also feed data into the Application DCIM, WCF App DCIM, XML DCIM or TCP/IP App DCIM. Software applications can run on the same Windows computer as CIMPortal Plus or on another networked Windows computer. Software applications using the TCP/IP App DCIM can run on non-Windows networked computers to feed data into CIMPortal Plus.
Database Clients	Database clients, such as SPC systems, can use ODBC to collect data from the database.
Data Collection Applications	Data collection applications, like R2R and FDC, can collect data from the CIMPortal Server. This data collection can occur even while CIMWeb and CIMStore are running and collection data.

Here are some modules that are not illustrated:

- Equipment Model Packages are used by CIMPortal Plus at runtime. Each Equipment Model packages provides an Equipment model, schema, ACLs, and DCIMs to be used to gather data at runtime for an Interface A Client or for database data collection.
- Cimetrix Configuration Service and Configuration Manager are used to manage client permissions and logging configuration for multiple Cimetrix packages.

1.2 Introduction to the SEMI EDA Standards

1.2.1 Overview

The Equipment Data Acquisition (EDA also known as Interface A) Standards are a collection of SEMI standards for the semiconductor industry to improve and facilitate communication between IC Makers' data gathering software applications and the factory Equipment. When implemented together, these standards provide a convenient interface for Equipment Data Acquisition using SOAP/XML messages over an HTTP or HTTPS connection. The main EDA SEMI standards include E120, E125, E132, E134, and E164. Solutions must comply with the specific SOAP/XML implementations of these standards; E120.1, E125.1, E132.1, and E134.1. The implementation of E164 is still pending.

EDA provides multiple client access to data gathering capabilities. It does not purport to replace the SEMI GEM/SECS standards (E4, E5, E30, and E37) or the SEMI 300mm standards (E39, E40, E87, E90, E94, and E116) since EDA does not provide any features for Equipment control or configuration. Instead, EDA must be supported in addition to other required interfaces. Over time, EDA could replace the need for any other data providing interfaces so that IC Makers only require the EDA and SECS/GEM/300mm connectivity standards. In practice, some initial EDA deployments may provide little more data than the SECS/GEM interface, but eventually it is expected to provide more data, particularly state information, sensor feedback, actuator states, and other raw data necessary for process, product and equipment analysis. All data must be supplied to EDA as directly from the source as possible with minimal software layers. In order to make all expected data available and achieve performance expectations, some equipment will require internal restructuring and architectural changes.

During 2005 and 2006 IC Makers started requiring integrated EDA solutions from the Equipment Suppliers. These ISMI specified a new Freeze Version of these Standards in 2010 indicating a renewed interest in EDA. The demand will continue to increase as IC Makers roll out plans to improve yield and equipment utilization.

1.2.2 Terminology and Acronyms

Term or Acronym	Description
3507	ID for E132 during the Task Force development stage.
3509	ID for E134 during the Task Force development stage.
Access Control List (ACL)	Part of E132, the Client Authorization details that grant or deny Client sessions and impose restrictions on Clients access to specific Interface A information and operations.
CEM	E120.1 XML Schema for the Common Equipment Model
Client Authentication	In order for an Interface A Client to gather data, E132 requires clients to first establish a session. The client must provide credentials to the Interface A Server. The Interface A Server must be preconfigured to grant the client permission (based on the credentials) to establish a session.
Client Authorization	Part of E132, before the Interface A Server accepts an operation request from the Client, the Server must verify that the Client has permission. These permissions are preconfigured in the Server using Access Control Lists.
Client Consumer	An Interface A Client that receives the Data Collection Reports and other E134 consumer operations.
Client Manager	An Interface A Client that establishes a session, identifies the Consumer to receive the data, sets up Data Collection Plans, and uses other E134, E132, and E125 manager operations.

Data Collection Plan (DCP)	Part of E134, a data gathering request that includes a set of Events (with a configurable set of Parameters), Exceptions (with a fixed set of Parameters), and Traces (with a configurable set of Parameters). After successfully creating a Data Collection Plan, it must be activated. Then the Client will receive the respective Data Collection Reports as configured in the plan.
Data Collection Report (DCR)	The Equipment sends the requested data in this standard format.
DCM	E134.1 Provisional Specification for SOAP Binding of Data Collection Management
ECA	E132.1 Provisional Specification for SOAP Binding for Equipment Client Authentication and Authorization
EDA	Equipment Data Acquisition. The combination of SEMI standards E120, E125, E128, E132, E134 and E138. This term is more common the term “Interface A”, but both refer to the same thing. Of course, the EDA acronym is also used in the semiconductor industry to stand for Electronic Design Automation, but this is unrelated to this document or the referenced SEMI standards.
Equipment	Equipment refers to the hardware and software received from Equipment Supplier to perform work for the IC Maker. In some context of Interface A, Equipment refers to the Interface A Server that represents the hardware and software to the Clients.
ESDS	E125.1 Provisional Specification for SOAP Binding for Equipment Self-Description
Exception	Part of E134, an Equipment alarm, error, or warning notification.
Event	Part of E134, a notification that something important occurred on the Equipment tied to a state machine. An Event request in a DCP can include any set of Parameters.
FF	Fire-and-Forget where a message sender does not expect a reply message.
Host	The software an IC Maker is running to communicate with the Equipment. A host typically refers to the SECS/GEM connection but could also refer to a client using Interface A.
HTTP	HyperText Transfer Protocol: the protocol for moving hypertext files across the Internet. Requires a HTTP client program on one end, and an HTTP server program on the other end. HTTP is the most important protocol used in the World Wide Web (WWW)
HTTPS	HyperText Transport Protocol (Secure): the standard encrypted communication mechanism on the World Wide Web. This is actually just the use of Netscape's Secure Socket Layer (SSL) as a layer under its regular HTTP application layering.
SSL	Secure Sockets Layer: cryptographic protocol that provides communication security.
CA	In cryptography, a certificate authority, or certification authority, (CA) is an entity that issues digital certificates.
Interface A	The resulting Equipment interface when SEMI standards E120, E125, E132, and E134 are implemented together. Interface A is also known as EDA.
Interface A Client	Software that attempts to use the Equipment’s Interface A by establishing a session. Typically, this is developed by the IC Maker or a third party hired by the IC Maker. Equipment suppliers can also develop Interface A clients, such as for capturing diagnostic information. This is also called an EDA Client.
Interface A Server	The Equipment software that implements the Interface A standards. This software should be installed on the Equipment’s internal computer and fully integrated into the Equipment’s system. However, the software can be run on an external computer to retrofit Equipment that does not have an integrated solution. This is also called an EDA Server or EDA Web Server.
Interface B	A collection of SEMI standards to implement data sharing between applications (primarily IC Maker applications), such as Statistical Process Control and Run-To-Run applications.

Interface C	A collection of SEMI standards to allow remote access to Equipment data. This is intended primarily for Equipment Suppliers to remote obtain the Equipment's diagnostic and maintenance information.
Metadata	Information that describes the data, such as when an Event occurs or the interpretation of a Parameter's value.
Operation	An Interface A transaction, method, or message initiated by an Interface A client or server. Each operation has a name, well-defined format and meaning. E125, E132, and E134 each define a set of operations for the client and server.
Parameters	The set of data available for gathering from the Equipment's Interface A connection.
PR8	Proposed Standard for the Equipment Data Acquisition: a previous name for the E134 standard.
RR	Request-Response where a message sender expects a reply message.
Security Admin	A part of E132, Security Admin is a utility provided with the Interface A Server to provide administrative configuration.
Session	Also called an Authenticated Session, a session is established between the Server and Client by following the E132 procedures. Once a session is established, the Client can send authorized operation messages.
SOAP	Simple Object Access Protocol (SOAP): In order to make the Interface A standards easier to implement, they use the SOAP protocol. It is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only bindings defined in this document describe how to use SOAP in combination with HTTP and HTTP Extension Framework. See www.w3.org/TR/2000/NOTE-SOAP-20000508 information.
Tool	A synonym for Equipment. The term Equipment is used more frequently in the standards.
Trace	Data polling performed by the Equipment as defined by the Client. The Client defines the polling frequency, the set of polled Parameters, and the conditions to start and stop polling.
UML	Unified Modeling Language (UML): All of the Interface A standards use UML notation for all class diagrams and for object oriented diagrams provided as examples. UML is a notation for representing object-oriented designs and views created by Booch, Rumbaugh, and Jacobson in order to merge their three popular notations plus aspects of other existing notations into a single object-oriented notation intended to be usable by all. UML is an open modeling language to specify, visualize, design, and document models of software systems. See www.uml.org or www.omg.org/technology/documents/modeling_spec_catalog.htm for more information.
XML	Extensible Markup Language: A markup language used for representing data rich with context and content in documents and in communications. XML is an extension of SGML, a document-oriented markup language. It was created by W3C for use on the Internet. See www.w3.org/TR/REC-xml for more information.
XML Schema	An XML Schema defines the structure, content and semantics of XML documents.

1.2.3 SEMI Standards

EDA comprises of multiple SEMI standards including the following. All of these are available for download from the SEMI website for a fee, http://www.semi.org.

1.2.3.1 E120 Specification for the Common Equipment Model (CEM)

E120 provides a general object model that represents an external view of Equipment. The model is composed of various classes organized in a logical hierarchy. A fully-implemented model summarizes all of the Equipment's major hardware and software components. SEMI standard **E120.1 XML Schema for the Common Equipment Model (CEM)** maps the E120 standard into a specific XML implementation.

E120 defines the following concrete classes.

Class	Description
Equipment	Models the equipment as a whole and contains MaterialLocations, Modules, Subsystems and IODevices.
Module	Models a major equipment subsystem that can process material like a process or inspection chamber. It may contain MaterialLocations, Modules, Subsystems and IODevices.
Subsystem	Models a major equipment subsystem that cannot process material like a load port or prealigner. It may contain MaterialLocations Subsystems, and IODevices.
IODevice	Models sensors, actuators or intelligent actuator or sensor devices.
MaterialLocation	Models the ability of an equipment component to hold material, specifying the material type as Carrier, Substrate, or ProcessDurable.

E120 also defines the following additional concrete and abstract classes.

Class	Description
SoftwareModule	Describes the existence and version of software that is in use on the equipment and on equipment components. The software might be supplied by the equipment manufacturer or a third party. Attributes identify the supplier, version and a description.
EquipmentElement	The basic information required for each hardware component, such as an AbstractModule, Subsystem or IODevice. Attributes identify the supplier, make, model, role, revision, and (if applicable) serial number.
AbstractModule	A subclass of EquipmentElement that models the parts of the equipment structure capable of processing material; namely Modules and Equipment. The attributes specify the process type as Measurement, Process, Storage or Transport and specifies the recipe type.
Nameable	The root of every class to provide unique identification and a description for each component.
Extension	Provides the ability for other standards and specific implementations to extend the CEM.

1.2.3.2 E125 Specification for Equipment Self Description (EqSD)

The E125 standard allows clients to request helpful descriptions about all information available for data gathering include the parameters (specific data, units, and types), events, exceptions, state machines, SEMI E39 data, and physical configuration. All of the available information is mapped into the E120 Common Equipment Model object hierarchy. SEMI standard **E125.1 Provisional Specification for SOAP Binding for Equipment Self Description (ESDS)** maps the E125 standard into a specific SOAP/XML implementation.

This standard allows the end-user to know information about what data can be monitored on the Equipment and the context (i.e. process chamber #1 or #2) without having to entirely rely on the Equipment's documentation. Interface A clients can implement plug-and-play methodology, automatically utilize new information immediately after an Equipment's interface is revised, and implement graceful error recovery if data is unavailable unexpectedly.

E125 defines an interface with client-initiated operations to query the Equipment's available metadata information.

Operation	Description
GetUnits	Retrieve all available unit definition information.
GetTypeDefinitions	Retrieve all available type definition information.
GetStateMachines	Retrieve all available state machine information and the associated events.
GetSEMIObjTypes	Retrieve all available SEMI E39 information.
GetExceptions	Retrieve all available exception information.
GetEquipmentStructure	Retrieve all SEMI E120 Common Equipment Model information. Each component in the Equipment structure is a node.
GetEquipmentNodeDescriptions	Retrieve one or more Equipment node descriptions including the associated state machines, SEMI E39 information, exceptions, and parameters. A parameter represents an element of data that can be gathered. Its attributes describe it meaning, how the value is changed, unit, and the type (integer, real, string, array, enumeration, or some other composite type).
GetLatestRevision	Retrieve the date and time the available metadata information changed.
NotifyOnRevisions	Request the Equipment to send notification when any metadata availability changes.

E125 also defines an equipment-initiated operation for consuming clients.

Operation	Description
MetadataRevised	Notify the client that the metadata in the Equipment Model has changed if NotifyOnRevisions is enabled.

1.2.3.3 E132 Specification for Equipment Client Authentication and Authorization

E132 defines two related security features for Interface A messaging, Client Authentication and Client Authorization. Client Authentication determines how the client establishes a session before it can do anything else. Client Authorization manages what the client can access after the session is established. The Equipment must provide a Security Admin, a utility that provides administrative configuration, for setting up the Client Authentication and Authorization after installation in the fab. SEMI standard **E132.1 Provisional Specification for SOAP Binding for Equipment Client Authentication and Authorization (ECA)** maps the E132 standard into a specific SOAP/XML implementation.

In order for an Interface A Client to establish a session where it may use E125 or E134 service requests, the client must provide credentials and be authenticated and the Equipment must be in the ALLOWED Session Establishment state. Credentials include a client ID, an encrypted session key, and an encrypted client ID proof-of-identity key. Any attempts to use services requested before the authentication are rejected. Once the session is established, and then the client authorization becomes effective based on the client's credentials while establishing the session.

The Session interface includes the following client-initiated operations.

Class	Description
PersistSession	Request the Equipment to maintain the session, even after shutting down the Equipment.
SessionPing	A check to see if the Equipment is still active.
CloseSession	Request to terminate the session.

EstablishSession	Request to establish a new authenticated session and to set the client endpoint, the consumer for all notifications from the equipment.

Authorization is configured using Access Control Lists (ACL). An ACL is a collection of ACL Entries where each entry gives the client access permission to something in the interface. The E132 standard uses the terms principal, a client defined in the ACL, and privilege, permission to use an operation or access specific data. In practice, the easiest way to setup the ACL is by defining what E132 calls roles and then assigning clients to one or more roles. For example, it might be convenient to define "Operator", "Technician", and "Manufacturer" roles. Then Interface A client applications can be assigned to these roles to give them access to the appropriate access level.

Each time a client sends a privileged E125 and E134 service request, the Equipment must check whether or not the client is authorized. The ACL assigned to the client determines the set of available E125, E132, and E134 operations, the set of available metadata, and the access level to data collection plans defined by other clients.

E132 also defines some equipment-initiated operations for consuming clients.

Class	Description
SessionPing	Used by the equipment to check if the client is still active.
SessionFrozen	Notification to the client that the session will be frozen.
SessionClosed	Used by the equipment to close an active session.

The Security Admin interface includes the following operations. Only one active Security Admin session is allowed at a time.

Operation	Description
GetDefinedPrivileges	Request the list of all defined privileges.
GetACL	Request the list of all defined Access Control List entries
AddACLEntry	Add a new ACL entry
DeleteACLEntry	Delete an existing ACL entry
GetActiveSessions	Request the list of information on all active sessions
SetMaxSessions	Sets the maximum number of active sessions
GetMaxSessions	Requests the maximum number of active sessions

1.2.3.4 E134 Specification for Data Collection Management

The E134 standard defines several methods for Interface A Clients to acquire the data described by the E125 information. A client can request data ad-hoc where the requested set of data is returned immediately. Typically, however, the client will define Data Collection Plans (DCP) to configure the desired data gathering. Once DCP are activated, the Equipment continuously sends fire-and-forget Data Collection Reports (DCR) as the data becomes available. In order to optimize data transmission efficiency, clients can enable the optional buffering so that the DCR are buffered at the Equipment and transmitted periodically at the end of each buffering interval. If the ACL allows, clients can activate DCP defined by other clients. E134 also defines how to manage Data Collection Plans when either the Equipment or client shuts down and restarts so the data collection setup can be persisted. SEMI Standard **E134.1 Provisional Specification for SOAP Binding of Data Collection Management (DCM)** maps the E134 standard into a specific SOAP/XML implementation.

A Data Collection Plan can include any number and combination of Traces, Events, and Exceptions to report when the DCP is activated.

Traces provide a convenient mechanism for continuously or intermittently polling data at a constant rate. A Trace definition includes a data gathering frequency, a set of E125 data to collect, a start condition, and a stop condition. This configures the Equipment to poll the data at the specified frequency and send it to

the client. The optional start and stop conditions, an event or exception, determine when the data collection begins and ends. Otherwise, the Trace data collection begins as soon as it is activated and ends when deactivated or the specified number of reports are gathered and sent.

Events provide a convenient mechanism to receive notification when something important happens and optionally gather any desired data. An E125 state machine transition triggers an event. In the DCP, the client chooses E125 Parameters to include in the Event's DCR. Each time the Event occurs thereafter, a DCR is gathered and sent.

Exceptions notify the client when errors, warnings, or alarms occur on the Equipment. The Exception report includes data, but the data is fixed by the Equipment Supplier in the Equipment Model. Clients cannot select the attached Parameters like with Events.

Traces, Events, and Exceptions that are not listed in activated Data Collection Plans are not sent to any clients. Therefore, the Equipment Supplier determines how much data is available, but the clients determine how much data is actually gathered and reported. Bandwidth requirements depend on the number of clients, the number of activated DCP, the number of Events, Exceptions, and Traces in each DCP, the number of Parameters defined for each Event and Trace, the frequency of activated Events and Exceptions, and the frequency of Trace data collection. Therefore the Interface A bandwidth requirements are difficult to determine. Clients can potentially submit enough data collection requests to overload the Equipment's computer and affect the Equipment's throughput. Therefore E134 defines a Performance Status method to notify consumers of an overload condition and allows the Equipment to take appropriate action like disabling certain DCPs or all of the data collection.

E134 defines the following client-initiated operations:

Operation	Description
DefinePlan	Submit a Data Collection Plan
GetDefinedPlanIds	Request a list of all Data Collection Plan IDs.
GetPlanDefinition	Retrieve the definition of a Data Collection Plan
ActivePlan	Activate the defined DCP
GetActivePlanIds	Request a list of all activated DCP IDs
DeactivatePlan	Deactivate the DCP
DeletePlan	Delete a DCP
GetParameterValues	Ad-hoc request to retrieve the current values of one or more E125 parameters.
GetObjTypeInstanceIds	Request a current list of unique instance IDs for one or more E39 ObjTypes.
GetCurrentPerformanceStatus	Retrieve the current Equipment performance status.

E134 also defines the following equipment-initiated, Fire-and-Forget operations.

Operation	Description
NewData	Data Collection Report from an active DCP
PerformanceWarning	The Equipment detected performance degradation. .
PerformanceRestored	The Equipment has detected a return to normal conditions.
DCPDeactivation	Notification that an active DCP for that consumer is deactivated.
DCPHibernation	Notification when one or more persisted DCP are put into the hibernation state as part of Equipment shutdown.

1.2.3.5 E164 Specification for EDA Common Metadata

The purpose of the E164 Standard is to promote commonality among implementations by defining common representations and conventions of equipment metadata based on E125 Specification for

Equipment Self-Description. The E164 Standard contains the requirements for each of the content Standards (E30, E40, E87, E90, E94, E116, E148, and E157)

1.2.4 Frequently Asked Questions (FAQ)

1. Where can I get more information?

More information is available from the following websites.

- SEMI www.semi.org
- SEMATECH www.sematech.org
- ISMI www.ismi.sematech.org
- Cimetrix www.cimetrix.com

2. Will the Interface A standards change over the next few years?

The Interface A standards are relatively new and have been prototyped only in a narrow set of applications. As the standards become adopted by a wide variety of Equipment Suppliers and utilized by IC Makers in diverse applications, the Interface A standards will undergo significant scrutiny. The standards will adapt to the needs of the semiconductor community and mature over time. Anyone who implements the Interface A standards must plan to dedicate resources to follow the standards and migrate with the inevitable changes.

3. Why use Interface A instead of SECS/GEM for data gathering?

Here are a few reasons. SECS/GEM has its merits, too, but they are not listed here.

SECS/GEM	Interface A
SECS/GEM requires support for only one client connection. IC Makers cannot run several data gathering applications at the same time without an infrastructure to share the data.	Interface A requires support for multiple concurrent clients. Independent clients can simultaneously use Interface A.
SECS/GEM is only partially self-describing and therefore relies on good documentation. IC Makers have complained that the documentation is often poorly maintained and poorly written.	Interface A is self-describing through the E125 standard's metadata including a listing and description of all available data. Inherently, it should be synchronized with the actual equipment configuration.
SECS/GEM data is relatively flat and unorganized. The IC Maker must study the documentation, hardware, software, and processing to understand how to organize the data.	Interface A presents the data in a hierarchy, organized by the major hardware components.
Data in a SECS/GEM message is highly structured and relatively inflexible.	Interface A uses XML; therefore the inherently design to accommodate additional metadata.
SECS/GEM is only used in a few industries; therefore there are a limited number of experts in the world.	SOAP/XML and HTTP are the backbone of most Internet and Intranet applications. There are many programmers worldwide that are familiar with this technology.
There are relatively few software packages in the world to deal with SECS/GEM technology; most are only known to the Semiconductor industry.	There are a tremendous number of software packages worldwide from many industries that can handle SOAL/XML and HTTP technology.

Interface A uses many of the same concepts as SECS/GEM. Here is a mapping between the similar concepts and technologies.

SECS/GEM	EDA
Status Variables, Equipment Constants, and Data Variables	Parameters

Alarms	Exceptions
Collection Events (S6,F11)	Events in DCP
E39 Objects	SEMIObjType and Instance IDs
Trace Data Collection	Traces in DCP
Reports	Data Collection Plans/Reports
State Machines	State Machines
Enable/Disable Collection Events	ActivatePlan/DeactivatePlan
Define & Link Reports (S2,F33 & S2,F35)	DefinePlan

4. Is the SOAP/XML over HTTP fast enough?

Yes, if it is implemented effectively. Cimetrix has run multiple tests and prototypes that proved adequate performance. The results have been presented in industry public forums.

5. What data should be made available through EDA?

Ultimately, each Equipment Supplier must negotiate requirements from the IC Makers to determine exactly what data they need. Until IC Makers develop and deploy systems that use EDA, it is difficult for them to establish clear requirements. Nevertheless, here are some guidelines for Equipment Suppliers:

- As many sensors and actuators as possible
- Processing or inspection setup information, states, sub states and results. Provide enough information to track equipment utilization, feed APC systems, and support e-Diagnostics.
- Material tracking
- Provide all data as near as possible directly from the source. This can require some software and hardware architectural changes.

6. What should be reported when the requested data is not available?

E134 defines a NoValue class with a ValueNotAvailable value for the NoValueReasonEnum enumeration to handle this situation.

7. How can a client gather SEMIObjType information, since the object IDs are not in the Equipment Model?

SEMIObjType objects can be created and deleted dynamically or exist statically on the Equipment. Either way, Interface A Clients manage them using the same operations. Dynamic objects include carriers and substrates. Static objects include substrate locations not in the carrier and Equipment Performance Tracking modules. First, a client should use E125 operation GetSEMIObjTypes to query the list of available SEMIObjType classes. These are part of the Equipment Model. Then use E134 operation GetObjTypeInstanceIds to get full instance ID information for the current set of objects. With these instance IDs, the client can use any of the data gathering operations such as Define Plan or GetParameterValues to query the object's attribute data. If the object gets deleted, then the client will receive a NoValue with a NoSuchParameter reason enumeration.

8. How does an EDA client receive data from the equipment?

The client must implement a web server with a single URL that implements the equipment-initiated E125, E132 and E134 operations. During the EstablishSession operation, a client passes its web server URL to the equipment. When data collection plans are activated, the equipment uses the NewData operation to pass the data collection reports to the client.

9. How can I implement EDA?

The most cost effective solution is to purchase a solution from a third party supplier like Cimetrix. Cimetrix offers a state-of-the-art EDA development package called CIMPortal Plus for Equipment Suppliers or IC Makers. When choosing a solution, it is important to consider the following technical and commercial challenges:

a. Evolving Standards

There is no question that the standards will change several times over the next few years. A reliable solution not only complies to today's standards, but is designed to anticipate and accommodate future changes. The software team must be committed to become experts in the standards, actively track the changes, and provide timely upgrades.

b. Data Source Management

You must ask these crucial questions; "Where does the data reside?" and "What is the most efficient way to get the data into EDA?" All of the data will not be in one location. The Interface A solution must have the ability to channel data directly from different sources.

c. Data Integrity

The EDA solution must manage data integrity. Inevitably, there is a time lag between data gathering and data reporting. Some data must be refreshed in every data collection report. Other data can be cached for limited periods of time. Still other data can be perpetually cached. In order to maximize EDA performance, the solution must be able to manage data integrity and make correct data-caching decisions.

d. Performance

ISMI has reported in public forums that IC Makers expect Interface A to achieve data rates of 50+ variables per chamber at rates up to 10 Hz. These rates are only possible with a software architecture designed for high performance.

e. Software Quality

Over the next short years, EDA will become an increasingly valuable feature to the IC Makers. It will become one of the critical features that IC Makers will validate before making a decision to purchase a tool. The Interface A solution must achieve the highest software quality and performance standards. A second-rate solution will jeopardize equipment sales.

f. Experience

A professional EDA solution requires experience. Experience means running on real machines in production for long periods of time. Experience means prototypes and demonstrations with simulators and real machines. Experience gives a solution the maturity to implement not only the requirements, but to include the intent of the standards and features beyond the requirements to manage IC Maker expectations.

g. Customer Dedication

Commercial solution providers must be committed to providing effective customer support. Check references and review past experiences. Ensure that the company is truly dedicated to its customers.

1.3 Interface A Compliance

The following sections are SEMI EDA standard compliance statements for CIMWeb. They also provide a quick summary of the developer's responsibility to support each line item. Please refer to the CIMPortal Plus Release Notes to determine the version(s) of supported SEMI standards.

- E120 SPECIFICATION FOR THE COMMON EQUIPMENT MODEL (CEM)
- E125 SPECIFICATION FOR EQUIPMENT SELF DESCRIPTION (EqSD)
- E132 SPECIFICATION FOR EQUIPMENT CLIENT AUTHENTICATION AND AUTHORIZATION
- E134 SPECIFICATION FOR DATA COLLECTION MANAGEMENT
- E138 COMMON COMPONENTS
- E128 XML MESSAGE STRUCTURES
- E164 COMMON METADATA

1.3.1 E120 Compliance

E120 addresses equipment structure.

1.3.1.1 General CEM Requirement Compliance Table

#	CEM Requirement	CIMPortal Plus supported	Work to do
1	An Equipment instance shall be defined using the CEM Equipment class	Yes	Start your equipment model with an Equipment node or run the New Model Wizard in EMDeveloper
2	When other equipment component instances are defined, they shall meet all requirements for the corresponding CEM classes. Please note that this specification does not require that all CEM classes be used in each equipment implementation. Instead, it requires that all requirements for a class be met whenever it is used.	Yes	Model your equipment structure in EMDeveloper and be sure to fill in all properties of the Hierarchy Structure nodes.
3	All object definitions that have been based on CEM classes shall be made available electronically to the factory host.	Yes	Deploy a model package. This data is provided by CIMWeb through the web interface after package deployment.
4	If the equipment provides a positional reference for an equipment component, the Locator form shall be used.	Yes	Nothing.
5	Association Navigability shall be supported	Yes	Nothing. This is supported through the SourceID concept in the E120 metadata.

1.3.1.2 CEM Class Compliance Table

CEM Class Requirement	Class/Superclass Definition	Navigable Via Association	CIMPortal Plus supported	Work to do in the model
Equipment	Nameable EquipmentElement ExecutionElement Equipment	Extension Module Subsystem IODevice MaterialLocation SoftwareModule	Yes	Equipment instance is the highest level of representation for a piece of semiconductor equipment. It defines the equipment as a whole.
Module	Nameable EquipmentElement ExecutionElement Module	Extension Module Subsystem IODevice MaterialLocation	Yes	Module represents major components of the equipment,

		SoftwareModule		such as process chambers or metrology locations. A *Module* adds value to the material it processes.
Subsystem	Nameable EquipmentElement Subsystem	Extension Subsystem IODevice MaterialLocation SoftwareModule	Yes	A Subsystem is not capable of processing material.
IODevice	Nameable EquipmentElement IODevice	Extension SoftwareModule	Yes	IODevice models sensors, actuators, or intelligent actuator/sensor devices.
MaterialLocation	Nameable MaterialLocation	Extension	Yes	Material locations hold substrates, carriers, and consumables.

1.3.2 E125 Compliance

E125 adds data elements such as parameters, events, exceptions, and state machines to the equipment structure from E120.

1.3.2.1 Equipment Self-Description Compliance Statement

Fundamental Requirements	Implemented using SEMI technology mapping	Implementation complies with ESD specification and technology mapping	ESD data associated with all information sources and items accessible via data acquisition are described	Work to do
Metadata Access	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Deploy a package.
Management of Equipment Metadata	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Deploy a package.
Equipment Physical Structure	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Deploy a package.
Equipment Node Description	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Deploy a package.
Parameters	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add parameters to your model.

Types	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add Parameter Type Definitions to your model
Units	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add Units to your model.
Exceptions	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add exceptions to your model.
State Machines	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add State Machines to your model.
ObjTypes	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Yes, CIMWeb provides this.	Add SEMI Objects to your model.

1.3.3 E132 Compliance

E132 provides connection session and security concepts.

1.3.3.1 Fundamental Requirements

Requirement	CIMPortal Plus supports	Work to do
Authentication services and state model	Yes	Nothing
Authorization	Yes	Nothing
Session and ACL administration	Yes	Define Interface A clients in Configuration Manager
Authenticated session management	Yes	Nothing
All communication and authenticated session state models	Yes	Nothing

1.3.3.2 Optional Capabilities

Requirement	CIMPortal Plus supports	Work to do
Equipment console administration interface	Yes. Client and ACL administration through the Configuration Manager. Session administration through the CIMPortal Plus Control Panel or CIMWeb Admin pages	Nothing

1.3.4 E134 Compliance

E134 covers data collection.

1.3.4.1 Fundamental Requirements

Fundamental Requirements	Implemented using SEMI technology mapping	Implementation complies with specification	Implementation complies with technology mapping	Work to do
DCM Interface	Yes	Yes	Yes	If a parameter is only valid in the context of an event, be sure to only put it in the EventMap for

				that event in your model and set its isTransient property to true. CIMWeb will use this information to generate an InvalidParameterRequest error with invalidContext set to true if an Interface A client tries to use that parameter outside of that event. Also, be sure that the isTransient property of all the parameters in the model is set correctly. CIMWeb will not allow a client to include isTransient = true parameters in a trace request. The same InvalidParameterRequest error is returned.
DCP Privileges	Yes	Yes	Yes	Use Configuration Manager to define Interface A clients and assign privileges to them.
DCP Definition	Yes	Yes	Yes	Nothing, CIMWeb provides this.
DCP State Models	N/A	Yes	N/A	Nothing, CIMWeb provides this.
Operational Performance Monitoring	Yes	Yes	Yes	Configure CIMWeb performance monitoring as needed. Use the AppDCIM to generate PerformanceWarning and PerformanceRestored notifications programmatically.
Data Representation	Yes	Yes	Yes	CIMWeb provides this based on the definitions in the model you create.
DCP Notifications	Yes	Yes	Yes	Nothing, CIMWeb provides this.

1.3.5 E164 Compliance

E164 addresses the EDA Common Metadata. Compliance to this standard is determined by running the ISMI Metadata Conformance Analyzer against the equipment model developed in CIMPortal Plus.

1.4 Glossary

Term	Description
ADD	Architecture and Design Document contains the design of the application.
APC	Advanced Process Control Application An application is a logical functional grouping as

	perceived by a user. An application may be synonymous with a component.
ASP.NET	ASP Stands for Active Server Pages. It is a Microsoft developed language that allows a programmer to develop interactive web pages on a Windows web server. ASP.NET is later version of ASP that runs in the .NET framework. ASP is an interpreting language whereas ASP.NET is a compiled language.ASP.NET is not fully backward compatible with ASP.
CEID	Collection Event ID - a GEM term
CEM	E120-0703 Common Equipment Model Component is a generic term used to define a portion of some system. It is a fully functional, self-contained piece of functionality. An application is a Component.
Certificate Generation	When you generate a request for a new certificate, the information in that request is first passed from the requesting program to CryptoAPI. CryptoAPI will pass the proper data to a program known as a cryptographic service provider (CSP) that is installed on your computer or on a device accessible to your computer. If the CSP is software-based, it will generate a public key and a private key, often referred to as a key pair, on your computer. If the CSP is hardware-based, such as a smart card CSP, it will instruct a piece of hardware to generate the key pair. After the keys are generated, software CSP encrypts and then secures the private key in the registry on the computer. A smart card CSP stores the private key on a smart card and the smart card controls access to the key. The public key is sent to the certification authority, along with the certificate requester information. Once the CA verifies the certificate request according to its policies, it will use its own private key to create a digital signature in the certificate and then issue it to the requester. The certificate requester will then be presented with the certificate from the CA and the option to install it in the appropriate certificate store on the computer or hardware device.
CIMPortal Plus Control Panel	Refers to the CIMPortal Plus GUI utility that performs administrative duties. This tool includes the functionality to perform Security Administration.
collection frequency	The rate at which the collection of one or more data values is performed. This is not the same as the frequency with which the equipment internally samples these data from its components
collection result	The set of values obtained during trace data collection
data source	A physical or logical entity associated with the equipment that is capable of providing data values independently of other equipment entities
DCIM	Data Collection Interface Module. This is a Cimetrix product that performs actual data collection duties for CIMPortal Plus. See section 2.15 for more information.
DCP	3509B-1.9 Data Collection Plan
DCR	Data Collection Report
DLL	Dynamic Link Library
DSH WSDL	Data Source Handler Web Service Interface
ECAA	3507-1.7 Equipment Client Authentication & Authorization
EDA	PR008-0703 Equipment Data Acquisition
EDA client	A client that makes use of the web services defined by EDA. Also known as an Interface A client.
EDM	Equipment Data Model
Equipment Modeler	The CIMPortal Plus GUI Utility used to build a model of the equipment, namely to create equipment nodes (E120) and equipment description (E125) metadata.
ESD	E125-0703 Equipment Self Description
event source	a physical or logical entity associated with the equipment that is capable of generating events independently of other equipment entities
exception source	a physical or logical entity associated with the equipment that is capable of generating exceptions independently of other equipment entities
FC	Fault Classification

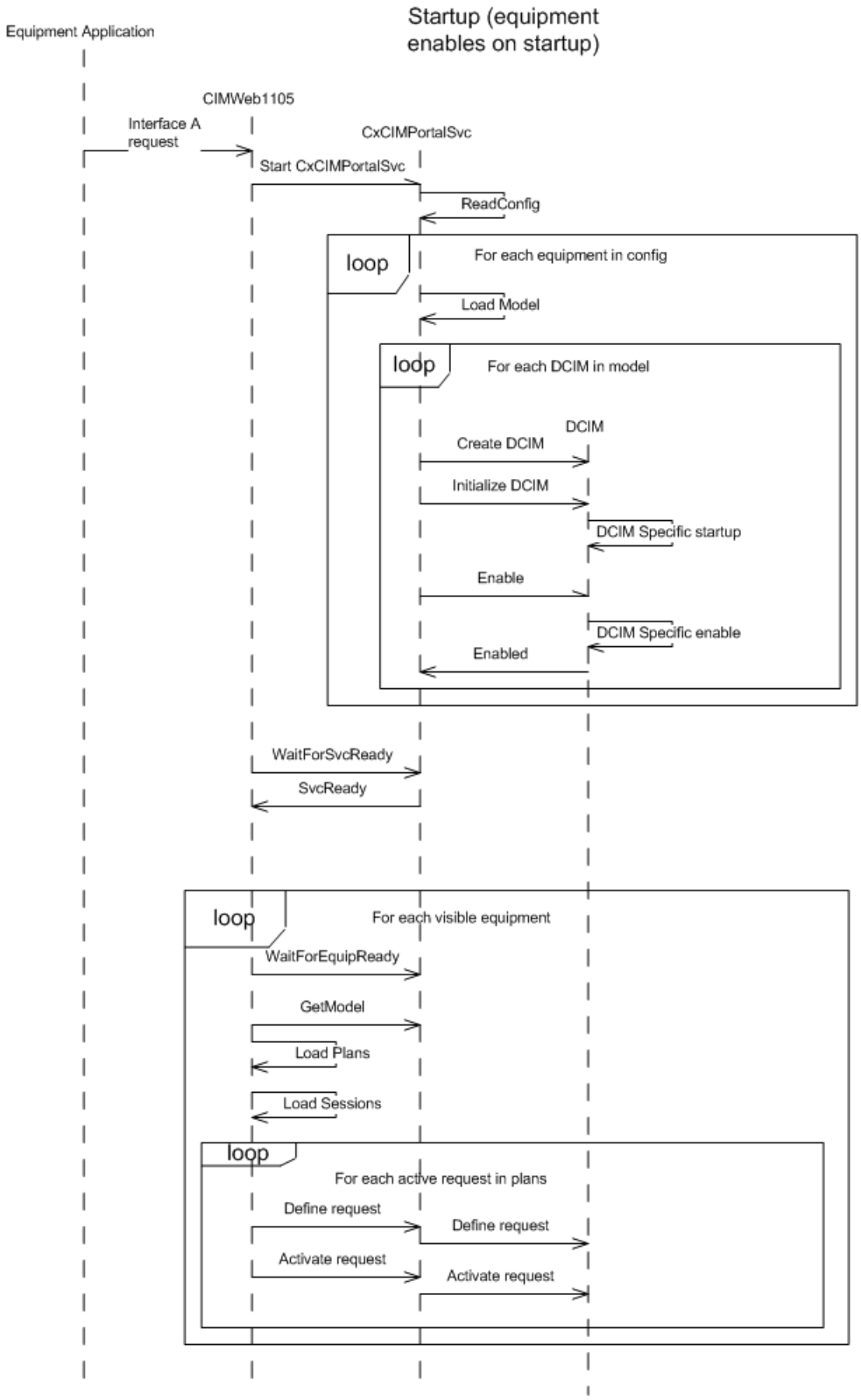
FD	Fault Detection
FICS	Factory Integration and Control System
FP	Fault Prediction or Prognosis
GUI	Graphical User Interface. Also known as Human-Machine Interface
IIS	Internet Information Service. It is a Microsoft web server product. Current version is 6.0 running on Windows 2003 Server edition.
parameter source	a physical or logical entity associated with the equipment that is capable of providing parameters independently of other sources. This term may be used interchangeably with 'data source'.
PCS	Process Control System
PKCS#12	The Personal Information Exchange Syntax Standard developed and maintained by RSA Data Security, Inc. This standard specifies a portable format for storing and transporting a user's private keys, certificates, and miscellaneous secrets.
S-HTTP	A protocol for transmitting data securely over the World Wide Web. S-HTTP is designed to transmit individual messages securely, whereas SSL creates a secure connection over which any amount of data can be transmitted securely.
SOAP	SOAP stands for Simple Object Access Protocol. It is a simple XML based protocol to let applications exchange information over HTTP. In other words, SOAP is a protocol for accessing a Web Service.
source id	a name or other token that uniquely identifies a specific origin or producer of information from among possible sources
SPC	Statistical Process Control
SSL	Short for Secure Sockets Layer, a protocol developed by Netscape for transmitting private documents via the Internet. SSL creates a secure connection between a web client and a server, over which any amount of data can be sent securely. SSL works by using a private key to encrypt data that's transferred over the SSL connection. Both Netscape Navigator and Internet Explorer support SSL and many Web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with https instead of http.
SUGX	E121-0703 Style and Usage Guides for XML URL Link Any item in a page that can be clicked to access a specific HTTP address
VID	Variable ID - A GEM term
WSDL	WSDL stands for Web Services Description Language. It is an XML-based language for describing Web services and how to access them.
X.509	A widely used standard for defining digital certificates. X.509 is actually an ITU Recommendation, which means that it has not yet been officially defined or approved for standardized usage. As a result, companies have implemented the standard in different ways. For example, both Netscape and Microsoft use X.509 certificates to implement SSL in their Web servers and browsers. But an X.509 Certificate generated by Netscape may not be readable by Microsoft products, and vice versa.
XMS	E128-0703 XML Message Structure

2. Using CIMPortal Plus

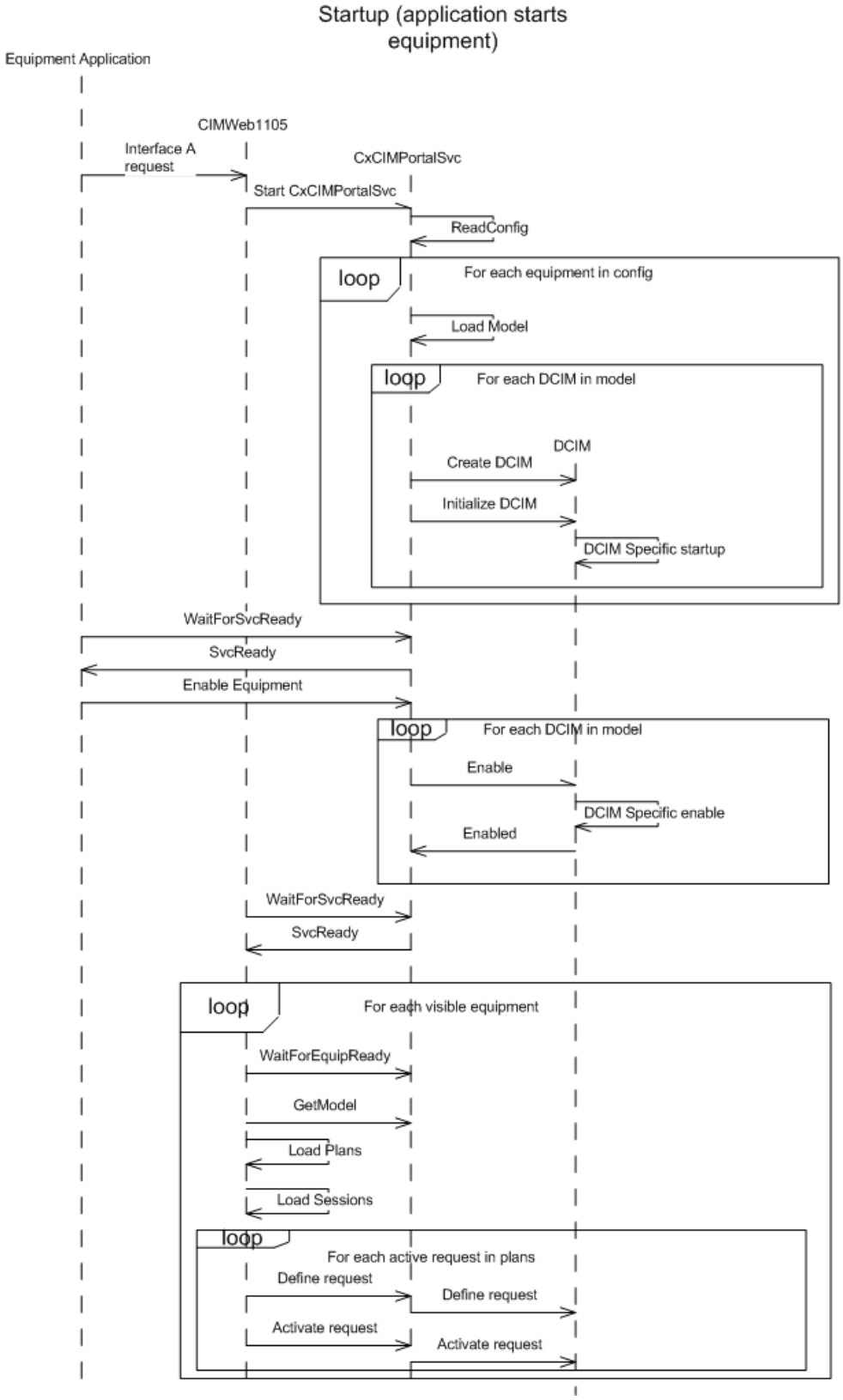
2.1 Startup and Shutdown

There are a number of different ways CIMPortal Plus Service can be started up. CIMPortal Plus Service can be started by, a client application, the operating system, or by a user. CIMPortal Plus Service can be started by any way an application or system service or out-of-process COM server can be started. In addition, CIMPortal Plus Service can automatically enable deployed equipment or it can wait for the equipment to be enabled by another process.

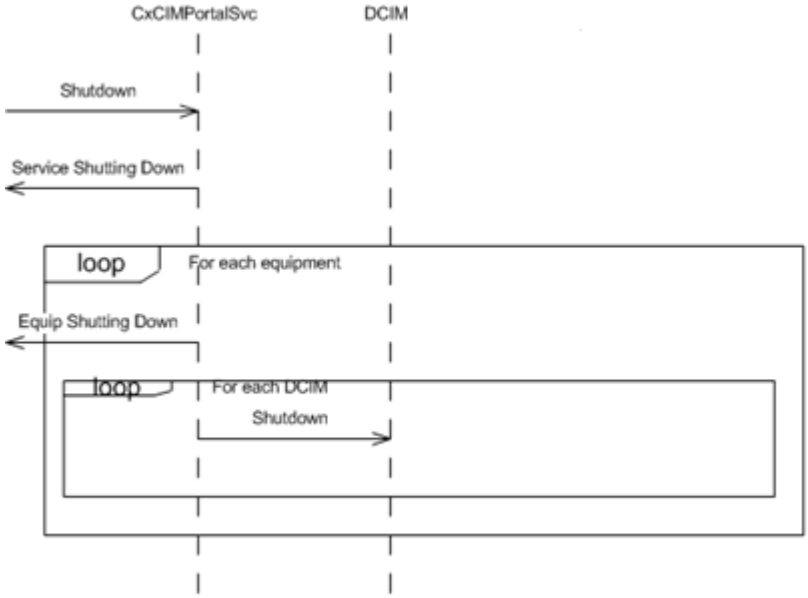
The following sequence diagram is for CIMPortal Plus Service starting up with equipment that is automatically enabled:



The following sequence diagram is for CIMPortal Plus Service starting up with equipment which will be enabled by a client process:



The following sequence diagram is for CIMPortal Plus Service during shutdown:



2.2 CIMPortal Plus Control Panel

This section gives an overview on the CIMPortal Plus Control Panel. The GUIs are accessed by selecting the **CIMPortal Plus Control Panel** option from the start menu under **Cimetrix | CIMPortal Plus**.

The Control Panel consists of a dialog with multiple tabs. You must enter a client name and password with CIMPortal Plus service administrator privileges to work in the Control Panel. After installation, there is a default account called "admin" with password "admin" is available. This should be changed during the OEM configuration.

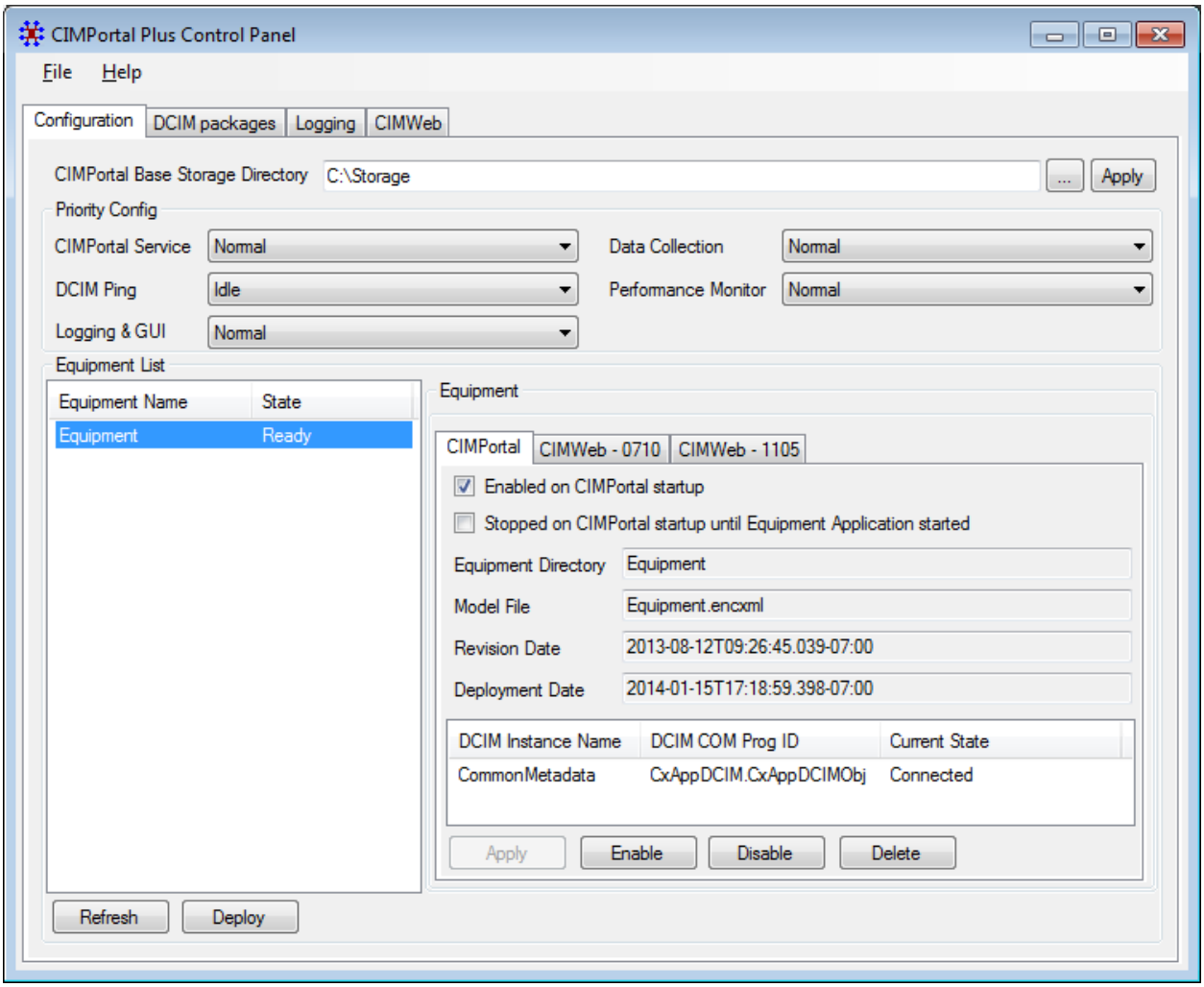
The Control Panel has the following tabs: Configuration, DCIM Packages, Logging and CIMWeb. Each tab is discussed in its own section below.

The menu has pretty much one item: File | Shutdown CIMPortal Plus. This will display a confirmation prompt. If the shutdown is confirmed, the service will be shutdown.

Anytime the Control Panel loses connection with the service, including when it's shutdown, the user will be asked if an attempt should be made to restart the service. If this is answered affirmatively, the service will be restarted when the user logs in. If the user does not choose to restart the service, the Control Panel will be closed.

2.2.1 Configuration tab

The Configuration tab looks like this:



On the top are several properties that are general for all of CIMPortal Plus.

The "CIMPortal Base Storage Directory" is the location where all the equipment packages are installed. This includes the DCIM packages and model files. The storage directory can be changed by typing the new storage directory in the edit field or by browsing with the "..." button. This change takes effect when the Apply button is pressed. Changing the storage directory is not commonly done.

There are several key background threads that can have their CPU priority individually set. Typically they should be set to the default value of Normal. Setting to other values may impact system performance and should only be done in a test environment until their impact on equipment operation is validated.

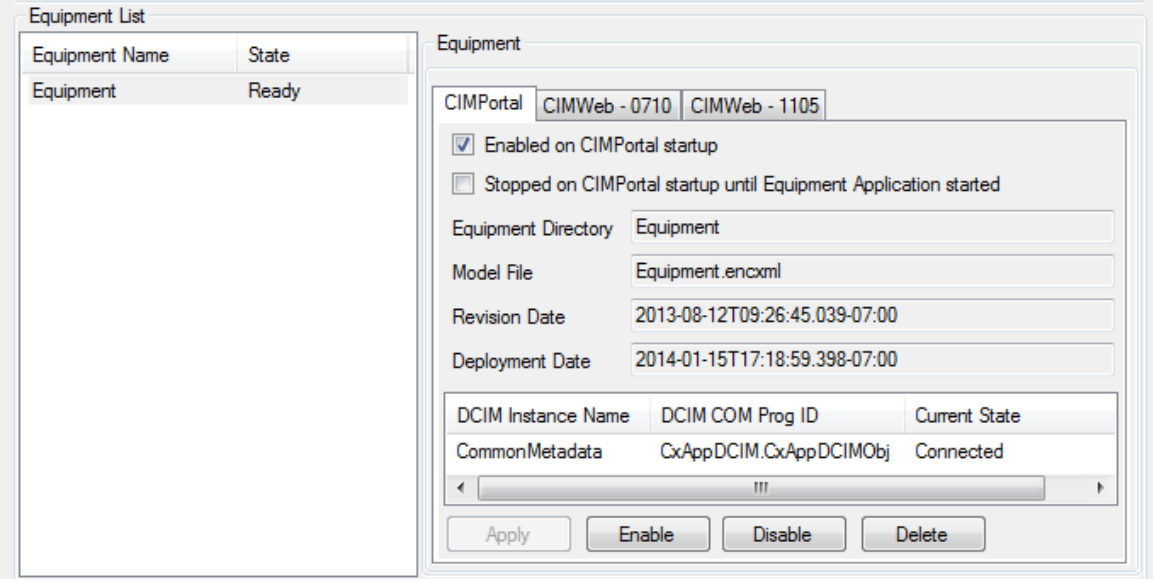
On the bottom of the tab there are a couple buttons. Click the "Refresh" button to refresh the information shown on this tab. Click the "Deploy" button to add new equipment.

2.2.2 Equipment configuration

The equipment list shows the equipment that has currently been deployed to CIMPortal Plus. (For more information, see Deploying Packages, section 2.2.2.5.) The list shows the equipment name and its state. The pane to the right shows additional information about the selected equipment. The CIMPortal Plus tab exists for all deployed equipment. Additional tabs may be displayed based on additional installed modules. By default CIMWeb is the only installed module and adds a tab for each supported EDA standard. Currently supported standards are freeze versions 1105 and 0710. The CIMWeb tabs contain the same type of information, just different values for the different versions.

2.2.2.1 CIMPortal tab

This tab shows a couple configuration parameters and read-only status information.



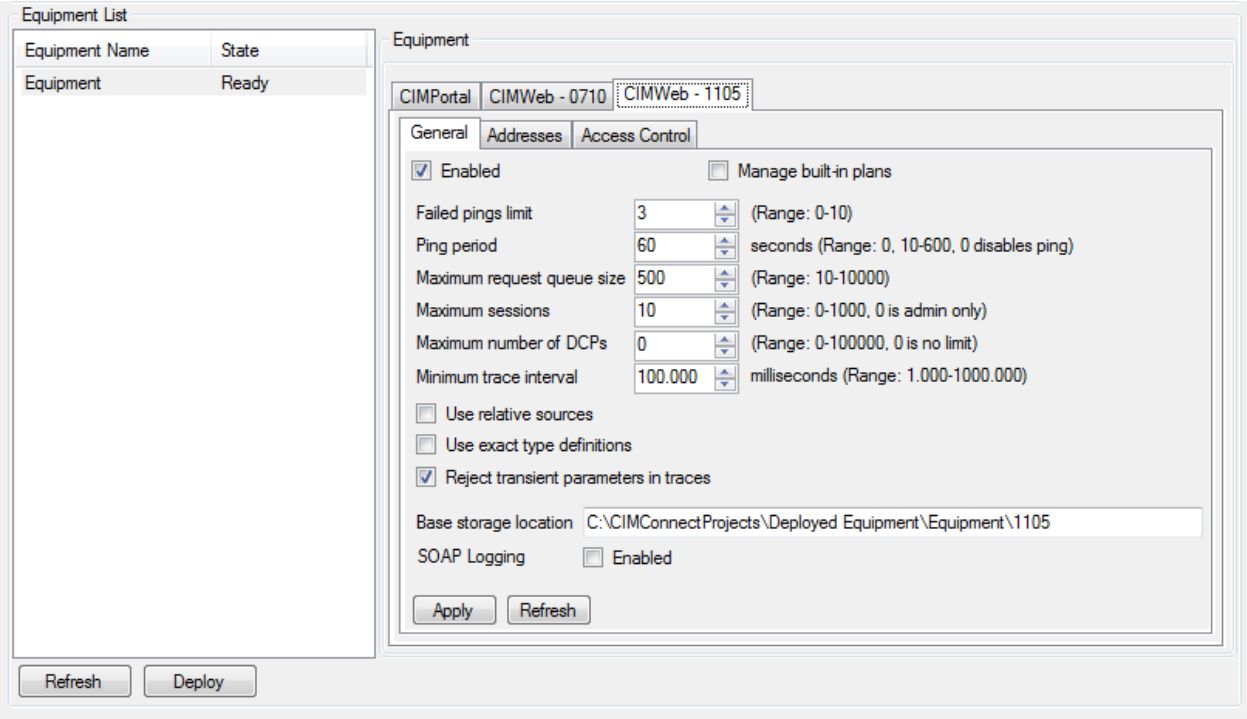
The "Enabled on CIMPortal startup" checkbox causes the equipment (and its DCIMs) to be started and enabled during CIMPortal startup. If this box is not checked, the equipment will need to be started by an application before it will be available.

If the equipment should not start up automatically when CIMPortal Plus Service is started, check the "Stopped on CIMPortal startup..." checkbox. This box should be checked if an application needs to modify the equipment model before the equipment is started.

The equipment directory is located in the CIMPortal base storage directory and the model file is located in the equipment's base directory.

The DCIM instance list shows the DCIM instances that are used by the equipment model. The list shows the DCIM instance name, the COM ProgID, and the current state of the DCIM instance.
At the bottom there are several buttons. Click the "Apply" button to save changes to the selected equipment's properties. Click the "Enable" button to enable the selected equipment. Click the "Disable" button to disable the selected equipment. Click the "Delete" button to remove the selected equipment.

2.2.2.2 CIMWeb equipment General settings



The "Enabled" check box indicates if this EDA standard is enabled. If it's not enabled, users will not be able to connect to the addresses on the Addresses sub-tab. If it is enabled, connections will be accepted and sessions may be established.

The "Manage built-in plans" check box controls if a client named `urn:semi-org:equipment` will be allowed to log in and manage (define/delete) built-in plans. This setting is reset every time CIMWeb is restarted. (See section 2.11 for details about managing built-in plans.)

The "Failed pings limit" setting specifies the maximum number of consecutive Ping failures. Once this limit is reached, a client's active DCPs will automatically be deactivated.

The "Ping period" setting specifies how often the CIMWeb server will ping clients. Set this value to 0 to turn off the ping.

The "Maximum request queue size" indicates how many client requests may be pending at any given point in time.

The "Maximum sessions" indicates how many client sessions may be concurrently established. An error is reported to clients that try to connect after this limit is reached. Equipment that is already in use by Interface A clients, and if the maximum sessions value is decreased to a number less than the current open sessions, CIMWeb will not automatically disconnect sessions. It is the responsibility of the Administrator to use the CIMWeb Current Sessions tab to disconnect sessions (see section 2.2.5.5).

The "Maximum number of DCPs" is used to limit the maximum number of DCPs that can be defined. Set this value to '0' for no limit.

The "Minimum trace interval" indicates the smallest trace interval a DCP can be set to.

The "Use relative sources" check box controls how parameter references are resolved. E125 describes relative sourceIDs (empty string) for parameter refs in SEMIObjTypes and nodes if the parameter is in the same E120 node. Using relative sources will leave the sourceID blank; otherwise the absolute sourceID will always be used. Generally, relative sources are not used.

The “Use exact type definitions” setting changes the way CIMWeb handles GetTypeDefinitions service for Structure and Array PTD's. When it is not checked (old behavior, default setting) Structure's fields and Array's type definitions will contain a copy of the Value type definition. When it is checked these elements will be represented as a reference (ParameterTypeDefinitionReferenceType) with the name of the base PTD.

The “Reject transient parameters in traces” will cause an error to occur when defining a DCP if any of the parameters in the trace are transient. This is version dependent and may not be visible if the version does not allow it.

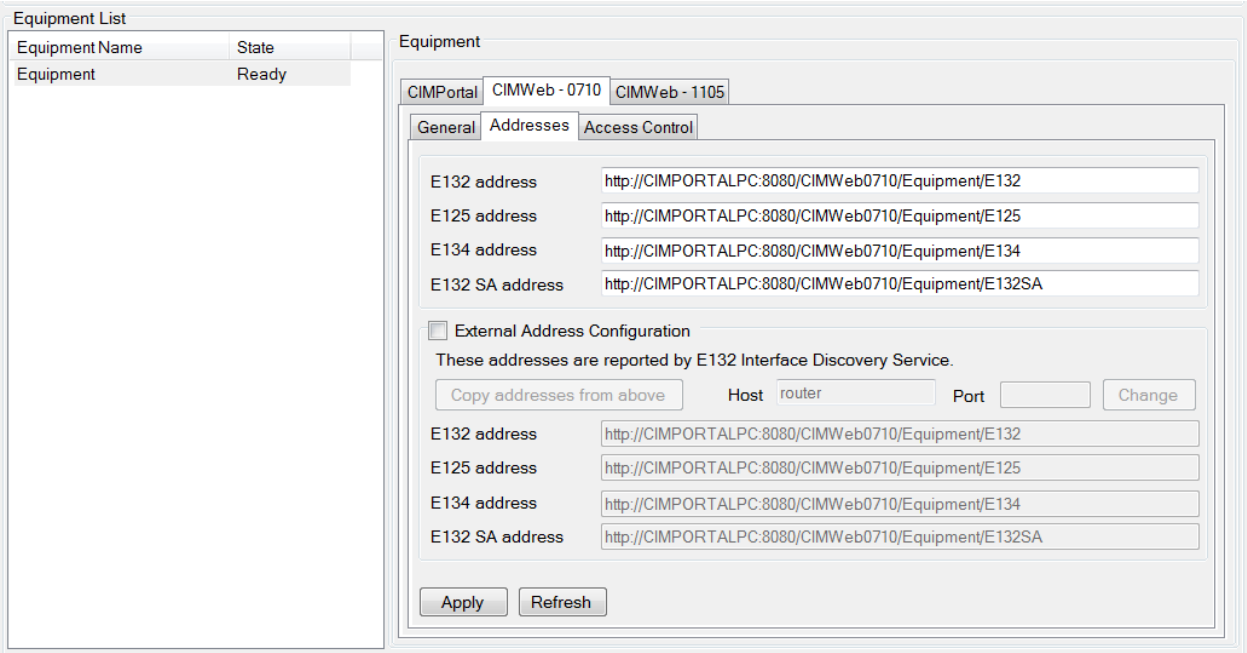
The “Base storage location” indicates where version specific configuration files will be placed. By default it’s the concatenation of the CIMPortal Base Storage Directory with the Equipment Directory with the version number.

The “SOAP Logging” setting indicates if SOAP logging is enabled for this equipment/version combination. See section 2.2.5.4 for more information.

Finally, press the “Apply” button to apply and save changes. Press the “Reset” button to undo any changes made since the settings were last applied.

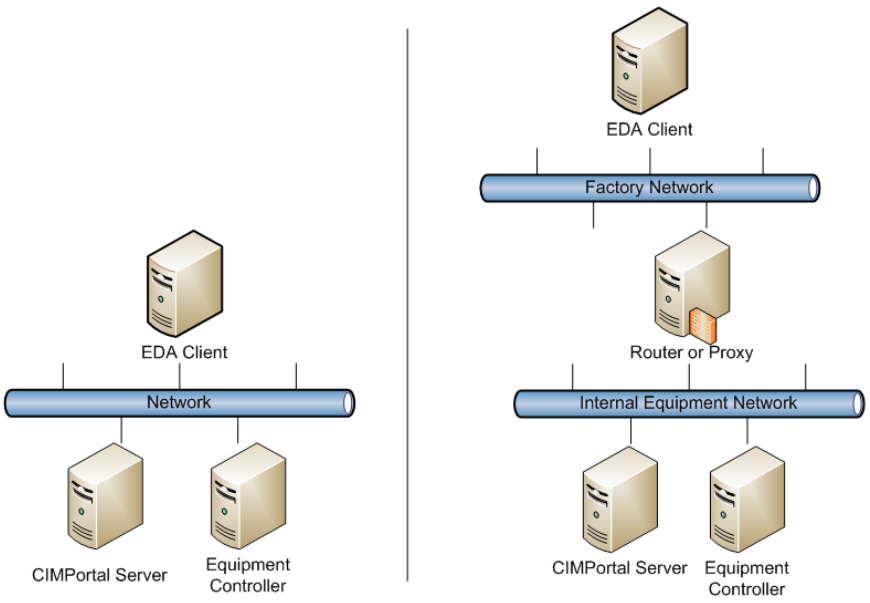
2.2.2.3 CIMWeb equipment Addresses settings

Addresses must be specified to define listening endpoints that are used by CIMPortal for incoming EDA requests. Equipment will have different addresses for each EDA version.



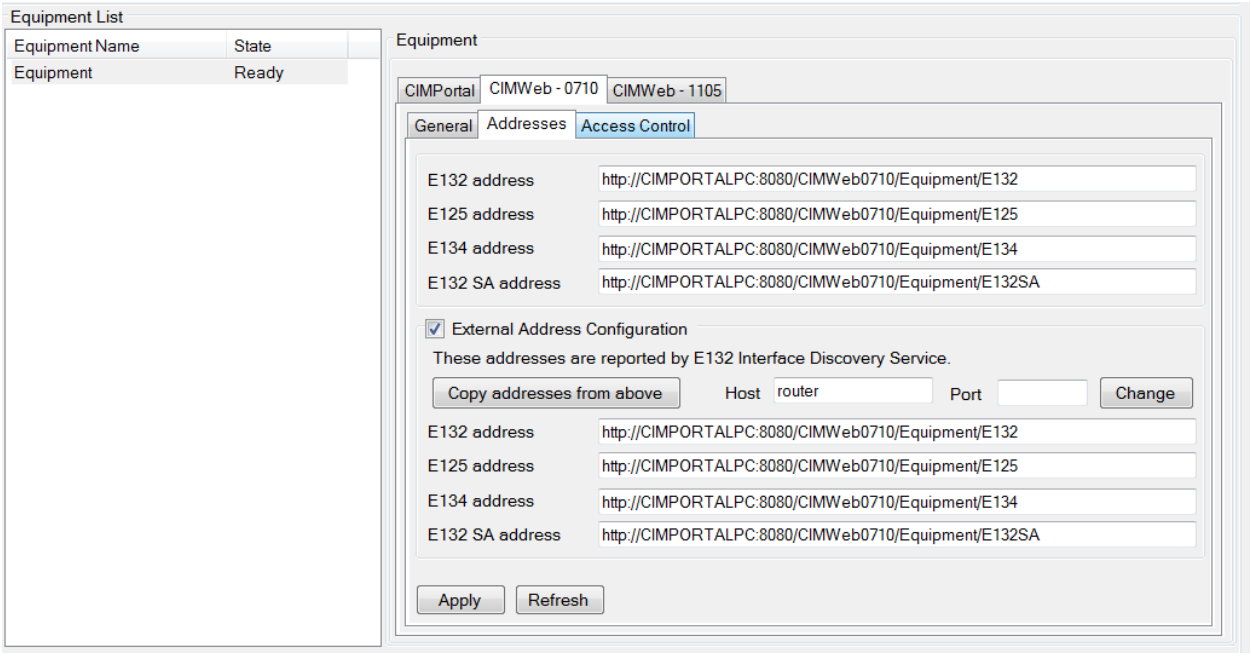
Default addresses for E125, E132, E132SA, and E134 services are automatically provided. Since multiple equipment and multiple EDA versions may be deployed, both the equipment name and version values are included in the address by default. If it is known that a particular installation will not have this need, these values may be simplified to shorten the URL.

Address configuration depends on the network topology. In the case where an EDA client is connected to the same network as the computer with CIMPortal, configuration of the default addresses is sufficient.



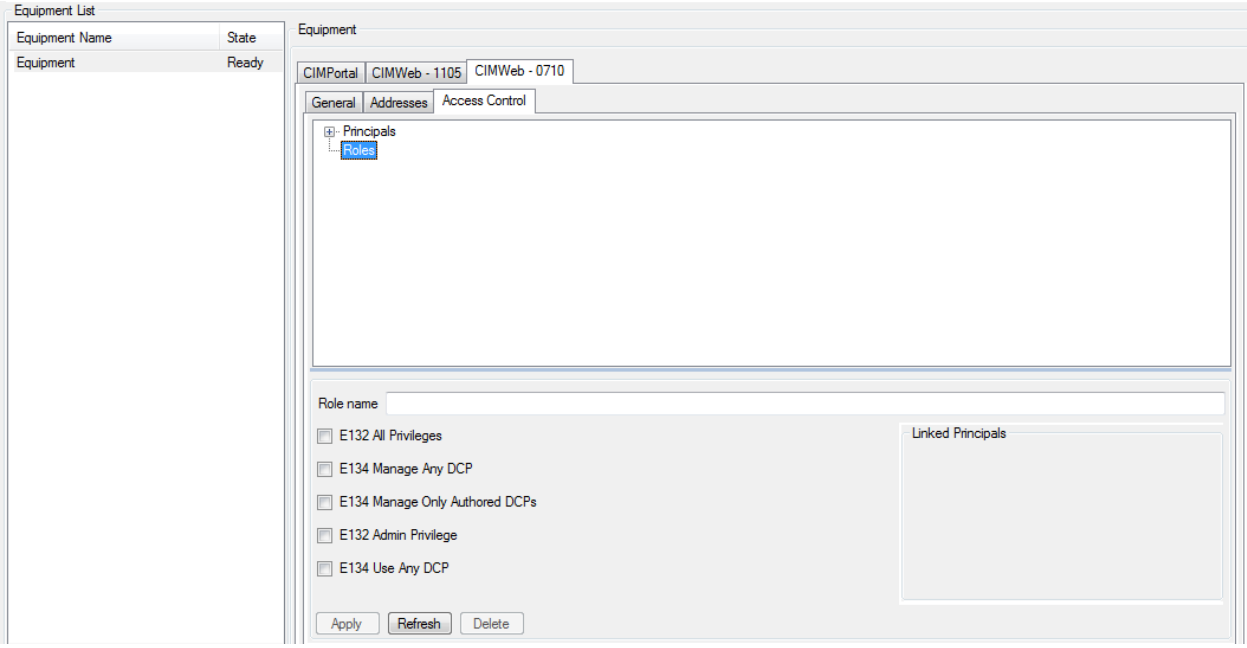
In the case where an EDA Client is connected to a network external to the computer with CIMPortal, a router or proxy device is used. The EDA client will use the E132 Interface Discovery service to obtain the service addresses. Therefore the addresses must include the name or IP address of the router/proxy device. The device must redirect requests from the external address to the address used on the internal network. How to configure the device in this way is beyond the scope of this document.

To configure the service addresses for an external EDA Client, check the box "External Address Configuration". The fields will become enabled. By default the addresses are the same as the internal addresses above. Generally the host name portion should be modified to match the name or IP address of the router or proxy. The Change button will apply the host and port values to all the service addresses that follow.



Click on Apply to apply any changes to this screen. The Refresh button reverts any changes to the settings previously stored.

2.2.2.4 CIMWeb Access Control settings



This tab allows administration of access control for Interface A clients. Prior versions of CIMPortal used the Configuration Manager to control access to both CIMPortal and CIMWeb, but Control Panel should be used instead as Configuration Manager has been obsoleted and will be removed in a future release. There are new rules in the Interface A standards that impose new, improved restrictions on how access is controlled. Additionally, there are now multiple interfaces for each equipment that need to be configured. The Configuration Manager will allow entry and modification of CIMWeb users but it does not validate any changes to ensure proper compliance with standards. It is possible to make non-conforming changes with it. Because of all this, the accepted way to control CIMWeb access now is through the control panel exclusively. However, low-level CIMPortal Plus access control is still done through the configuration panel.

2.2.2.5 Deploying Packages

2.2.2.5.1 Objective

This section will enable you to deploy packages that have been created with Equipment Model Developer so they can be accessed by Interface A clients.

2.2.2.5.2 Prerequisites

- Create an Equipment Model. This equipment model must be stored in a pkg file accessible to the target machine.
- Install CIMPortal Plus on the target machine.
- **IMPORTANT:** If CIMPortal Plus is running on the same machine as EMDeveloper was run to create the deployment package, be sure to close any DCIMs that are their own processes to avoid RPC errors on deployment when CIMPortal Plus cannot attach to the DCIM.
- Start CIMPortal Plus on the target machine.
- The user privileges required to deploy a package are nearly entirely dependent on the DCIMs in the package. At the minimum, the user must be able to create files and directories in CIMPortal Plus's deployment packages directory. Normally, the ability to register COM objects is required, which requires administrative privileges.

2.2.2.5.3 Description

Once a package has been created with EM Developer, it must be deployed on an equipment so CIMPortal Plus can use it. The deployment package contains the equipment model and all the DCIMs needed to collect the data described in the equipment model.

2.2.2.5.4 Steps to complete:

1. Run the CIMPortal Plus Control Panel application.
2. On the "Configuration" tab, click the "Deploy" button at the bottom of the tab. This will bring up an "Open File" dialog box.
3. Using the "Open file" dialog box, browse to the pkg file containing the equipment file, select it, and click the "Open" button.
4. A dialog should pop up where the password used to create the package file should be entered.
5. CIMPortal Plus Service will extract the equipment model and DCIMs to the storage directory. It will also run the installation batch file for each DCIM package that is not currently installed. This will register any files needed by the DCIM package. Finally, CIMPortal Plus Service will start all DCIM instances used by the equipment. The newly deployed equipment should show up in the equipment list. By default, equipment models will come up in the "enabled" state. Once all the DCIMs have reported being ready, the equipment should change to the "ready" state.
6. NOTE: If deploying a very large equipment model, CIMPortal Plus could get a timeout error. It might be necessary to modify the CIMPortal Plus configuration file to adjust the maximum deployment time.
7. You may wish to change the equipment settings at this time.
8. To assign E132 privileges, use the Cimetrix Configuration Manager. By default, there are no privileges granted, which denies access to all Interface A clients. This application controls privileges for secured Cimetrix products. For testing purposes, select the "urn:semi-org:auth:anyPrincipal" user and click the "Properties" button. Click the "Add/Edit" button. Select the "CIMWeb1105.Equipment Name" module filter in the dropdown at the top. Select the "urn:semi-org:auth:allPrivileges" entry in the "Privilege" list box, and click the "OK" button. This creates an ACL Entry which allows anyone to connect to the equipment.
9. Open an Interface A client application (ECCE, EDAConnect Client or other) and verify that it can establish a connection to the equipment.

2.2.2.6 Redeploying Packages

This section will enable you to redeploy packages that have been modified so they can be accessed by clients. In particular, when using CIMPortal Plus Interface A it is important to follow this procedure to ensure that the "MetadataRevised" operation is sent to interested EDA clients.

2.2.2.6.1 Prerequisites

- Deploy the package on the target machine
- Modify the Equipment Model and create a new deployment package file. This modified equipment model must be stored in a pkg file accessible to the target machine.
- Do not delete the previous equipment deployment

2.2.2.6.2 Description

Once a package has been modified with EM Developer, it must be redeployed on equipment so CIMPortal Plus can use the modified package.

2.2.2.6.3 Steps to complete:

1. Run the CIMPortal Plus Control Panel application.
2. On the "Configuration" tab, click the "Deploy" button at the bottom of the tab. This will bring up an "Open File" dialog box.
3. Using the "Open file" dialog box, browse to the pkg file containing the modified equipment file, select it, and click the "Open" button.
4. A dialog should pop up where the password used to create the package file should be entered.
5. If the files have changed, CIMPortal Plus Control Panel will display a dialog to prompt if the existing files should be replaced. If yes, CIMPortal Plus Control Panel will prompt to shut down the current equipment. If yes, CIMPortal Plus will shut down the equipment. During shutdown, any connected Interface A clients will receive an E132 SessionFrozen callback.
6. You may be prompted to keep existing settings for deployed DCIMs.
7. After the equipment has been deployed and the equipment has been restarted, any connected Interface A clients will receive an E132 SessionPing callback. If the equipment metadata has changed, the Interface A clients will also receive an E125 MetadataRevised callback.

2.2.2.7 Deleting Equipment Deployment

2.2.2.7.1 Objective

This section describes how to delete equipment previously deployed in CIMPortal Plus Service.

CIMPortal Plus Interface A users should note that if equipment is deleted, then the Interface A clients will not receive an E125 MetadataRevised callback. Instead, deploy the equipment again without deleting it.

2.2.2.7.2 Prerequisites

- Deploy the package on the target machine

2.2.2.7.3 Description

In order to delete deployed equipment, please follow these steps:

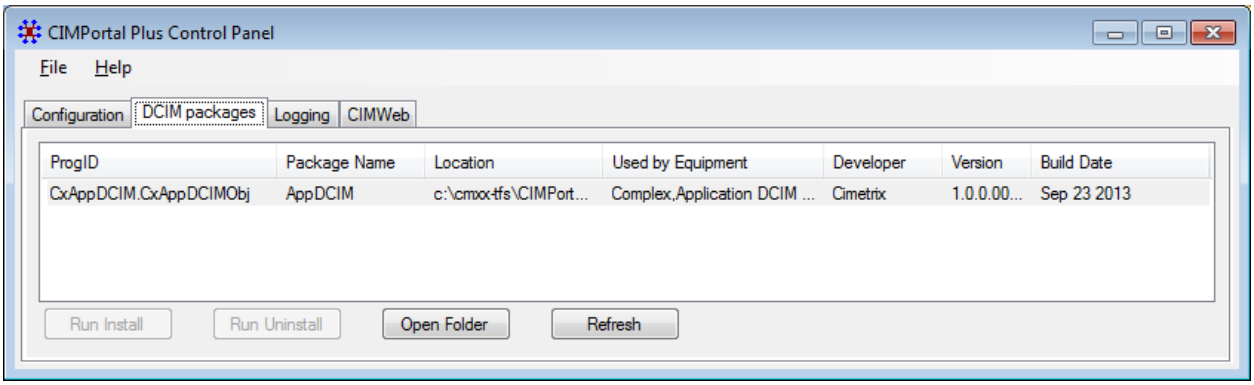
2.2.2.7.4 Steps to complete:

1. Run the CIMPortal Plus Control Panel application.
2. On the "Configuration" tab, select the equipment name in the list and then click the "Delete" button in the right panel.
3. A dialog box will appear and confirm whether or not to delete the equipment. Click on the "yes" button.

Another dialog box will appear. This determines whether or not to delete all of the files and directories associated with the equipment. Selecting "Yes" will remove all of these files and folders including data collection plans and persistent sessions. Selecting "No" will not delete any of the equipment files or folders such that if you deploy the same equipment again, the information will be persisted.

2.2.3 DCIM Packages

The "DCIM Packages" administration tab looks like this:



The DCIM package list shows the DCIM packages that have currently been deployed by CIMPortal Plus. The list shows the COM ProgID, package name, location, any equipment using the package, the developer, the version, and the build date.

At the bottom of the DCIM packages list there are several buttons.

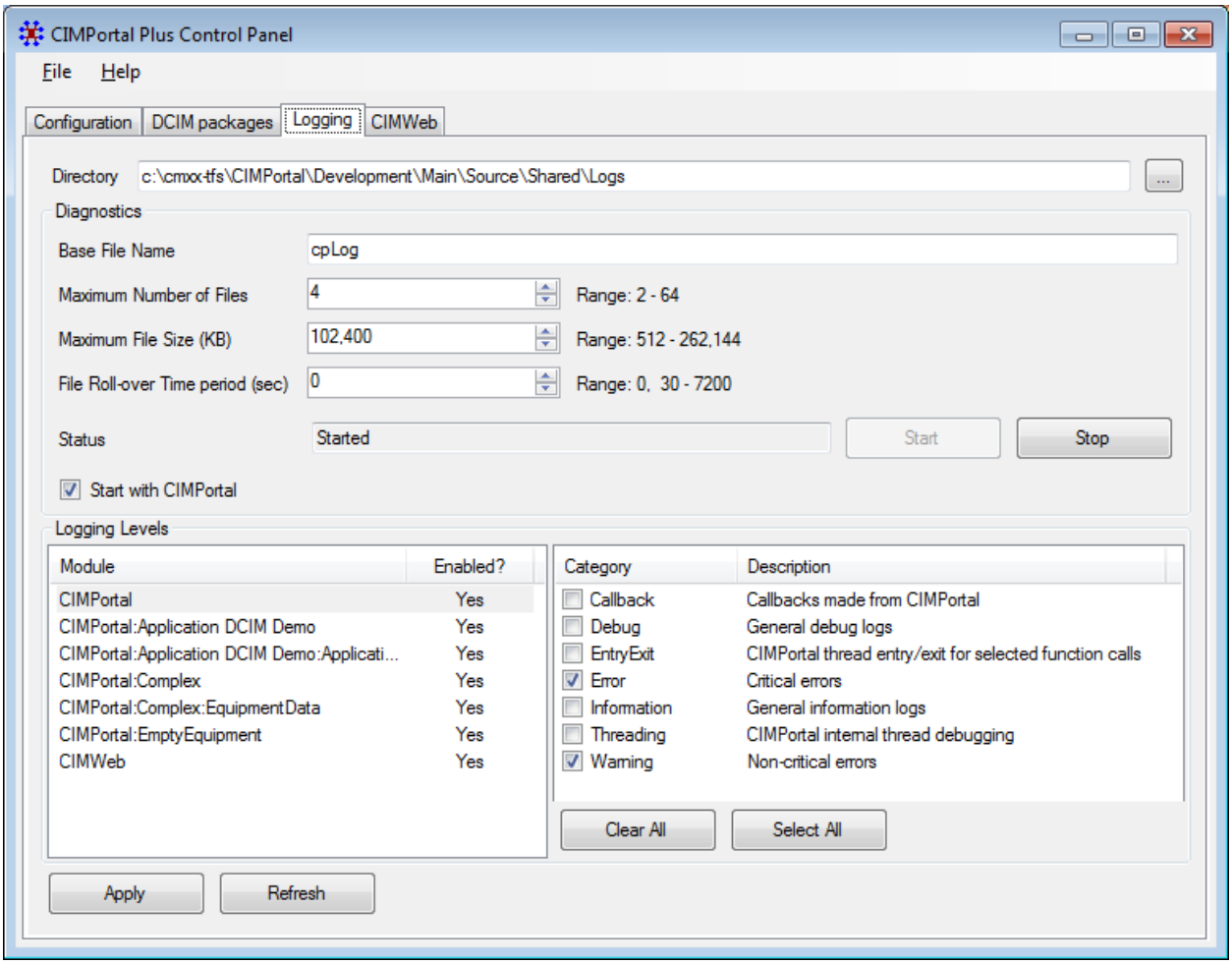
The "Run Install" button will run the DCIM package's install batch file. This is normally run during deployment and would typically not be done manually.

The "Run Uninstall" button will run the DCIM package's uninstall batch file. This is normally run during undeployment and would typically not be done manually.

The "Open Folder" button will open a Windows File Explorer in the package's location directory.
The "Refresh" button will refresh the information which appears in this tab.

2.2.4 Logging

Logging information can be extremely useful for debugging purposes. However, logging too much information can affect system performance. Also, information that is irrelevant to the problem being investigated makes relevant information less noticeable. These steps allow you to control what information CIMPortal Plus logs.



The logging tab allows configuration for diagnostic purposes. The top section controls overall behavior while the bottom section controls the level of detail for each equipment and DCIM.

The **Directory** is the location where the logging files will be placed. The path may be entered manually in the text box or the button to the right can be pressed to browse for the location. The default log file directory for CIMPortal is a Logs subfolder of the directory in which CxCIMPortalSvc.exe lives. Typically this is: c:\Program Files(x86)\Cimetrix\Comm Products\bin\Logs.

The **Base File Name** is the main part of the filename for log files. Additional information will be appended to this, along with a file extension, to create the actual name that will be used to create files in the Directory.

Maximum Number of Files indicates how many log files will exist in the Directory. Older files will be deleted to keep the number below this number. The minimum value is 2 and the maximum value is 64.

Maximum File Size indicates how large a file will be allowed to grow before a new file is created. This number is in kilobytes. The minimum value is 512 and the maximum value is 262144.

File Roll-over Time period indicates how frequently, in seconds, a new file will be created. A value of 0 indicates this field is ignored. The minimum value is 30 and the maximum value is 7200 seconds.

Status indicates the current status of logging: Started or Stopped. If the logger is stopped, the Start button will be enabled. If the logger is started, the Stop button will be enabled.

Start will cause logging to become active. Items will be logged to files in Directory until it is stopped.

Stop will cause logging to become inactive. Nothing will be logged until it is started again.

Start with CIMPortal indicates if logging will automatically start when the CIMPortal Plus service starts. If this is checked, it will. If it is not checked, it will not.

Logging Levels shows the various modules that can have logging options enabled on the left side and the specific options' status on the right side. Click on a line on the left side to show on the right the options available for that particular module. The *Enabled* column indicates if that module has any options enabled or not.

The right hand list will show all the options for the selected module with their description. A check indicates if that particular category is selected for logging or not. **Clear All** clears all the check boxes for the selected module. **Select All** checks all the boxes for the selected module. No changes to the status are actually made until the *Apply* button is pressed.

Apply will save all the changes since the last save or *Refresh*. This includes all changes on both the top and bottom sections of this tab except for the current status.

Refresh will reload the last saved values into the tab. Any changes that have not been applied will be lost.

2.2.4.1 Enabling Logging for Modules

- CIMPortal Plus Service -- select “CIMPortal” in the Module list and then selecting the desired logging Categories.
- Specific CIMPortal Plus equipment -- select that equipment (CIMPortal:<equipment name>) in the Module list and then selecting the desired logging Categories.
- Specific DCIM instance -- select the DCIM instance (CIMPortal:<equipment name>:<DCIM instance name>) in the Module list and then selecting the desired logging Categories.
- CIMStore Service -- select “CIMStore” in the Module list and then selecting the desired logging Categories.
- Specific CIMStore equipment -- select that equipment (CIMStore:<equipment name>) in the Module list and then selecting the desired logging Categories.
- CIMWeb logging -- select “CIMWeb” in the Module list and then selecting the desired logging Categories.

2.2.4.2 Logging Levels Defined

This section describes each of the available logging levels and when they should be used.

With the exception of Warnings and Errors, logging should not be turned on unless needed for a specific purpose. The Warnings and Errors levels of logging should be turned on at all times and will provide notice of any unusual behavior.

CIMPortal Categories

Category	Description	Use
Callback	Callbacks made from CIMPortal Plus	Logs registered callbacks.
Debug	General debug logs	Logs debug messages in the CIMPortal Plus source code.
Entry/Exit	CIMPortal Plus thread entry/exit for selected function calls	Logs when calls are made to and return from CIMPortal Plus internal functions. Very verbose (time intensive) and should only be turned on when requested by Cimetrix.
Error	Critical errors	Logs any errors caught by CIMPortal Plus. This is good to have on by default to know when errors are happening.

Information	General information logs	
Threading	CIMPortal Plus internal thread debugging	Logs when threads lock and unlock components to determine possible deadlocks in CIMPortal Plus. Should only be turned on when requested by Cimetrix.
Warnings	Non-critical errors	Log any warnings generated by CIMPortal Plus. This is good to have on by default to know when warnings are happening and may possibly indicate a potentially worsening problem.

CIMStore Categories

Category	Description	Use
Activity	Client Activity	Logs CIMStore database activity.
Admin	Administrative	Not used.
COMClient	COM Client	Not used.
COMClientCB	COM Client callbacks	Not used.
DataRequest	Data Request	Not used.
DBError	Database Errors	Logs errors generated as a result of failures communicating with the database.
Debug	Debugging	Logs debug messages. Fairly verbose.
Error	General Errors	Logs CIMStore errors.
Threading	Threading	Not used.
Warnings	General Warnings	Logs CIMStore warning messages.

CIMWeb Categories

Category	Description	Use
Debug	Debugging information	Logs expected behavior.
EntryExit	Entry/Exit	Logs when calls are made to and returned from internal functions. Should only be turned on when requested by Cimetrix.
Errors	Errors	Logs errors generated by CIMWeb.
Information	Information	Logs general operational messages.
Threading	Threading	Logs when locks are acquired and released. Very verbose. Should only be turned on when requested by Cimetrix.
Warnings	Warnings	Logs warnings generated by CIMWeb.

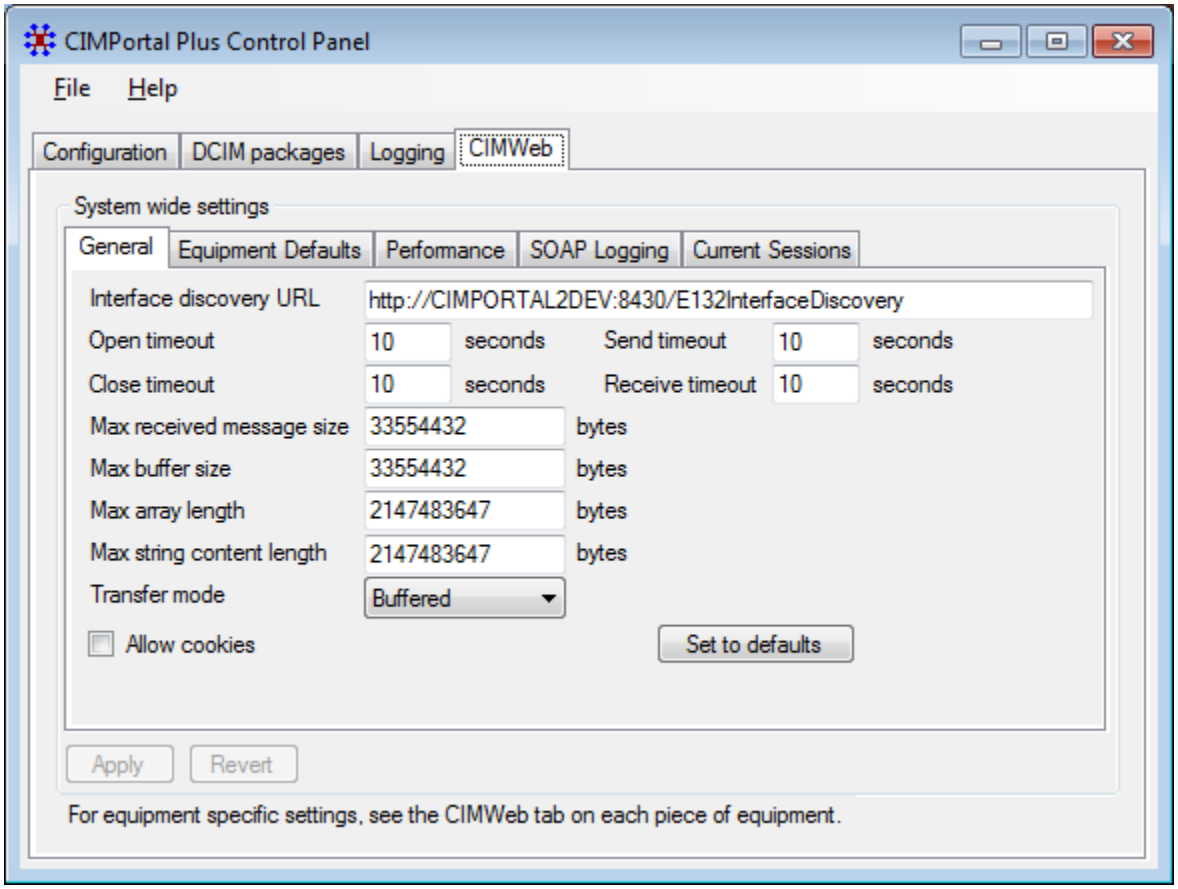
2.2.5 CIMWeb configuration

Equipment specific CIMWeb configuration is covered in sections 2.2.2.2, 2.2.2.3 and 2.2.2.4. This section covers overall configuration. It is divided up into general settings (section 2.2.5.1), equipment defaults

(section 2.2.5.2), performance administration (section 0), SOAP logging (section 2.2.5.4) and current sessions (section 2.2.5.5). Each one of these is on a separate sub-tab on the CIMWeb configuration tab.

2.2.5.1 CIMWeb General Settings

All but the first setting are used when establishing WCF connections and are set to maximum values. Changing these values may result in data loss for large models and DCP results.



The "Interface discovery URL" is the location for the E132 InterfaceDiscovery service. This is the URL used by clients to find out about other interfaces supported by this Interface A server.

The four timeout settings for Open, Close, Send and Receive are used by WCF when binding to HTTP connections. These are the timeout values it uses when establishing and closing sessions and sending and receiving data.

The four maximum length settings are used by WCF when setting up the transport and binding elements.

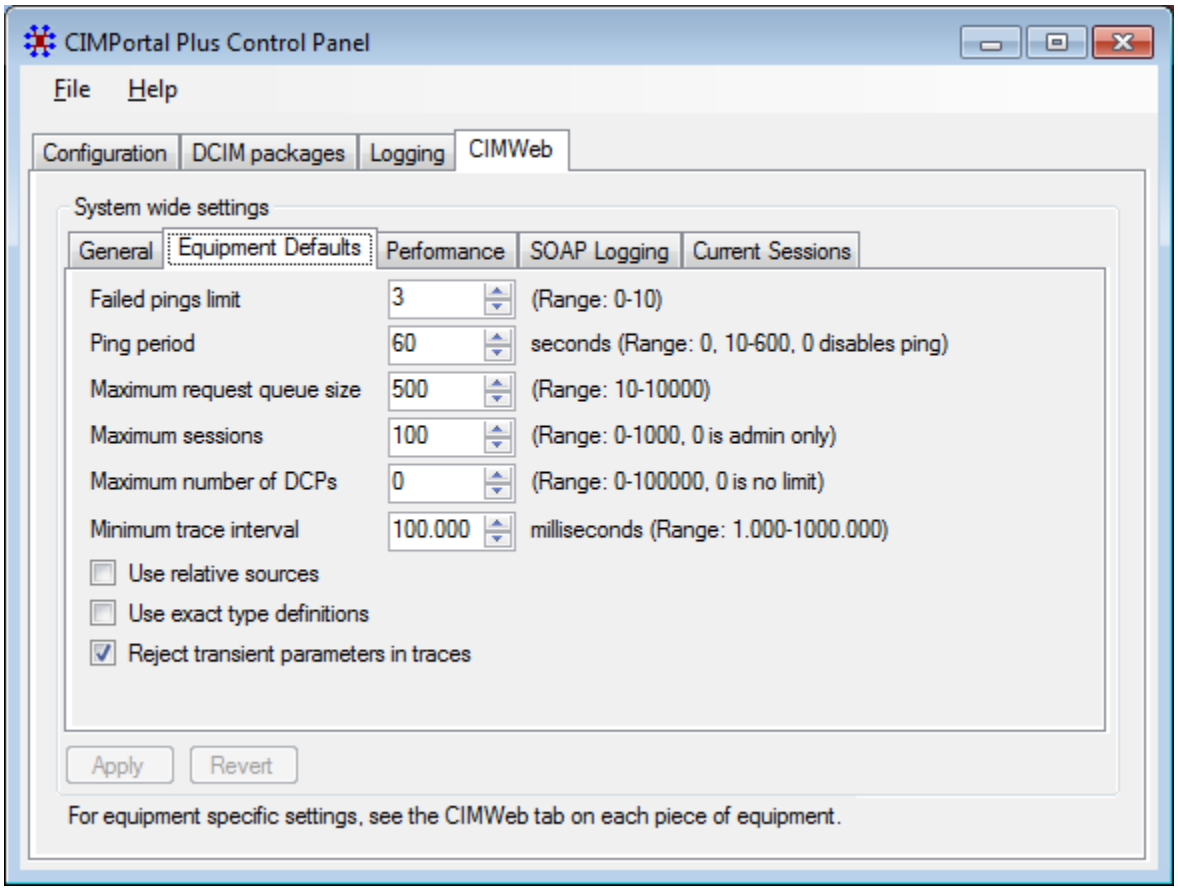
Transfer mode is not currently used. WCF is set to always use buffered mode.

"Allow cookies" indicates if WCF will allow the client to send cookies.

The Set to defaults button will set all values to the default values recommended by Cimetrix.

2.2.5.2 CIMWeb Equipment Defaults

The Equipment Defaults settings are not used directly by CIMWeb during normal operation. They are used to set the default values when a new piece of equipment is deployed. After deployment, the equipment has its own settings and any changes that are applied to it only impact it. Also, during deployment, the values from this tab are copied to the equipment. Any changes to these values will not impact already deployed equipment.



These fields all correspond directly to fields of the same name on the equipment/version specific general CIMWeb tab. See section 2.2.2.2 for more details.

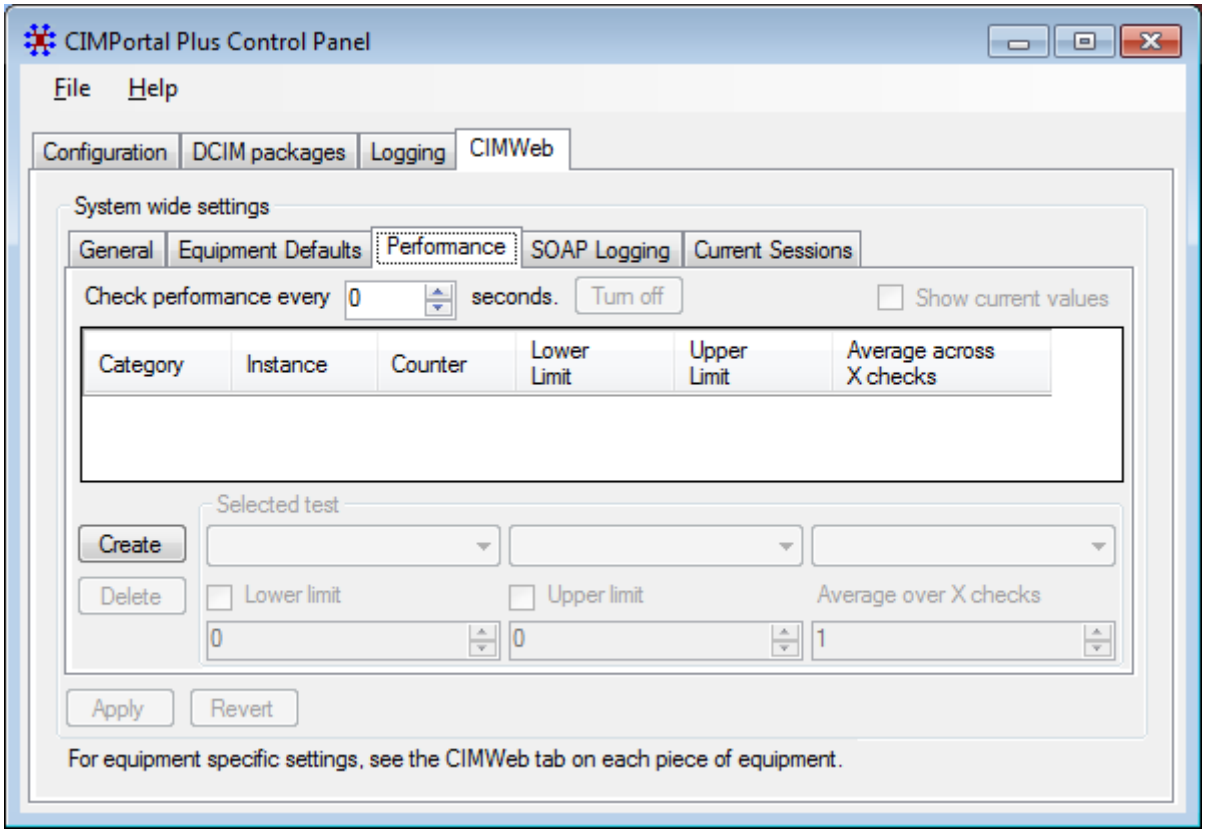
2.2.5.3 CIMWeb Performance Administration

The Performance tab is used to configure the required Performance checking that must be done by an Interface A server.

This section defines the meanings and options on the 'Performance' settings tab. For a detailed description of how CIMPortal Plus uses these values to send the PerformanceWarning and PerformanceRestored messages to the Interface A consumer, refer to CIMWeb Runtime Performance Monitoring (section 2.9.1).

The E134 standard requires that an Interface A server have the ability to continually monitor its performance impacts to a piece of equipment. The standard itself does not explicitly define the performance criteria to be monitored but instead specifies that the equipment manufacturer define the appropriate measurement criteria and response(s) to performance degradation.

The Performance tab is used to configure how CIMWeb limits itself to prevent performance degradation. CIMWeb uses a collection of tests to measure the performance of the computer. It runs all the tests at specified intervals. If a particular test fails, CIMWeb will deactivate data collection plans associated with that test. This page defines what these tests are, how often they are done, and the data collection plans associated with each test. These tests are all optional.



The 'Check performance every X seconds' field indicates the time, in seconds, between each time the tests are run. Must be greater than or equal to 1 to activate the “Turn on/off” button. Note that if there are no tests defined, then this value is not used.

The 'Show current values' button will retrieve and display the current value of the Windows Performance counter at the frequency set by the “Check performance every X seconds” field.

The list view shows all currently defined tests. Pressing the 'Create' button causes new tests to be added to the list view. Selecting a previously defined test will cause the test parameters to be displayed below the list view.

The three drop down combo boxes in the 'Selected test' group are used to define performance tests. These list boxes are populated with all of the measurable test options from the Windows Performance Monitor test application. In other words, the same tests that can be run from the Windows Performance monitor are available. These combo boxes represent test 'Category', test 'Instance', and test 'Counter' respectively. They correspond to the main control list boxes available in the Windows Performance Monitor (Perfmon.exe) application.

'Lower Limit' defines the minimum acceptable average for the test. An average below the 'Lower Limit' is a test failure. The 'Lower Limit' check box indicates if the limit is enabled or not.

'Upper Limit' defines the maximum acceptable average for the test. An average above the 'Upper Limit' is a test failure. An 'Upper Limit' checkbox indicates if the limit is enabled or not.

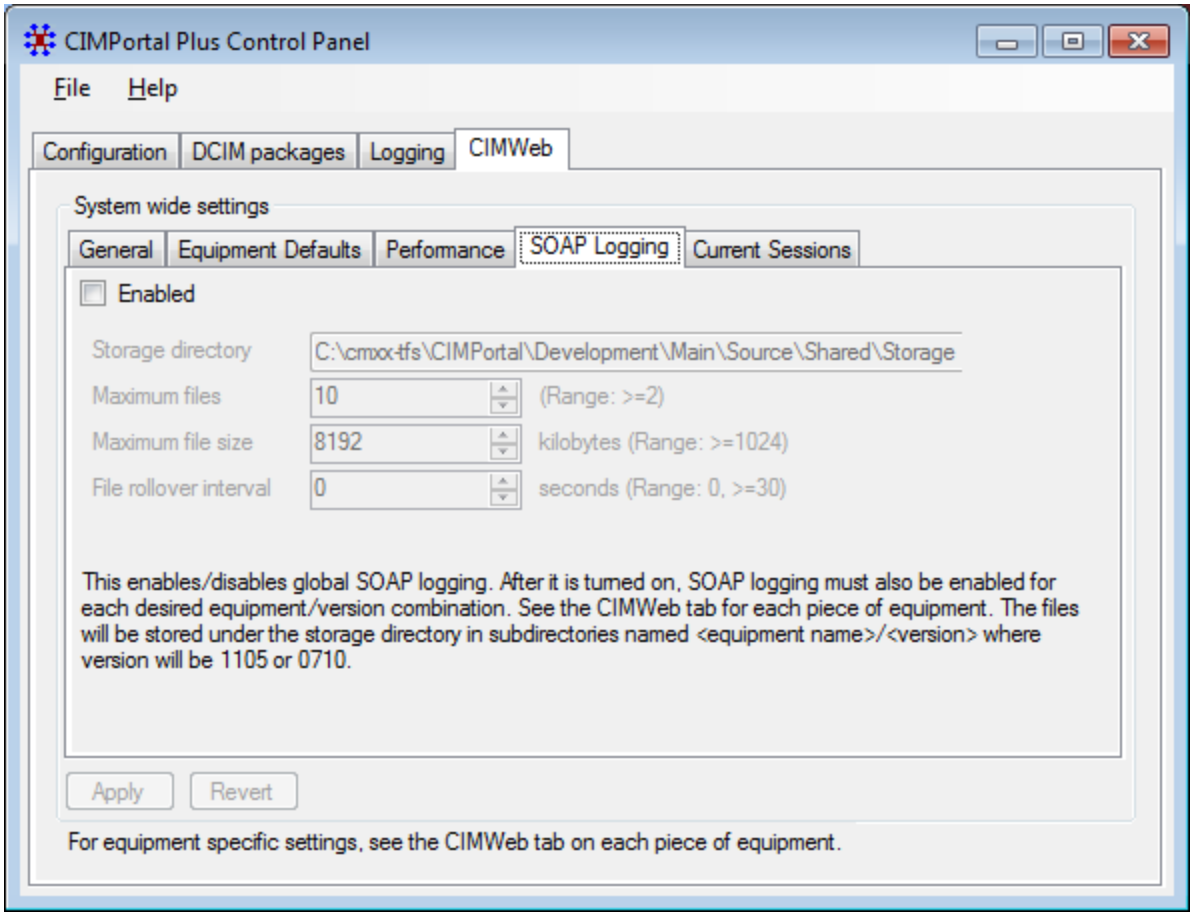
'Average over X checks' defines the number of test results to average together. Averaging a number of tests together can prevent a single spike from disabling data collection. This value must be greater or equal to 1. A value of 1 means no averaging will take place.

If a test fails, all active data collection plans are deactivated. If any one test fails, an E134 PerformanceWarningNotification will be sent. The E134 PerformanceRestoredNotification will be sent once all defined tests pass.

To delete a test, select the test in the list view and press the “Delete” button.

2.2.5.4 CIMWeb SOAP Logging

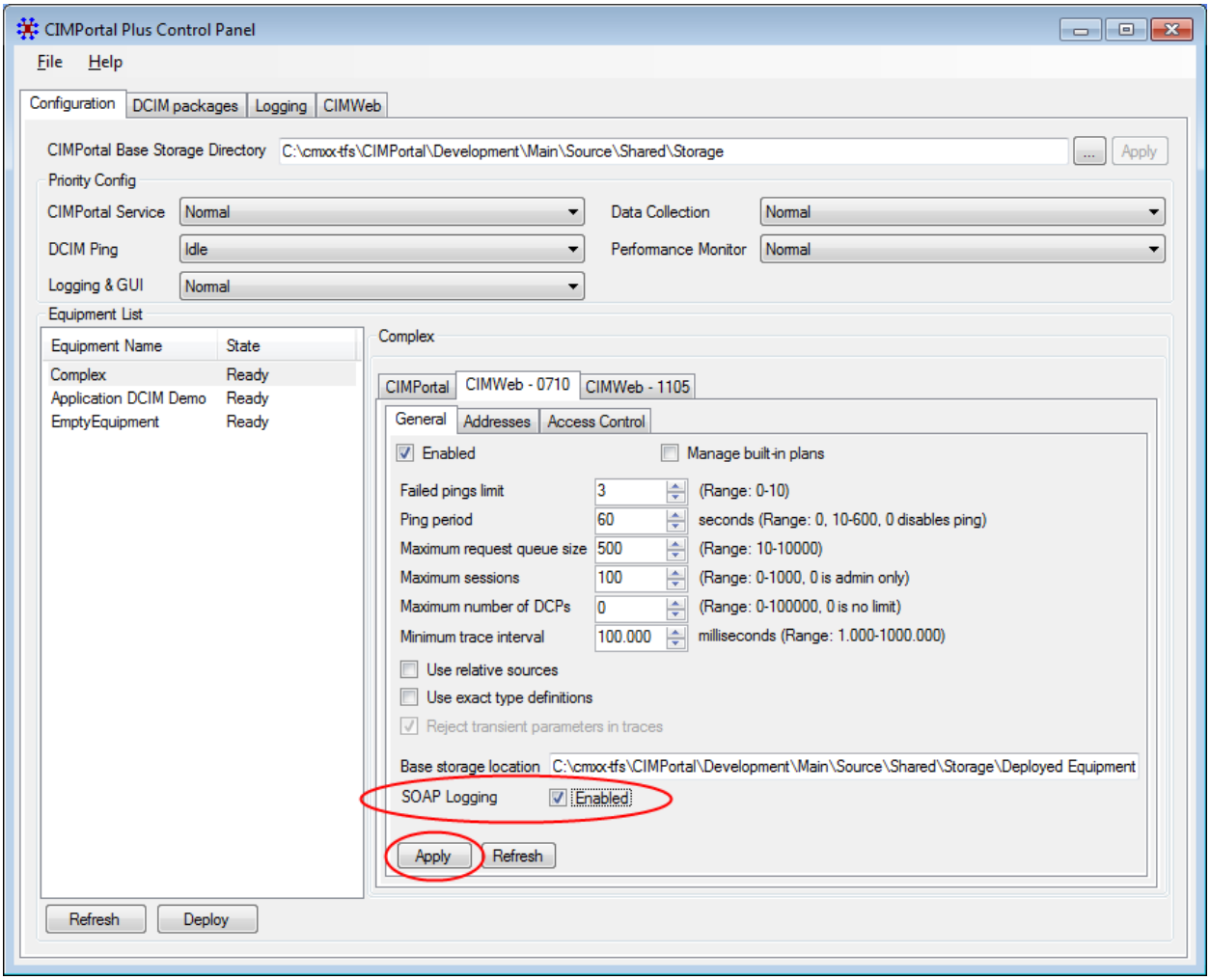
All the data transferred over the SOAP interface can be captured. It is separated into different files by equipment, interface version and client name. To enable this is a two-step process.



First, SOAP logging needs to be enabled globally. Do this on the CIMWeb / SOAP Logging tab. Click “Enabled” and then change any other parameters as needed. When done, click the “Apply” button.

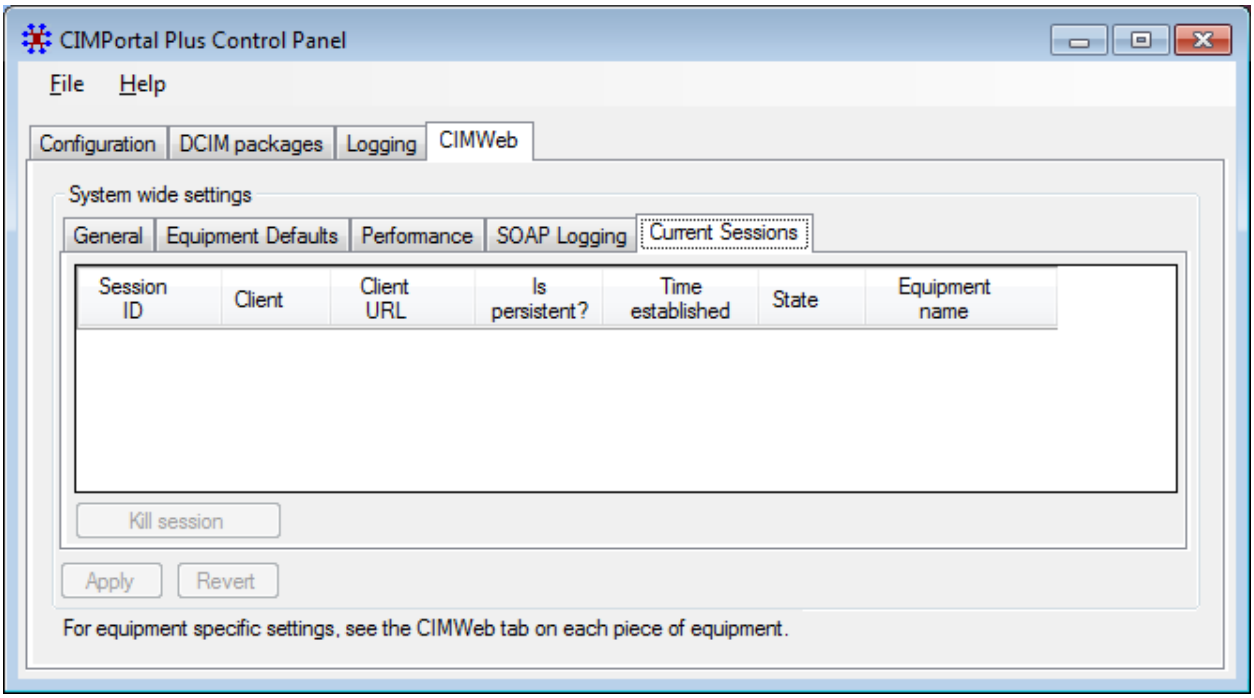
Since logging all this traffic may be a very expensive operation from both CPU and disk usage perspectives, this global enabled state is not saved when the CIMPortal Plus service is shutdown. The parameter values however are saved. When the CIMPortal Plus service is started, the global SOAP logging is always disabled.

Next, for each equipment/version combination where logging is desired, go to the appropriate CIMWeb tab for that combination and enable "SOAP logging." Click the "Apply" button.



Log files will now be generated for all SOAP traffic for that equipment and version combination. The files will show up in the directory indicated on the global tab in sub-directories for each equipment and version. Files will be created for each client name that connects and will rotate based on the parameters specified in the global SOAP Logging tab.

2.2.5.5 CIMWeb Current Sessions



The Current Sessions tab displays any currently connective active Interface A sessions. The list is automatically updated when a new session is established or an existing session is disconnected. Each column displays the information indicated in the header. Note that the Equipment Name includes the Interface A version information.

To manually kill a session, select the session in the list and press the "Kill session" button. This will stop any active DCPs for that session and disconnect it from the client.

2.3 Viewing log information

The log information can be viewed from a file for later analysis.

To view a log file, double-click a log file in the File Explorer. If the .n2l extension is not registered, the "Cimetrix LogViewer" program may be started from the Cimetrix Start menu or directly where it was installed.

Once the log viewer has been opened, there are many features which can be used to make analysis easier. These features are accessed by right-clicking on the log view window. One useful features are searching for text (Edit | Find).

2.4 Programmatically Enabling Logging

An application may enable and disable logging in CIMPortal Plus using API.

To configure file logging, use the ICxCPAdmin::SetFileLoggingConfig() API. Keep in mind the disk space needed is maxFiles * maxFileSize. If you do not use maxFileSize and use fileRollTime instead, then there is no formula for disk space usage. Use the filedir argument to specify the directory you want the log files to go into and filePrefix to specify the base name of the log file.

To enable and disable logging by particular modules, you will need to use the configuration service API. Instantiate CCxCMXConfig COM object to connect to the configuration service. Use the ICxCMXConfig::LogIn() API to log into the service with the normal "admin", "admin" or an account you have created. Then use the ICxCMXConfig::SetLoggingConfig() API to adjust an individual module's logging level. The module name is the same as it is in the CIMPortal Plus Control Panel logging tab. The level is an integer whose value is the sum of the desired LogCategory values. Error+Warning is a good starting level and

is recommended to leave at least this level on all the time. Threading will generate very noisy logs is not recommended unless suggested by Cimetrix Customer Support.

```
enum LogCategory
{
    None = 0,
    ChangeRecord = 4,
    Error = 8,
    Warning = 16,
    Debug = 32,
    Threading = 64,
    EntryExit = 128,
    Callback = 256,
    History = 512,
    Information = 32768,
    All = -1
};
```

An example:

```
using CxCIMPortalSvcLib;
using CxConfigSvcLib;
using CxNotify3;
using DCIMInterfaceLib;
.
.
.
             int level = (int)LogCategory.Error + (int)LogCategory.Warning;

             //change module logging settings
             var configsvc = new CxCMXConfig();
             configsvc.LogIn("admin", "admin");
             configsvc.SetLoggingConfig("CIMPortal", level);
             configsvc.SetLoggingConfig("CIMPortal:Equipment", level);
             configsvc.SetLoggingConfig("CIMWeb", level);

             var cpadmin = new CxCPAdmin();
             cpadmin.SetFileLoggingConfig("C:\\Logs", "Equipment", 4, 20480000, 0);
```

2.5 Equipment Model Developer Overview

The Cimetrix Equipment Model Developer (EMDeveloper) allows the user to create and edit an equipment model. The equipment model is composed of hierarchy elements and exposes data that can be queried by Interface A. The equipment model is used by CIMPortal Plus to create data collection plans (DCP) to report data through Interface A. Equipment Model Developer is part of the CIMPortal Plus SDK to allow model creation during development of the

equipment. It is used to create a deployment package for CIMPortal Plus to use at run-time. This tutorial describes how to create an equipment model for use with CIMPortal Plus.

2.5.1 EMDeveloper Tour

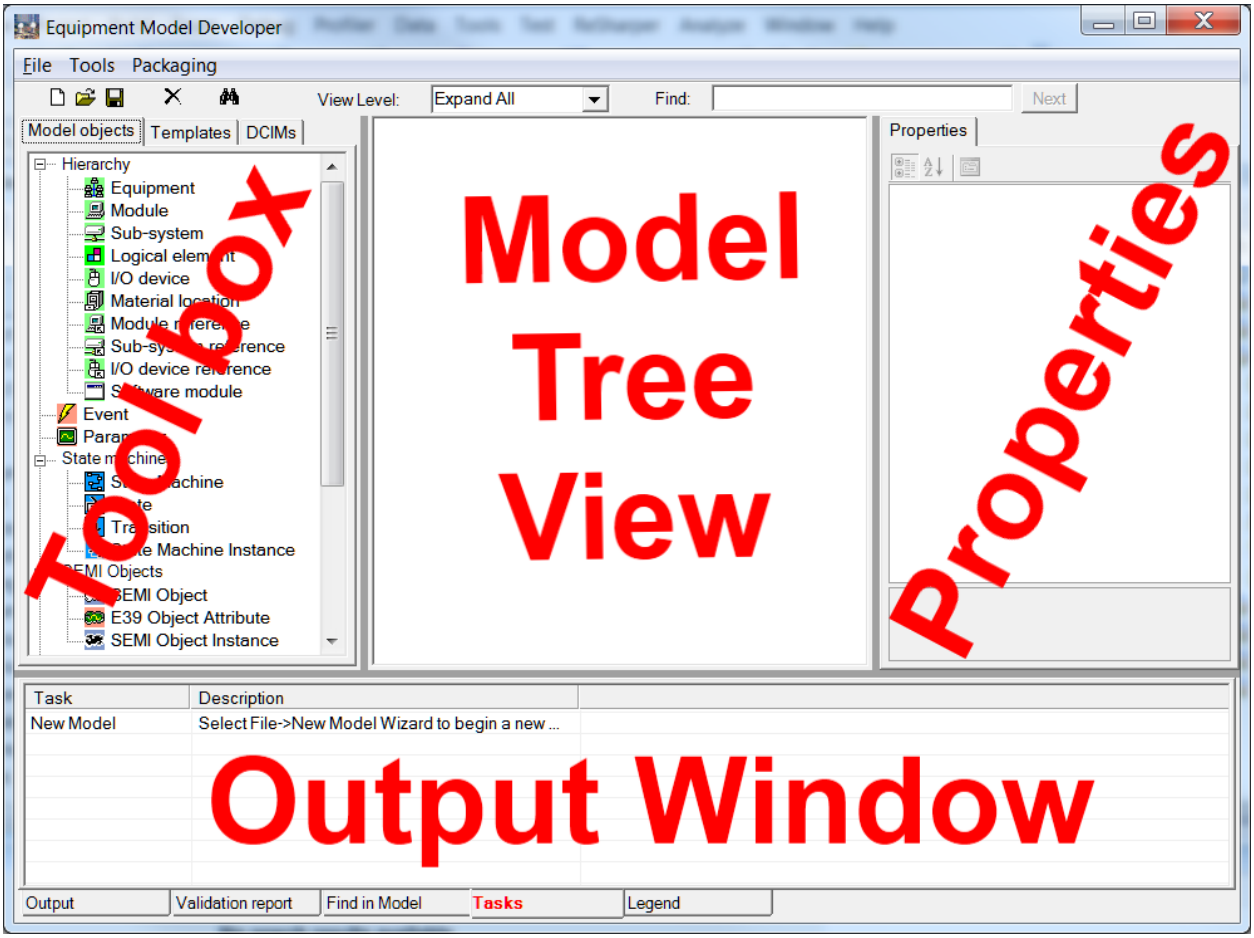
This section of the tutorial gives an overview of the EMDeveloper application and how to use the tools provided.

2.5.1.1 Launching EMDeveloper

The CIMPortal Plus installer adds EMDeveloper to the start menu. Select Programs | Cimetrix | CIMPortal Plus | Equipment Model Developer from the Start Menu to launch EMDeveloper.

2.5.1.1.1 EMDeveloper Main Panel

When EMDeveloper starts, the main application window appears as follows:



In addition to the menu and toolbar, the main window for EMDeveloper is divided into four sections: the Tool Box, Model Tree View, Properties and Output Window. Each of these panes can be resized by dragging the splitter bar between the panes.

2.5.1.2 EMDeveloper Menus

2.5.1.2.1 File Menu

The file menu has the following entries:

Action	Description
New Model	Create a new equipment model using a new model wizard. The new model wizard asks for equipment

Wizard...	information and creates the hierarchy structure for the equipment. It also adds several predefined state machines and SEMIObjects.
Open Model...	Load an existing equipment model using a file selection dialog. If a model file conforms to an older schema, EMDeveloper will ask if it should be converted. If the user answers in the affirmative, a conversion will be attempted and any errors output to the output tab. This conversion might take from minutes to hours, depending on the model. The file is not loaded if the user does not want it converted.
Clear Model	Clear the current equipment model
Save Model	Save the current equipment model to a file. If the current model has not been saved yet, this menu item behaves as Save Model As...
Save Model As...	Save an equipment model to a file provided by the user with a file selection dialog.
Exit	Exit EMDeveloper

The file menu also contains a most recently used files list that contains a configurable number of the most recently used model files. These files may be loaded by simply selecting the desired file from the list.

If a model has been previously saved, selecting Save will automatically update the existing file. Selecting "Save Model As…" will prompt for a file to save using the current filename as a suggested save file name. If a model has not been saved previously, a file selection dialog will appear to allow the user to select or enter the file to which the model will be saved. The name of the equipment node in the model will be provided as a suggested file name.

2.5.1.2.2 Tools Menu

The tools menu contains the following entries:

Action	Description
Find In Model	Find all occurrences in the model of nodes containing the specified name. See "Find in Model" in the Output Window section below.
Match Empty DCIM	Assign DCIM information to events, exceptions and parameters in the model that have not yet been assigned based on activated DCIMs.
Match DCP References	Match event and parameter references in the model to definitions of the events and parameters already in the model or create them if they do not exist
Resync DCIM/DCP references	Update any parameter, event and exception DCIM information in the corresponding parts of the model that have been changed. See sections 2.5.5.3.1 and 2.5.5.9 for additional details.
Fix model problems	Tool to help fix model problems frequently found during conversion from older models as well as new models. See section 2.5.4.4 Fix model problems tool for more information.
Generate Model UIDs	Generate new UIDs for every hierarchy node in the model.
Create Hardware XML...	Create a Hardware List from the current model.
DCIM Administrator...	Launch the DCIM Administrator Dialog. This allows the user to create new DCIM Instances to be used in model building. It also allows for editing existing DCIM Instances' settings. The DCIM Administrator dialog is discussed in section 2.5.2.
Options...	Display the Tools Options Page

2.5.1.2.3 Packaging Menu

The packaging menu allows the user to validate the model and, when valid, create a deployment package.

Action	Description
Final Validation	Make sure the model is valid and prepare for packaging.
Create Deployment Package...	Pack up the model package and its associated DCIMs into an encrypted package for deployment in CIMPortal Plus. This option is only available after the model has been validated. Please see section 2.5.6 for details.
Clear output before validation	If checked, the output window will be cleared before performing validation from this menu. Default behavior is to clear the output window when validation is started.

2.5.1.2.4 Toolbar

The toolbar allows the user to

- Create a new model
- Load an existing model from disk
- Save the current model to disk
- Clear the current model
- Find text in the current model
- Set the View Level - This expands and collapses the tree to the level specified in the combo box.

These options behave the same way the corresponding menu options from the File and Tool menus behave.

2.5.1.3 EMDeveloper Tool box

The Tool box is located on the left side of the EMDeveloper main window. It contains three tabs.

2.5.1.3.1 Model Objects

The first tab contains model objects. Items in this tab may be dragged to the desired location in the model. The cursor will change to indicate whether the selected node in the model can accept the type of node dragged from the toolbox.

Hierarchy items can be added to the model to provide structure (see Model Hierarchy). Equipment, Module, Subsystem, IO device, Logical element and Material location correspond with the E120 elements in the SEMI standards. Software Modules, Hierarchy references and Extension Models can be created from this toolbox.

The State Machines, SEMI Objects, Exceptions and Parameter types sections allow the user to create nodes related to these various parts of the model.

Please see section 2.5.4, Model Building, for more information.

2.5.1.3.2 Templates

The second tab contains a list of templates. These are common parts of models that are prebuilt for easily composing into a larger model. The templates installed by default are used by the "New Model Wizard" to construct an MCA compliant model. The user may create custom templates if so desired.

Please see section 0, Template Overview, for more information.

2.5.1.3.3 DCIMs

The third tab lists the DCIMs that have been configured in the DCIM administrator and are available for model building. A check mark next to a DCIM's name activates it and makes their DCP objects available for use when creating events, exceptions and parameters.

Please see section 2.5.3, Using DCIM Instances, for more information.

2.5.1.4 Model Tree View

The large main section of the EMDeveloper application window is a tree view of the current model. The model view control covers the upper center part of the EMDeveloper main window and contains a tree representation of the model. Each node in the tree represents an element in the model with the exception of Collection nodes. Collection nodes are used to clarify the model view by grouping like items. They aren't actual elements of the final model.

The model tree view is the target for Model Objects and Templates dragged from the toolbox.

The model tree has a context menu. The context menu is displayed by right-clicking on the tree view panel. The model context menu provides the ability to:

Action	Description
Copy	Copy the current node, its children and any dependencies to the clipboard for pasting in a different location in this model or other models.
Paste	Paste a node previously copied to the clipboard to the selected node. Dependent nodes are copied as needed. Collections are merged.
Delete	Delete the current node in the tree and all of its children.
Find In Model…	Find all instances of a string in the model tree. Results are displayed to the Find in Model tab of the output window.
Create Template…	Create a template file from the selected node and its children. Only available if a template can be created from the node.
Create Partial Template…	Create a partial template file from the selected node and its children. Only available if a template can be created from the node.
Generate UID	Generate a new UID for the selected hierarchy node. Only available for hierarchy nodes.
Go To Definition	If the selected node is a reference node, selecting this option finds the definition node in the tree, highlights it and makes it visible.
Show Instances	If the selected node is a definition of some type, this option will show all instances derived from this definition in the output window. Clicking on a blue link in the output will highlight the instance in the model tree.
Locate DCP object	If the selected node is a parameter or event, this option will show the DCP object in the DCIM instance.
Add Item	For hierarchy collections, this will add a new item of that type to the collection.

2.5.1.5 EMDeveloper Properties

The properties section of the EMDeveloper is located on the right of the main window. There may be multiple tabs shown in this section. The first tab always displays the properties for the model node currently selected. A property's value can be changed if it is enabled (not greyed out). Any change to a value in the properties box will be applied to the node in the model.

The remaining tabs, one for each activated DCIM, display the DCP objects contained in the DCIM. If a CIMConnect based DCIM does not display all the parameters that are expected, it may be because of an EPJ file issue. Please see Troubleshooting section 3.6.7 in the CIMConnect DCIM section of this guide.

2.5.1.6 EMDeveloper Output Window

The EMDeveloper output window is located in the lower part of the main window and contains five tabs.

2.5.1.6.1 Output

The output window displays log messages generated during the operation of EMDeveloper. A timestamp is prepended to each log message. Log message are displayed in one of three colors:

- black - Informational status messages
- red - Error messages

- gold - Warning messages

Some error messages have underlined, blue links. The user can click on a link and be taken to the node in the model that caused the entry.

The output window has a context menu that allows manipulation of the message log. The context menu is displayed by right-clicking in the output window. The menu provides the following options.

- Select All - Selects the entire log in the output window
- Copy - Copy the selected log items to the clipboard
- Save As - Saves the log to a text file.
- Find - Search the log for a specific string
- Clear - Clear the output window

Hot keys are available for most items in this menu.

There are also checked items that allow filtering the output of the window. Status, error and warning messages can each independently be turned on or off.

2.5.1.6.2 Validation Report

The Validation Report tab contains messages, comments, and errors related to selecting the Final Validation option from the Packaging menu. Messages are general status items, comments are things that don't necessarily need to be fixed but should be looked at, and errors are items that must be fixed in order to create a deployment package.

Across the top of the validation report tab are buttons to suppress comments. If a comment has been evaluated and deemed to not need any changed, it can be suppressed by selecting the comment and pressing the "Suppress comments" button. This comment will not be shown any more for this model in any future validations. Entire categories of comments may also be suppressed in the same way. To see comments that have been suppressed, click the "Show comments" button. A new node containing the suppressed comments will be shown alongside the Messages, Comments and Errors. Selecting sub-nodes in this section and pressing the "Restore comment" button will cause them to no longer be suppressed during validation. All these options are also available on a context menu.

The context menu also has an option to copy all the items in the validation report into the clipboard. It will put them in a comma delimited format for easier data manipulation.

2.5.1.6.3 Find in Model

The tree view context menu (displayed by right clicking on the model tree view) and the Tool Menu contain an entry called Find in Model.

The user enters the string to find dialog and the types of nodes to search. All instances of the string in node names and descriptions are listed in the Find in Model output window. Clicking any of the entries locates the node in the tree view and highlights it. Double-clicking changes focus to the model pane after locating and highlighting the node.

2.5.1.6.4 Tasks

The Tasks window contains a list of tasks and helpful hints for creating a model. Tasks may be automatically added and removed from this list to prompt the user on things that need to be done. Additionally, the user can add, edit and remove tasks manually using the context menu options.

Some tasks are associated with specific nodes. Double clicking on one of these tasks will select the related node in the model tree.

2.5.1.6.5 Legend

Except for Collection nodes, each node in the model tree displays an icon particular to that type of node. The Legend window displays all of the available icons and a description of what they mean.

Icons with green backgrounds are used for hierarchy (E120) elements. Icons with orange backgrounds are used for data collection elements (events, exceptions, parameters and references). Icons with blue backgrounds are used for E125 elements.

Double-clicking on any item will list all nodes of that type in the Find in Model tab.

2.5.1.7 Hardware List

With the exception of MaterialLocations, E120 model elements contain specific information for the hardware represented by the node. The hardware information contained in the model is:

- Supplier
- Make
- Model
- ModelRevision
- ImmutableID (Serial Number)

Software Module model elements contain specific information for the software represented by the node. The information contained in the model is:

- Supplier
- SoftwareVersion

The hardware list facilitates changing the above model properties without changing the model file itself. This allows a single model file to be used on different physical pieces of equipment where things like serial numbers and versions might be different. It allows minor changes to the hardware without having to redeploy a new model.

After the model is developed, the user creates this list by selecting the Tools | Create Hardware XML… menu option. The user is prompted to enter the name of the file in which the information will be stored. The name of the file used by CIMPortal Plus in updating the hardware for each model is called Hardware.xml. Once the hardware information is stored to a file, this file can be used as input for future updates in the field as described in section 2.8.1, Update Hardware.

See section 2.12, Metadata Update, for a tool used to make changes to the hardware.xml file.

2.5.1.8 Option configuration

This dialog allows the user to change a number of configuration values used by EMDeveloper. It is accessed via the Tools | Options menu item.

2.5.1.8.1 Base Directory

EMDeveloper uses several directories for storing information while running. These directories are configured for typical use with CIMPortal Plus during installation. All of these directories are subdirectories of a single base directory. This field makes it possible to change the base directory used by EMDeveloper for building models and creating deployment packages.

2.5.1.8.2 Max MRU Files

This setting specifies the maximum number of files that will be displayed in the most recently used files list under the File menu of EM Developer.

2.5.1.8.3 Backups

If set, this option will cause backups of the model to automatically be made each time the model is saved. The backups will be stored in the directory specified in the “Directory” text box and will be named with the name of the model file post-pended with the date and time of the save.

If a relative directory is entered, a directory will be created relative to each model file. If an absolute directory is entered, all the model file backups will be in this one directory. To recover an earlier version of a file, use Explorer to find the desired file in the backup directory and copy it to the desired location, renaming as needed.

2.5.1.8.4 Save Changes

Clicking the OK button will check to see if the given base directory exists. If the base directory or any of the subdirectories do not exist, EMDeveloper will create them automatically.

2.5.1.8.5 Restore Defaults

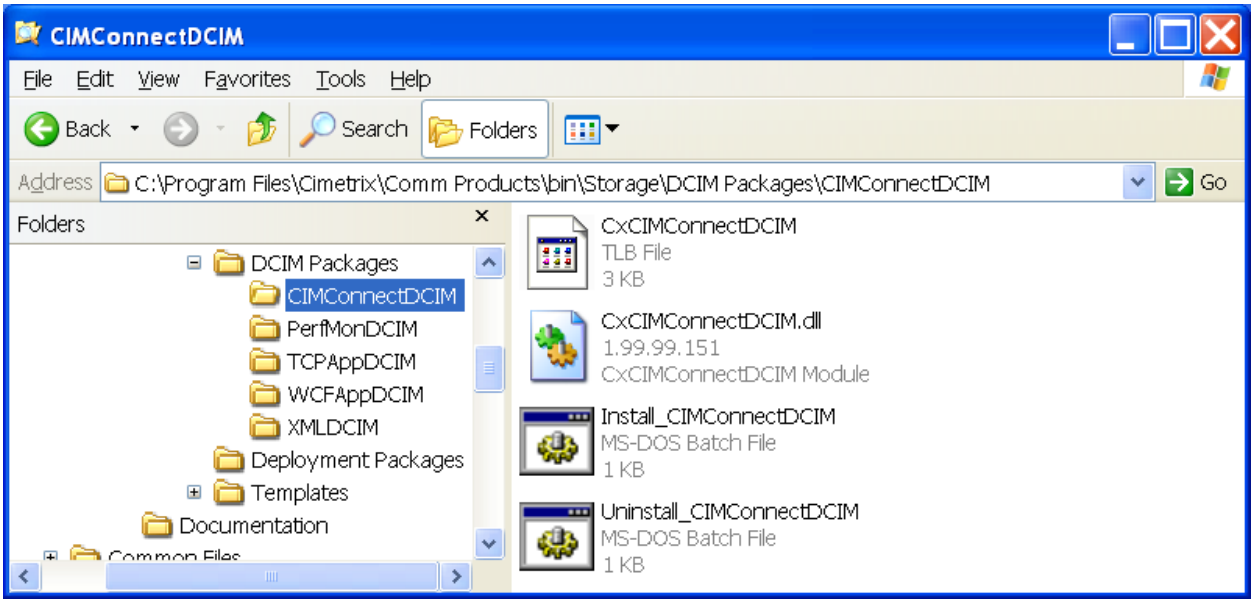
The Restore Defaults button will restore the factory defaults for the settings on this panel.

2.5.2 DCIM Administrator

The DCIM Administrator is accessed through the "Tools | DCIM Administrator…" menu item. It is used to create or update DCIM Instances to be used by EMDeveloper.

2.5.2.1 DCIM Package

A DCIM Package contains the files required to configure and execute a DCIM. These files include DLLs and/or executable files, install and uninstall batch files. All of the files that define a single DCIM are stored in a subdirectory of the basedir\DCIM Packages directory where basedir is defined by the configuration options. The name of the subdirectory is used as the name of the DCIM Package. Cimetrix provides several DCIM Packages with the standard install. The following directory structure shows the DCIM Packages provided by Cimetrix as well as the contents of the CIMConnectDCIM directory.



2.5.2.2 DCIM Instance

A DCIM Instance contains configuration information about a DCIM package. Like DCIM Packages, DCIM Instances are stored in subdirectories of the basedir\DCIM Instances directory. The name of the subdirectory is the DCIM Instance name. Each DCIM Instance subdirectory contains a DCIM.ini file and any configuration files required by the DCIM type for the DCIM Instance. For example, for an AppDCIM, an .adc file will be included in this directory. DCIM instances of other types might have different files in them. The DCIM.ini file contains information about the DCIM Instance. The following example displays the contents of a DCIM.ini file:

```
name=AppDCIMWaferProcessor
desc=Parameters, Events and Exceptions needed to complete wafer processing
runtimeinit=WaferProcessor1.adc,1
modelinit=WaferProcessor1.adc,1
dcimpkg=CxAppDCIM.CxAppDCIMObj
files=WaferProcessor1.adc
```

The DCIM Administrator makes it easy to create and edit DCIM Instances.

2.5.2.3 Creating a DCIM Instance

1. Enter a name and a description for the DCIM Instance.
2. Select a DCIM Package from the drop down list of DCIM packages.
3. Click the "Configure DCIM…" button to display the configuration dialog provided by the DCIM.
4. View and modify if needed configuration information in configuration dialog. The dialog will open at the last size set by the user. If this is not large enough to view everything, scroll bars will be shown to scroll hidden items into view. Or, the dialog can be resized to show all the information.
5. Click the Close button when done with the configuration. The "Model Config String" and "Runtime Init Strings" fields will be automatically filled in with information from the configuration dialog. Please note that the user can manually enter or modify these strings if desired.
6. Add additional needed configuration files by pressing the Add File(s) button. These files will be included in the deployed package when the DCIM instance is bundled for deployment.
7. Click the Save DCIM Instance button to save the DCIM Instance.

2.5.2.4 Loading a DCIM Instance

The DCIM Administrator also allows the user to load and modify existing DCIM Instances.

Press the Load DCIM Instance button to display a dialog that allows the user to select any DCIM Instances configured in the system.

The user selects the desired DCIM Instance and clicks the OK button.

The DCIM Instance Administrator fields are populated with the settings for the DCIM Instance.

Changes can be made to the fields.

When done, click the Save DCIM Instance button to update the files.

2.5.2.5 Closing the DCIM Administrator

When DCIM Administrator is closed, the DCIM tab on the tool box is updated with any new DCIM Instances that may have been added by the DCIM Administrator. If any DCIM Instances are open, they are also refreshed in case that DCIM changed during the DCIM Administrator session.

2.5.3 Using DCIM Instances

The DCIM tab in the tool box will display all the configured DCIMs. A computer used for model development may have multiple DCIMs available to develop models with, but any given model may only use a sub-set of those configured DCIMs. Therefore, information contained in DCIMs (DCP objects) is not available to be used until the DCIM is activated.

To activate one or more DCIMs, select the DCIM tab in the tool box and click the checkbox in front of the DCIM(s) that are to be used in the model. When this is done, the DCIM(s) will become available for use in the model by nodes that require DCIM information (e.g. events, parameters and exceptions) and a tab will be added to the Properties section for each activated DCIM.

To deactivate a DCIM, simply uncheck the DCIM on the DCIM tab in the tool box. If there are any model nodes referencing objects from that DCIM, a warning message will be displayed asking if the DCIM should actually be disabled. If there are no model nodes referring this DCIM, it will simply be deactivated. When deactivated, the DCIM's tab will be removed from the Properties section and any model nodes that reference the DCIM will have their DCIM information cleared. Model nodes with missing DCIM information will cause validation errors or comments (depending on the model node).

2.5.4 Model Building

2.5.4.1 Model Hierarchy

Hierarchy (or E120) nodes provide the framework to which all data that can be queried through a Data Collection Plan can be attached. These include Equipment, Modules, Subsystems, IO Devices, Logical Elements and Material Locations. A model must start with an Equipment node as its root. Other nodes may be added as children of this root.

The details about what a model should look like and how to model various features of a piece of equipment is beyond the scope of this document. In general, the nodes on the Model objects tab that should go in the model can be dragged onto the model tree view. There are rules in the standards that indicate which nodes can go where, so invalid combinations will not be allowed. Some general guidelines regarding model building are included below in the section called Modeling Best Practices.

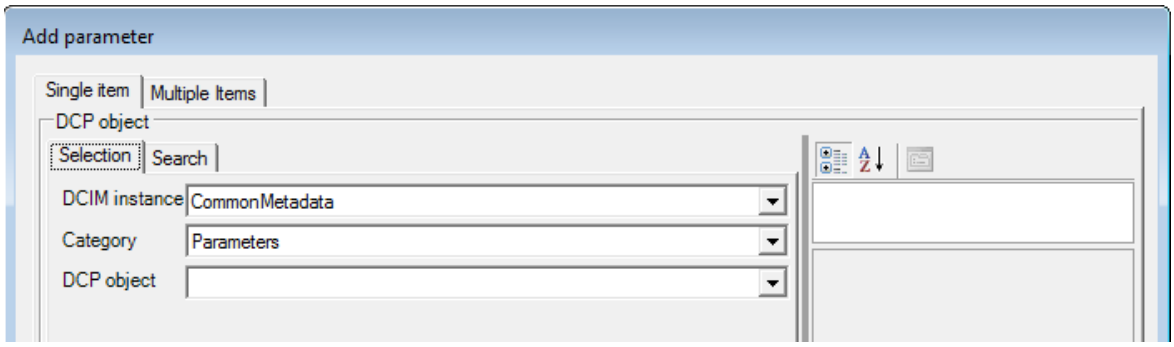
The model built in EMDeveloper is designed to work with either the 1105 or 0710 versions. It is independent of the standards version. At runtime, CIMWeb will convert this model to a version dependent model that is subsequently published to Interface A clients based on their connection point.

While, in general, nodes may be dragged and dropped on the model, there are a few nodes that will generate wizards or have special rules associated with them. This ensures models can be properly interpreted for the multiple versions supported. The following sections detail some of these cases.

2.5.4.1.1 Events, Parameters and Exception Instances

Events, parameters and exception instances link to an object in a DCIM, therefore at least one DCIM should be activated prior to using them. They can be created prior to activating a DCIM, but a reference to a DCIM object will have to be made as a second step prior to creating a package.

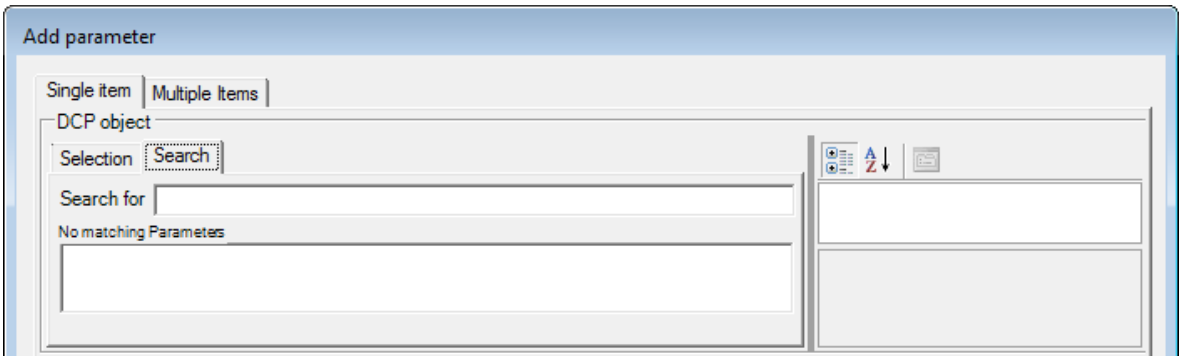
When any of these are added to a model, a dialog is shown with the following top section.



This allows the selection of a DCP object from the active DCIMs. The DCP object can be selected either through a direct selection process or by searching on the name.

To directly select the DCP object, use the Selection tab. The first combo box is used to select the DCIM; only the active DCIMs will be listed. If there is only one active DCIM, it will be automatically selected. When this is done, the Category combo box will be enabled and populated with the categories appropriate for the node type and defined by the DCIM. Like the DCIM instance, if there is only one item, it will be automatically selected. When the Category is selected, the DCP objects in that Category are populated in the third combo box. When the DCP object is selected it is displayed in the properties section on the right side of the dialog. This object will be associated with the event or parameter when the OK button is pressed.

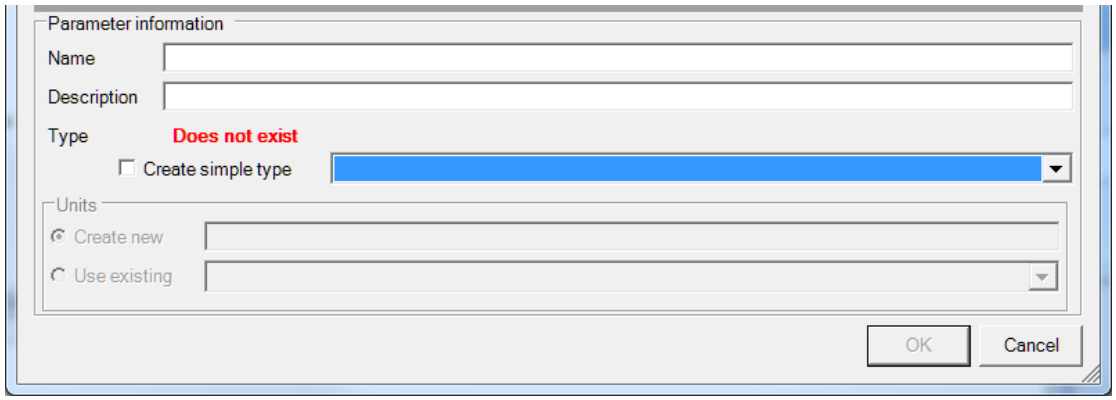
To search for a DCP object, select the Search tab.



In the "Search for" edit box, enter text to search for. When typing is finished, all DCP objects that have the entered text in either their name or description from any of the active DCIMs will be displayed in the box immediately below the entered text. (The box can be made larger by sliding the horizontal splitter down.) Selecting a line in the results will show that DCP object's properties in the property box on the right.

When creating any event, there are two additional fields to fill in: ID and description. For state machines' events, DCP object assignment is optional. If they are unassigned, the events will need to be specifically assigned when the state machine instance is created. If they are assigned, they will be used as the default values when state machine instances are created based on the state machine. For state machine instance's events, DCP object assignment is mandatory.

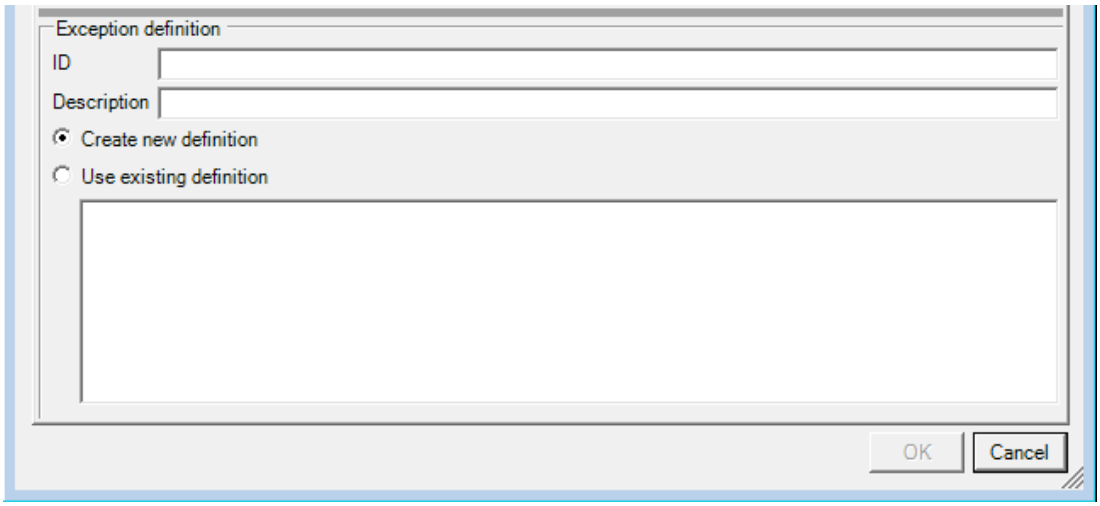
Parameters show the following part below the DCP object selection section.



The user can fill in the ID and description as appropriate. The data type, as defined by the DCP object, will show in the Type area. If it is not yet defined in the model, the red "Does not exist" text will show. This is to alert the user that the type will need to be created before final validation can be done. If the "Create simple type" box is checked, the wizard will create a new type with the name from the DCP object with the type selected in the combo box as the base type.

If the type for the parameter is of a type that requires a unit to be associated with it, then the "Units" section of the dialog will become enabled. The user must then either create a new unit in the model as named in the text box after the "Create new" option or select an existing unit from the combo box after the "Use existing" option.

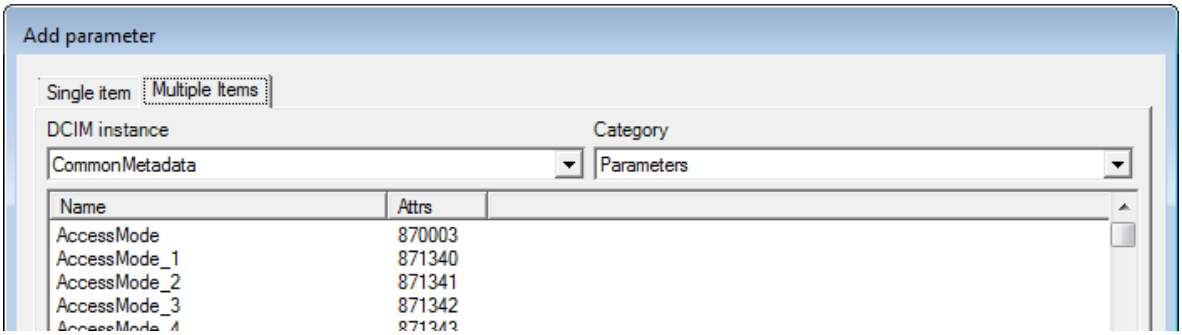
Exception instances show the following part below the DCP object selection section.



The user can fill in the ID and description as appropriate. Each instance must be associated with a definition. The user has the option to either create a new definition based on the existing instance or select an existing definition, if one exists.

Regardless of the object type, the OK button will become enabled when all the required fields are filled in.

This dialog also allows the user to select multiple DCP objects at the same time and adding them all to the model. The top tab can be changed to the multiple items view.



In this view, the user may select multiple items from the list by control clicking to select items one by one, or shift clicking to select everything from the currently selected item to the clicked on item. This is standard Windows multi-selection behavior.

The downside to this way of selecting DCP objects is any specific values that can be set on the single item view will receive default values using this method. If they need to be changed, then the changes will have to be done in the property view for each node once added to the model. For example, parameters' simple types will be created if needed but other types will not be. Units will be unassigned and generate a validation error until assigned. Exception definitions will be automatically created, if needed, based on the name of the DCP object.

Parameters have a parameter class property that can take on one of the values Configuration, Data or Control. If the GEM ID is set in the DCP object, these get mapped to a SECSVarClass in the EDA metadata as follows: if the parameter is transient, then it is a DV (data value). If the parameter is non-transient and a Data parameter class, then it becomes an SV (status value). If the parameter is non-transient and either a Configuration or Control class, then it becomes an ECV (equipment configuration value).

2.5.4.1.2 State machines

State machines are comprised of a State Machine node, at least one State node and at least one Transition node. State nodes can only be added to State Machine nodes. Transition nodes can only be added to State Machine nodes that already have at least one State node.

While event names cannot be duplicated within a hierarchy node (see section 2.5.5.2.3), they can be duplicated between transitions in a state machine. During final validation, state machines that have an assigned SEMI standard property may have duplicated event names and no comments will be generated. However, for state machines that are not indicated to be a SEMI standard, duplicate event names are flagged as a comment since many times when this happens outside a SEMI standard state machine, this is an unintentional modelling error. If these comments are reviewed and it is determined they are in fact intentional, the comment can be suppressed by using the normal suppress comment tool (see section 2.5.1.6.2).

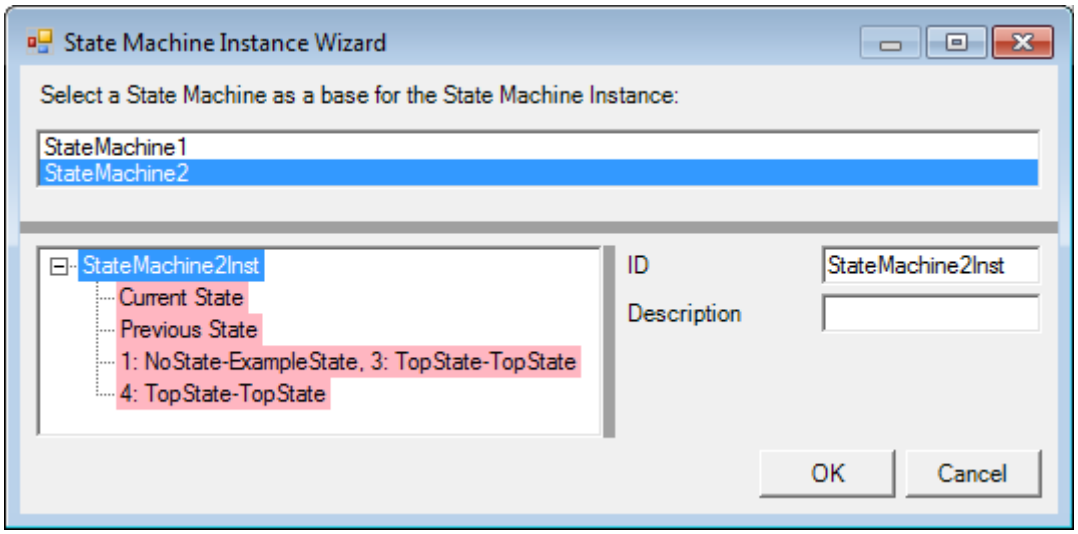
State machine instances can only be created based on an existing state machine and so no state machine instances can be created until the associated state machine is created. Deleting a state machine will delete any linked state machine instances.

2.5.4.1.3 State Machine Instances

Hierarchies of state machine instances must match the state machine hierarchy. For example, a state machine instance can only be an instance of a sub-state machine if it is also parented by a state machine instance that is associated with the sub-state machine's parent state machine.

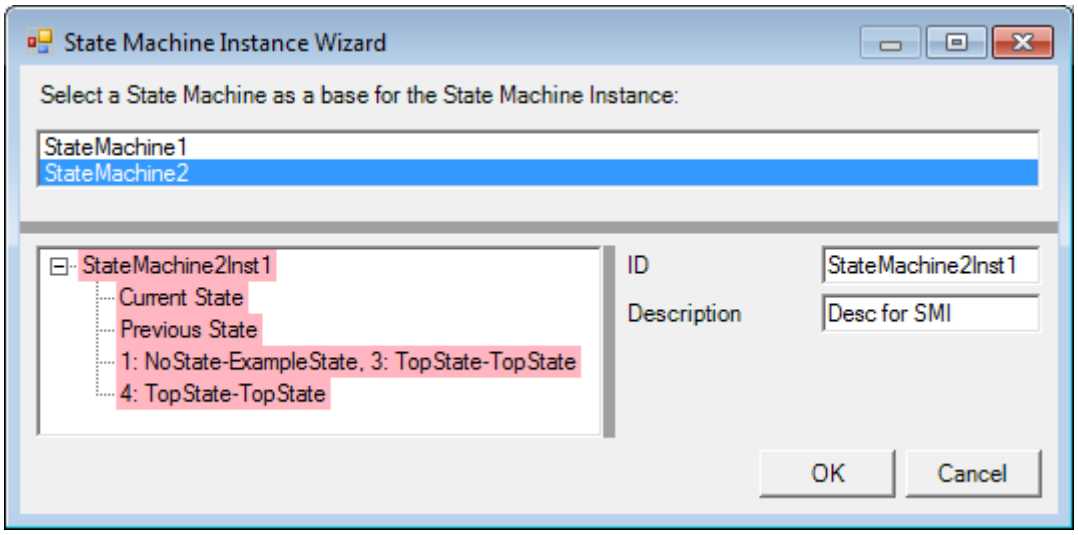
When a State Machine Instance node is dropped on the model, a wizard is shown. This wizard shows all the state machines defined in the model at the top of the dialog. Selecting one of these state machines will display the proposed state machine instance hierarchy in the lower part of the dialog.

A pink background for a node indicates a warning condition for that node or one of its sub-state machine instance nodes. Warnings are defined as an empty ID field for state machine instance nodes, an unassigned parameter for Current State and Previous State nodes and an unassigned DCP object for events. ID fields must be defined, so if there are any that are empty, then the OK button will be disabled. All the other items are optional for this wizard, but if left empty may cause warnings or errors for validation or undesired behavior during runtime.

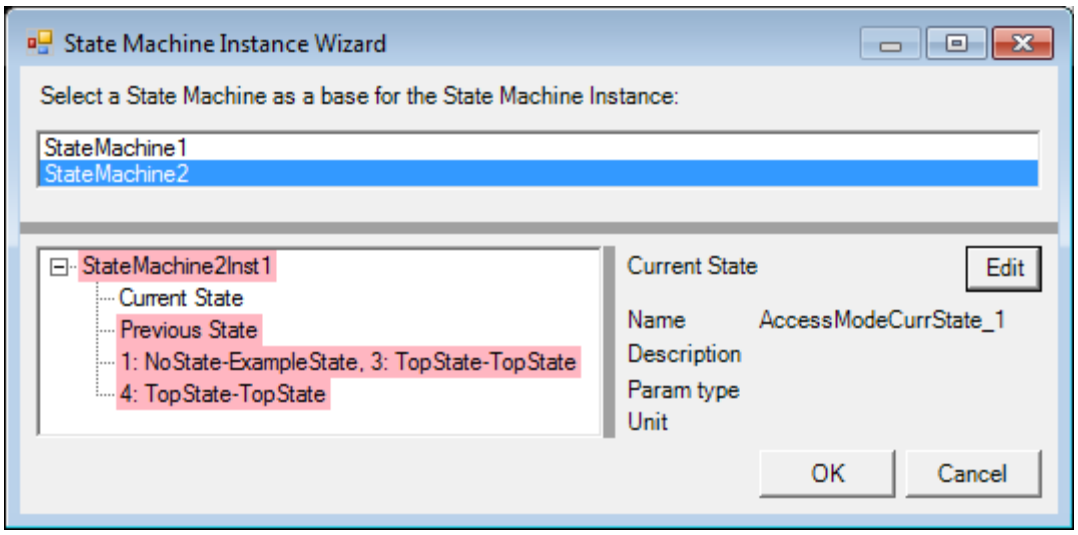


If the OK button is pressed at this time, the state machine instance will be created as shown.

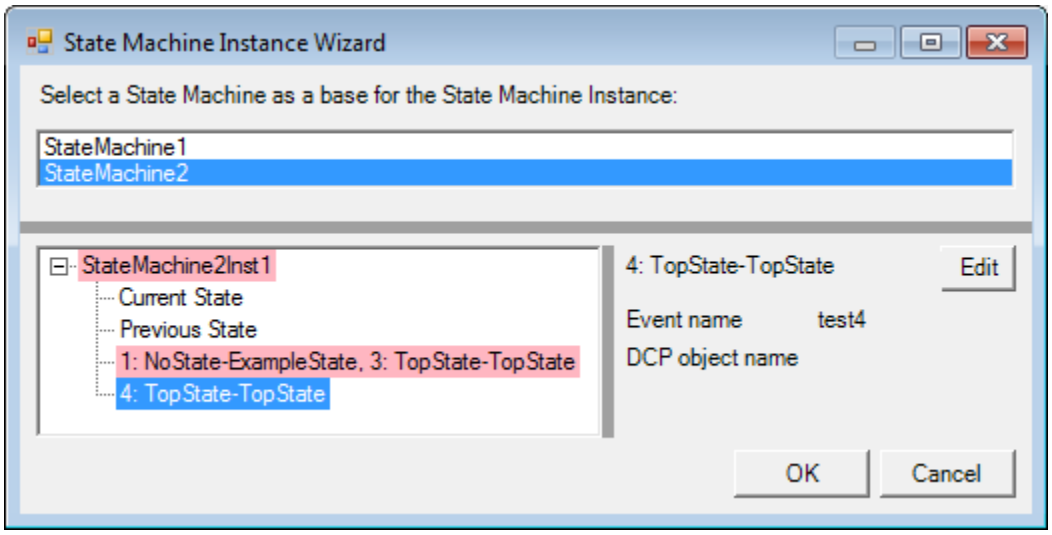
If desired, state machine instance IDs and Descriptions may be changed by selecting an instance node (e.g. StateMachine2Inst above). The ID and Description for that instance will show on the bottom right in edit boxes, allowing the user to change their values.



The user may optionally assign parameters to the Current State and Previous State. To do so, select the node to be assigned. A summary of the current values for that node will be displayed along with an Edit button. Pressing the button will bring up the "Add Parameter" wizard, allowing the user to specify a parameter for that state node. When that dialog is closed, the details for the selected parameter will be displayed.

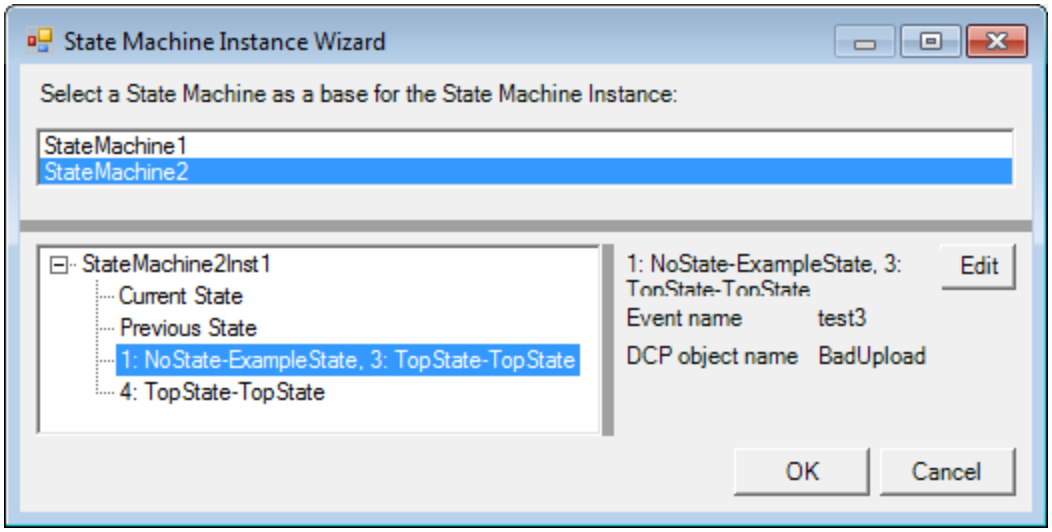


When a parameter is assigned to a state, that state will turn from pink, indicating a warning condition, to white. When both states for a state machine instance are assigned, and the instance's ID is not empty (the default condition), the state machine instance's background will also change from pink to white.



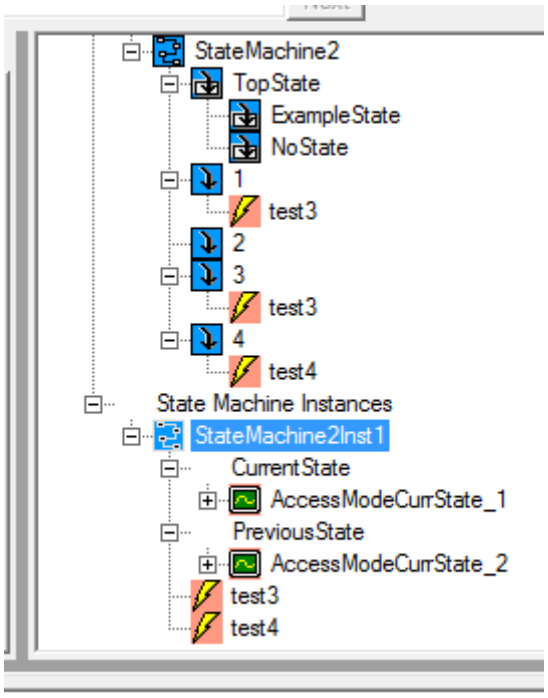
Each transition that has an event assigned to it in the state machine will be shown with their transition ID and description. If multiple transitions use the same event, they will be shown consolidated into a list as shown in the example above. When no DCP object is assigned to the event, the transition will be pink. The warning will be cleared when a DCP object is assigned for that transition's event. To assign a DCP object to an event, first select the transition in the left side. The event name will be displayed on the right. Select a DCP object for that event by clicking the "Edit" button on the right and using the DCP object selection dialog. If the state machine has a DCP object assigned to the event, the event in the state machine instance will be assigned that event as a default.

When all the states, events and IDs are assigned, there will be no warning conditions indicated.



When OK is pressed, the State Machine Instance will be created in the model.

When Cancel is pressed, nothing will be changed in the model.



2.5.4.1.4 SEMI objects

SEMI objects have a standards defined structure. When a SEM object node is created, several sub-nodes will also be added. These may be added to, changed or deleted as specific circumstances require. Any changes that are not standards compliant will trigger a final validation comment or error, depending on the nature of the violation.

Like state machines, SEMI object instances are based on a specific SEMI object and so instances cannot be added to the model until at least one SEMI object is created. Deleting a SEMI object will delete any SEMI object instances associated with it.

2.5.4.1.5 Exception Definitions and Exception Parameters

Exceptions changed significantly between the 1105 and 0710 standards. Exceptions are now similar to state machines and SEMI objects with two components: an exception definition and one or more exception instances. Each exception instance references exactly one exception definition but there may be multiple exception instances per definition. The exception definition must be associated with the root Equipment node; therefore each exception definition name is unique. The exception definition also specifies the severity.

Each exception definition has zero or more exception parameters. An exception parameter is not directly mapped to a DCIM's parameter. Instead, it indicates that the associated exception instances will have parameters and specifies where the definition will be located in the equipment model. In order to specify this, the exception parameter includes an attribute called the "Source". The Source attribute can be used two ways.

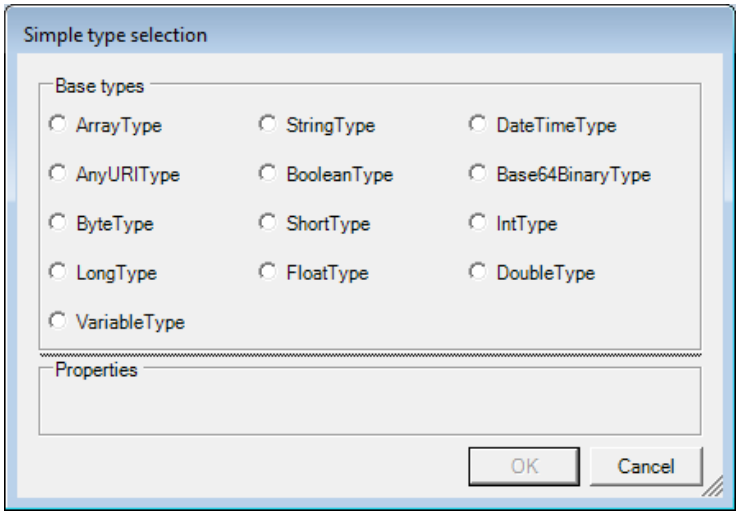
First, if the Source is an empty string, then a parameter with the same ID as the exception parameter must be defined in the same location as each exception instance. This allows each exception instance to map its exception parameters to unique parameters. In this case, the exception parameter has a relative mapping to a parameter, based on where the exception instance is defined.

Second, if the Source has a value, the value must be the location (using EDA Locator syntax) of the parameter with the same ID as the exception parameter. For example, the Source value might be "Equipment/MaterialManager" if the parameter is defined at that location. This allows all exception instances to map its exception parameter to an identical parameter. In this

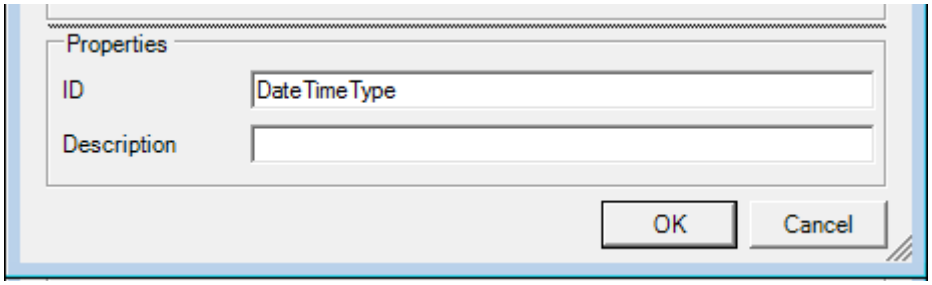
case, the exception parameter has an absolute mapping to a parameter, regardless of where the exception instance is defined.

2.5.4.1.6 Simple types

There are a small number of standards defined base types. Dropping the Simple Types model object node on the model will activate a small wizard showing all these types. This wizard will look something like this:



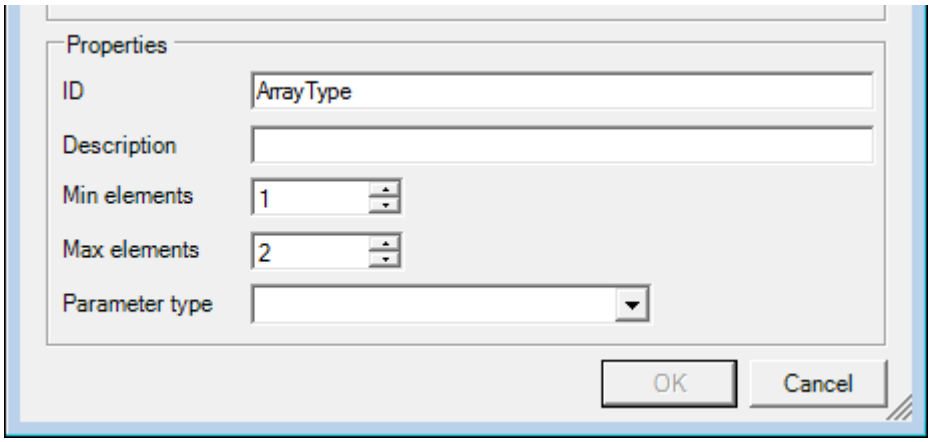
The section at the top are all the predefined types. When the user clicks on one, the properties section at the bottom is populated.



Most properties only have an ID and description. The ID is prepopulated with the name of the type selected. The Description is optional. Some types have additional properties that may or may not need to be filled in. The details for these are outlined in the sections below. When all requried fields are selected, the OK button becomes enabled. When the OK button is pressed, the new parameter type will be created. If Cancel is pressed at any time, the wizard will close and no changes will be made to the model.

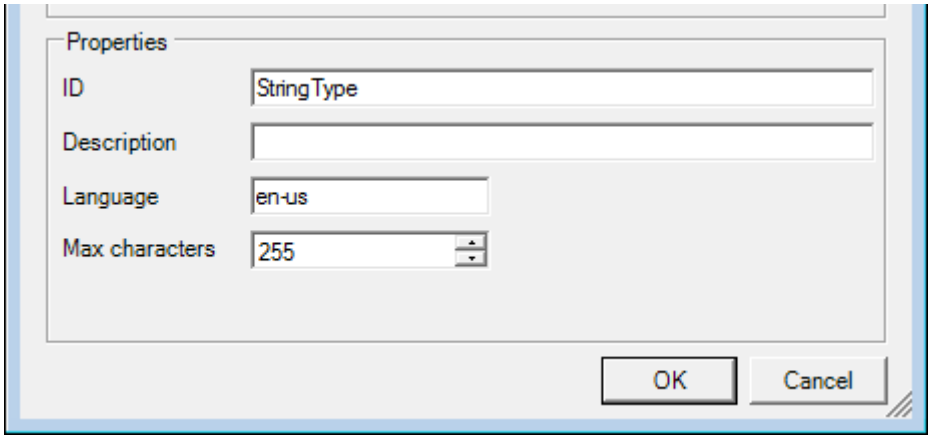
2.5.4.1.6.1 ArrayType properties

The array type has properites to specify the minmum and maximum number of elements as well as the type of the elements. All currently defined types will appear in the Parameter Type field. Note that the element's type contained needs to be created before the array so the type will appear in this property. The OK button will be enabled when the Max element count is greater or equal to the Min element count and a parameter type is selected.



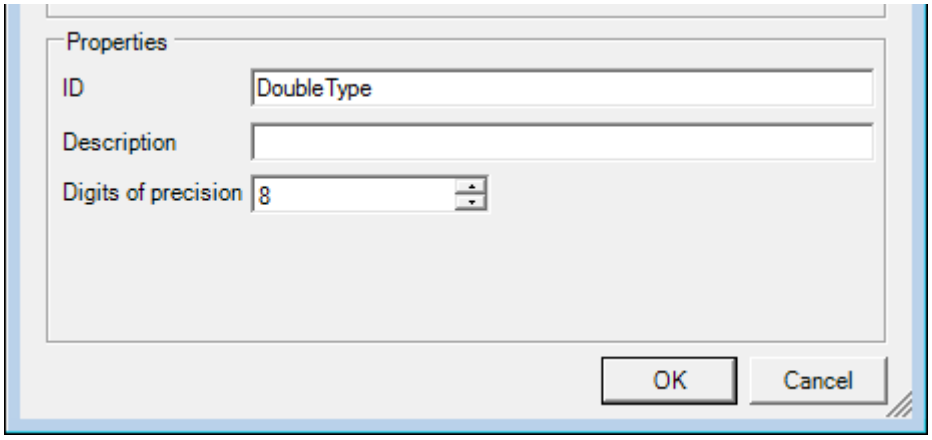
2.5.4.1.6.2 StringType properties

String types have a language and maximum number of characters they may contain. The language is a required field. If it is empty, the OK button will be disabled.



2.5.4.1.6.3 FloatType and DoubleType properties

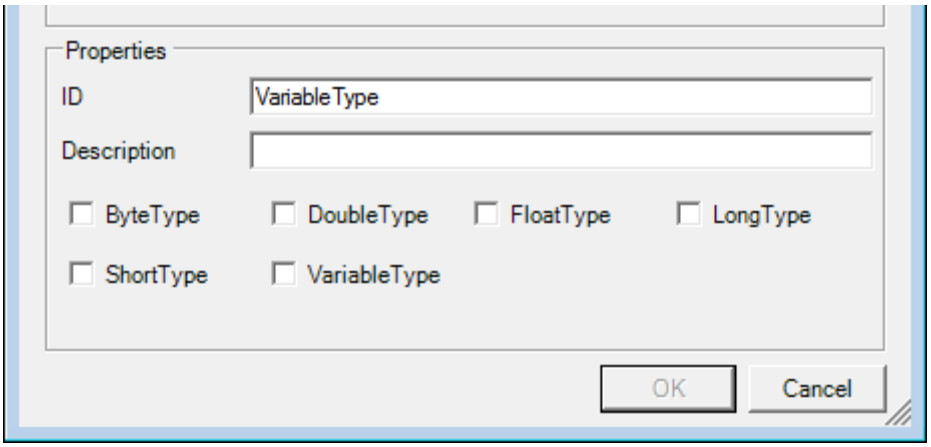
The FloatType and DoubleType types have a property to indicate the degree of precision stored in the type.



2.5.4.1.6.4 VariableType properties

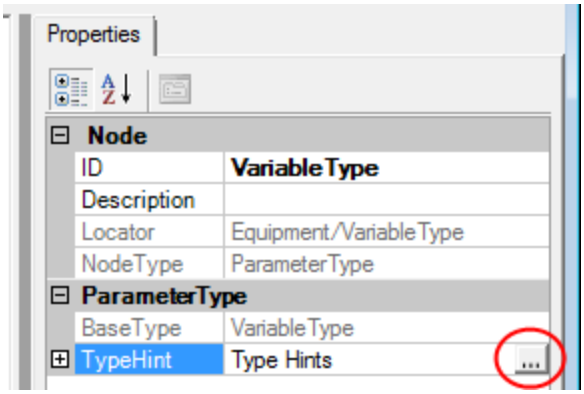
The VariableType contains a list of types that might be contained by the parameter that is of this type. It's sort of like a variant, in that it is a type that might contain any number of

different types in it. This is simply metadata that is communicated to the EDA client. These values do not impact CIMPortal's behavior.

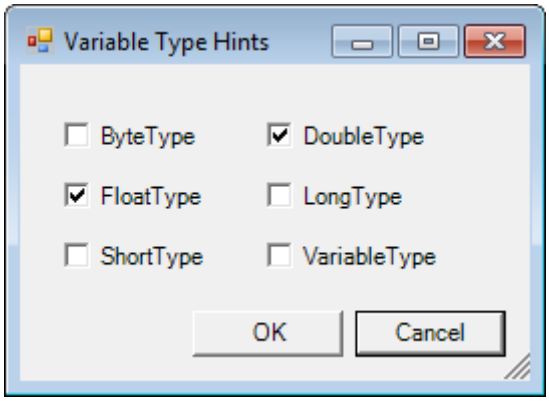


When this type is selected, all the currently defined types are shown as options to be included. The ones that are valid for this type should be checked. The OK button will be enabled when at least one type is checked.

Once the type is created, to edit the associated types, select the node in the model. The object's property grid (in the top right side of Equipment Model Developer) contains a field titled TypeHint. When this property has focus, an edit button is displayed.



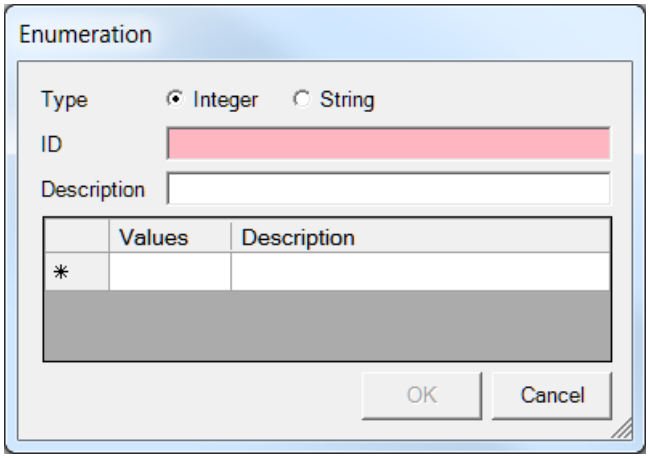
When this button is pressed, a dialog is shown similar to the wizard.



It displays all the currently defined parameter types and has checks by the ones that this variable type currently supports. Check the desired types to include them. Press OK when done and the model will be updated. Press Cancel to abandon any changes made without updating the model.

2.5.4.1.7 Enumerations

When the Enumeration node is dropped on the model view, the following dialog is displayed.



The type indicates if the enumerated values should be of type integer or string. If any value in the "Values" column of the grid does not match the type selected here, the OK button will not be enabled.

The user must enter a unique ID for the enumeration type and an optional description. At least one item in the "Values" column must be entered and be of the appropriate type.

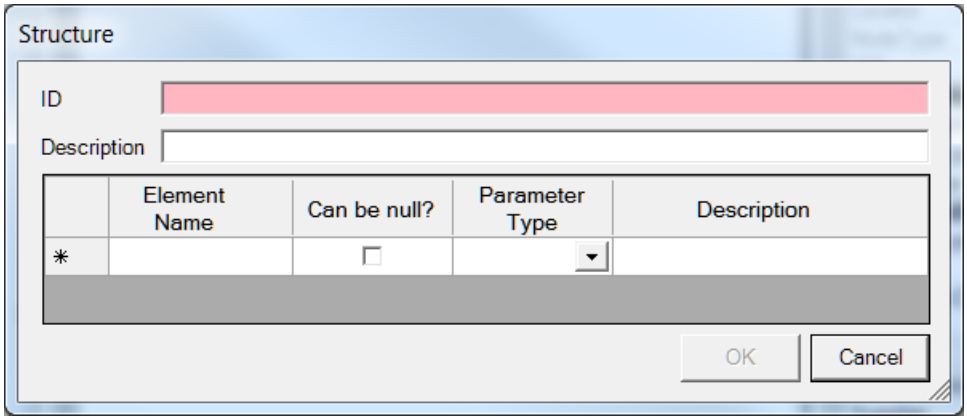
When OK is clicked, the enumeration with the defined values will be created.

To edit any of the values, simply click on the node to be changed in the model view and make the needed edit in the property window.

To add values to an existing enumeration, drop the Enumeration node from the model objects list to the existing enumeration node in the model view. The wizard will be displayed, prepopulated with the existing values. Any changes can be made to either existing values or new items added. When done, press the OK button and the model will be updated.

2.5.4.1.8 Structures

The following dialog is shown when the Structures node is dropped on the model view.



The user must enter the ID and at least one element to the structure before the OK button will be enabled.

Only parameter types that have already been created can be used, so make sure any types that are used by the structure are created before creating the structure.

Once a structure has been created, it may be edited by simply selecting one its constituent nodes in the model view and making changes in the property editor.

If element nodes need to be added to an existing structure, like the Element wizard, drop the Structure node from the model objects onto the appropriate existing structure in the model view. The structures wizard will be shown prepopulated with the data from the existing structure. Any changes made will be reflected in the model when the OK button is pressed.

2.5.4.2 New Model Wizard

The easiest way to get started creating a model hierarchy is to use the new model wizard. The new model wizard is launched from the File | New Model Wizard menu option or from the New Model button on the toolbar. The new model wizard assumes that the user has an intimate knowledge of the physical equipment to be modeled.

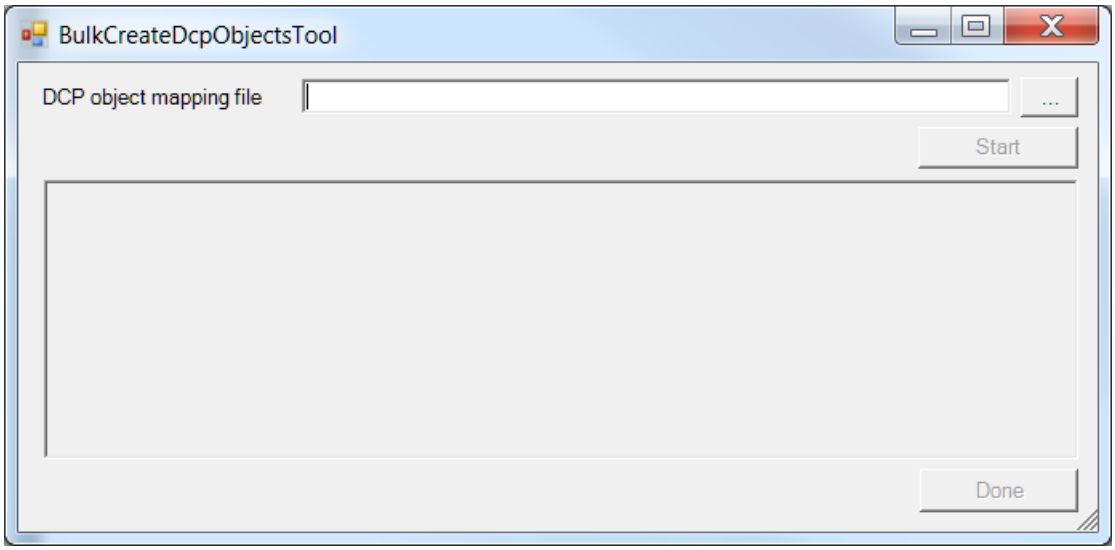
When launching the new model wizard, EMDeveloper first checks for unsaved changes to the current model and prompts to save changes if necessary. The new model wizard presents multiple tabs to be filled in by the user. The first tab contains the features that should be included in the model. The second tab prefills the parameterized values for the features and displays them to the user to verify and change if needed. The third tab allows the user to select which DCIMs to get DCP objects from for the events, exceptions and parameters in the template. The fourth tab allows the user to select where the resulting model should be saved. Finally, the fifth tab summarizes the previous tabs and displays any errors that need to be corrected. When all errors are fixed, the Finish button becomes enabled. The new model is generated and shown in the model pane when the Finish button is pressed.

After the model is created, the last step of the wizard will attempt to match parameters, events and exceptions in the model to DCP objects in the selected DCIM. If there are any DCP objects that are not found, the user will be prompted to save the DCP objects to an ADC file. (See section 2.5.5.3.3.) Some DCIMs have tools (notably the AppDCIM ADC Editor) that allow importing of these ADC files. (See section 2.6.8.) This gives the user a starting point to update their DCIM with information needed by E164 compliant models.

2.5.4.3 Bulk Create DCP objects tool

Customers may already have records outside of Cimetrix tools that associate DCP objects with model locations. To help provide an easier way than using drag-and-drop to recreate these structures, there is a “Bulk Create DCP objects” option on the Tools menu. This tool reads a text file that essentially contains mappings between DCP objects provided by DCIMs and the location in the model they should be added. The idea is customers who have this data in other data repositories can put the information in this file format and then import them into the model.

When this tool is selected on the menu, the user is presented a dialog box.



The location of the mapping file is entered in the "DCP object mapping file" field. A standard file selection dialog can be used using the "…" browse button on the right side. When a valid file is entered, it is preprocessed and, if it's a valid file format, the number of detected items to be added will be shown and the Start button will become active. If there are problems, they will be displayed in the large field in the bottom section of the dialog and the Start button will not become active.

When the Start button is pressed, the user will be given the option of saving the current state of the model and then, once saved, the tool will begin creating the objects. Any problems will be reported as they are encountered. Depending on the nature of the problem, some will generate a summary on the tool's dialog with details on the output tab of EMDeveloper. Others may have all their information on the tool's dialog. When all the items are processed, the Done button will become active. Once started, there is no way to cancel this process. Depending on the number of items to create, this may take a number of minutes. (For example, during testing 8,000 objects took approximately 20-30 minutes.)

2.5.4.3.1 Bulk Create DCP Objects file format

The file format for the BulkCreateDcpObjectTool is a name/value text format.

or / in first position of the line indicates a comment (line is ignored)

Otherwise the file is in the format:

SectionName

(whitespace)Name:Value

There may be one or more Name:Value lines per SectionName and any number of SectionName groups.

The SectionName can be one of the following:

- Is exactly "Settings": Indicates settings to controle the batch processing.
- Starts with "P": Parameter: File may have 0 or more of these sections. Each section defines a parameter to add to the model.
- Starts with "Ev": Event: File may have 0 or more of these sections. Each section defines an event to add to the model.
- Starts with "Ex": Exception: File may have 0 or more of these sections. Each section defines an exception to add to the model.
- Starts with "EP": Exception Parameter: File may have 0 or more of these sections. Each section defines an exception parameter to add to the model.

The remaining valid Name:Value lines are dependent on SectionName.

The following are valid values for the Settings SectionName:

- DcimName: This DCIM will be activated, if not already active. If it fails, no mappings will occur. This is optional and may occur multiple times.

The following items are common to all SectionName entries other than Settings. For the purposes below, the SectionName contains the DcpType.

- Locator: The locator of the model node the DcpType will be added to. This is required for all DcpTypes.
- DcpName: The name of the DCP object to add. Unless otherwise indicated, this is required for all DcpTypes.
- DcimName: The name of the DCIM containing the DCP object. This is optional. If not specified, all activated DCIMs will be searched and the first object matching the DcpName will be used.
- ModelName: The name of the new node. Unless otherwise indicated, this is optional. If not specified, the DcpName will be used.
- Description: The description of the new node. This is optional. If not specified, the DCP object's description will be used.

Parameter items

- ParamType: The value for the Parameter node's Parameter Type property. This is optional. If the type does not exist and it is a simple type, it will be created. Non-simple types must be created by hand.
- ParamClass: The value for the Parameter node's Parameter Class property. This is optional. If specified, it must be one of the values: Config, Control, Data.
- Unit: The value for the Parameter node's UnitRef property. This is optional. If the Unit does not exist, it will be created.
- TransientType: The value for the Parameter node's Transient Type property. This is optional. If specified, it must be one of the values: Transient, Restricted, Unrestricted.
- SmiState: If the Locator is a state machine instance node, this indicates if the parameter should be put in the current state or previous state property. Valid values are "CurrentState" and "PreviousState". Optionally, the value may be appended to the end of the Locator, e.g. Equipment/StateMachineInstance/CurrentState, instead of as a separate item.

Exception

- Severity: The value for the Exception Definition node's Severity property. This is optional. It is only used if the Exception Definition is automatically created. If specified, it must be one of the values: Undefined, Fatal, Error, Warning, Informational. If not specified and an exception definition is created, the severity from the DCP object will be used.

EParameter (i.e. Exception parameters)

- ExceptionName: The name of the exception definition. This is a required field. Locator field indicates the parent of this exception definition.
- DcpName: For this type, this is not used.
- ModelName: Name of the parameter to add to the exception. For this type, this is NOT optional!
- Source: The value for the Exception Parameter node's Source property. This is optional.

Event

- EventName: The value for the Event node's EventName property. This is optional.

2.5.4.3.2 Bulk Create DCP Object example file

```
Settings
 DcimName: CommonMetadata

Event
 Locator: Equipment/StateMachine/0
 DcpName: CarrierClosed
 ModelName: CC1event
 Description: This is a test event description from the import
 EventName: Ev-01

Event
 Locator: Equipment
 DcpName: CarrierClamped
 ModelName: CC2event
 Description: This is another test event description from the import
 EventName: Ev-02


Exception
 Locator:Equipment
 DcpName:AccessModeViolation
 ModelName:ExName1
 Severity:Fatal
 Description:Description for ExName1


EParameter
 Locator:Equipment
 ExceptionName:ExName1
 ModelName:ExParam1
 Source:testSource


Parameter
 Locator: Equipment
 DcpName: AccessMode
 ModelName: AMParam
 Description: This is a test parameter description from the import
 TransientType: Unrestricted
 ParamType:IntType
 ParamClass:Config
 Unit:testUnit1
```

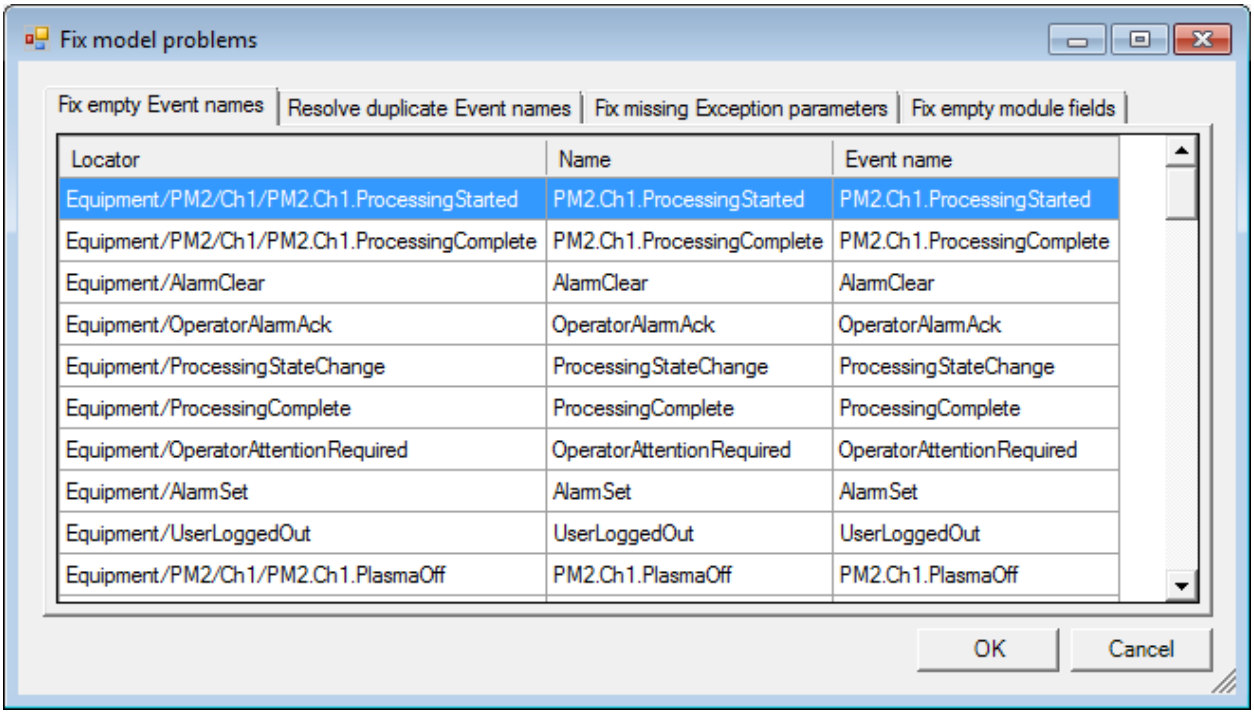
2.5.4.4 Fix model problems tool

Models may have a variety of different common problems. To help fix some of these, the Fix model problems tool exists. Each tab provides help on a specific type of problem. In general, if no problem exists for that type, the tab will be empty with a message indicating there were no problems found of that type.

To use this tool, go to each tab and, if desired, change the values on the tab as appropriate. When all the work that the user wishes to do is done, press the OK button and any changes on any of the tabs will be reflected in the model. To cancel all the work on all the tabs, press the Cancel button. The model will be left unchanged.

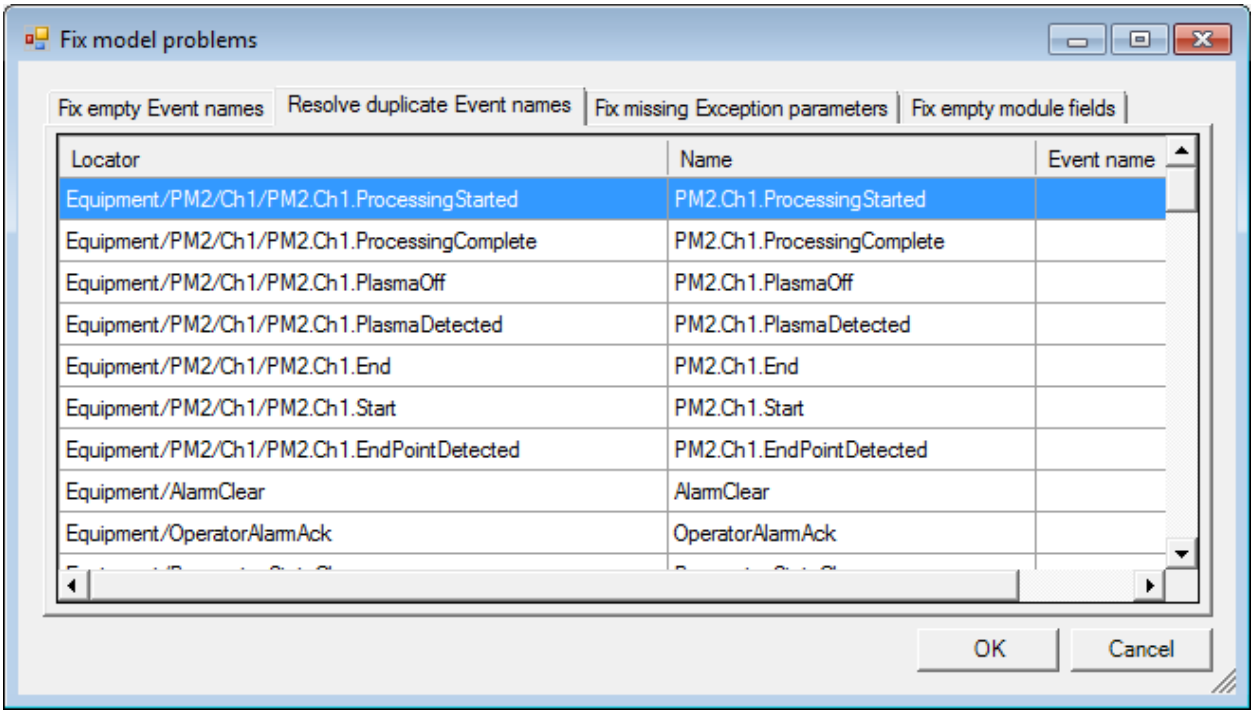
2.5.4.4.1 Fix empty event names

The Fix empty event names tab will simply assign the model's node name to the event name causing it to no longer be empty. They need to be further reviewed to ensure the model node name is the appropriate value.

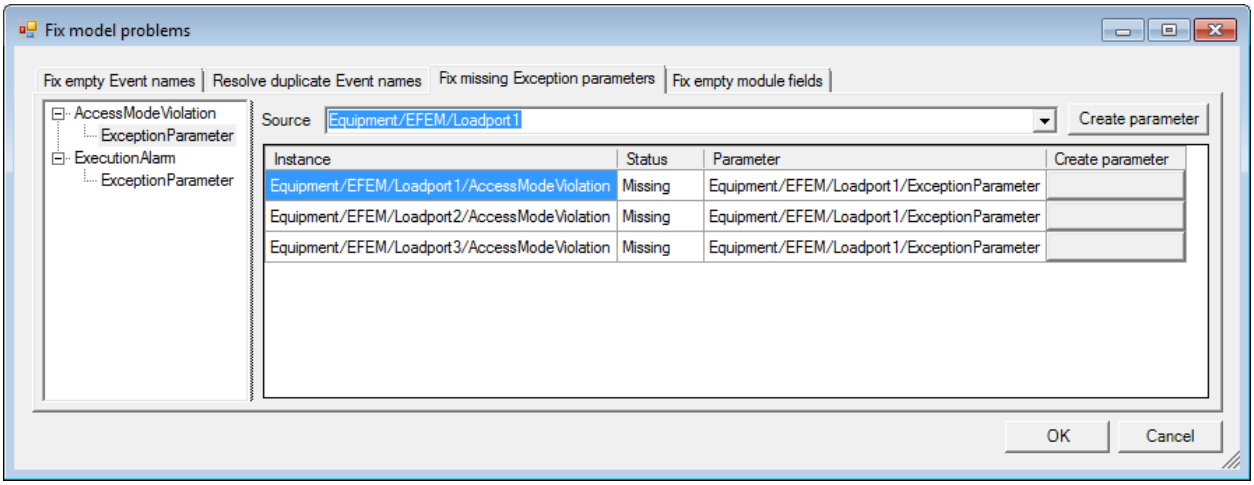


2.5.4.4.2 Resolve duplicate Event names

The Resolve duplicate Event names tab shows all the events with duplicate names and allows the user to assign new names to them.



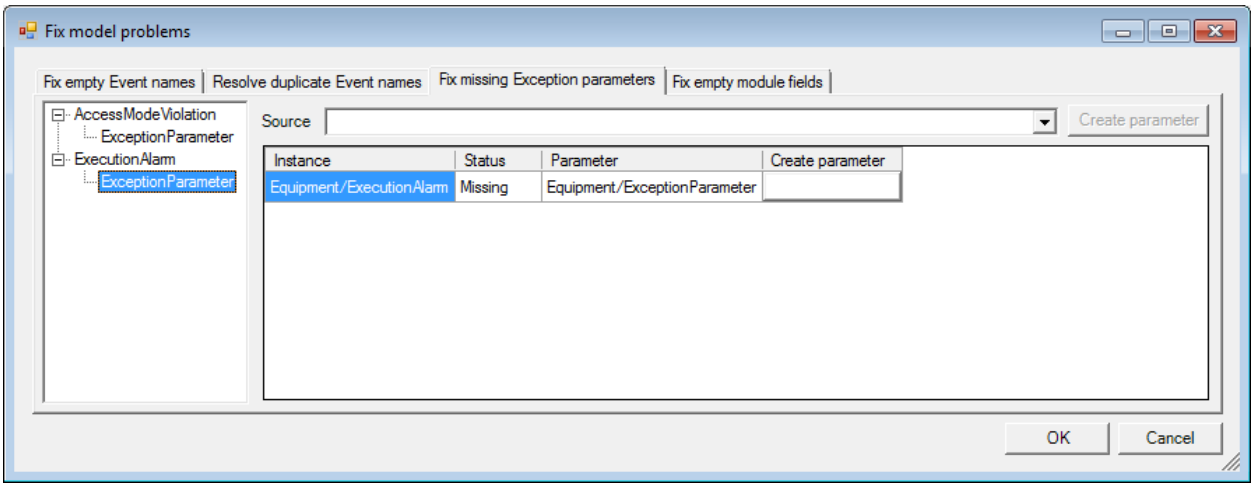
2.5.4.4.3 Fix missing Exception parameters



The Fix missing Exception parameters tab shows all the definitions with instances that have broken parameters. The left side is a tree view showing the definition and parameter hierarchy. The right side shows the parameter source at the top and the instances of that definition. Each instance has a Status indicator to show if that instance is correct, if the parameter is missing or if it will be fixed when the tool is closed.

This first screen shows an instance of a missing absolute parameter. In this case, the Source indicated does not exist and so all the instances have a missing status. To fix this type of condition, the Source is set to a valid location via the drop down list. The Create Parameter on

the right next to the Source will become enabled. When this is pressed, the Add Parameter wizard will display. When the wizard's information is completed and the OK button pressed, the Create Parameter will become disabled again and all the Status fields will change to Fixed.

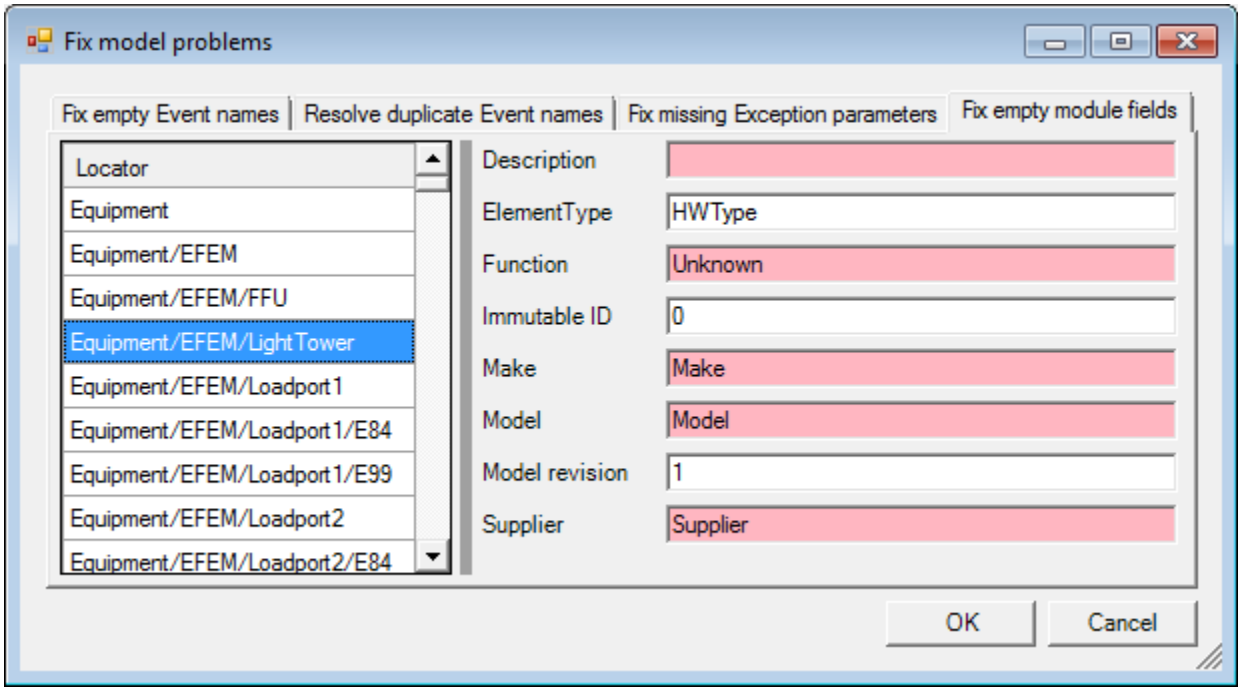


The second screen shows an instance of a missing relative parameter. The Source being empty indicates each instance should have its own parameter. In this particular example, the first instance is missing a parameter but the second one is OK. To fix this, press the Create Parameter button next to the broken instance in the right hand column. The Add Parameter wizard will display to allow selection the parameter information. When this wizard is closed, the Status will change to Fixed.

2.5.4.4.4 Fix empty module fields

Per E164 standard E164.00-RQ-00003-00, E120 nodes cannot have invalid values for the description, element type, function, immutable ID, make, model, model revision and supplier fields. Invalid values are defined as empty, unknown, n/a, na and none. This tab identifies model nodes where any of these fields have an invalid value and allows the user to change them.

The tab looks like this:



On the left side is a list of all the nodes that have at least one field with an invalid value. On the right side are the details for the selected node. To fix the fields, click on a field with a problem, indicated by the pink background, and change its text to an appropriate value. Fields with a white background have values that meet the E164 standard as not being invalid.

The gray bar separating the two sides is a splitter bar and may be adjusted side to side as needed.

2.5.4.5 Template Overview

This section discusses templates. A template contains the definition of a single model node and all of its children. The template can be applied to a parent node in the model. The node defined by the template becomes a child of the parent. The templates shipped with CIMPortal Plus are all used by the New Model wizard. The user may also use them independently from the wizard to add things to their model. Additionally, the user may create their own templates, for their own use. User created templates will not be used by the new model wizard.

2.5.4.5.1 Adding Templates to the Model

Templates are accessed through the Templates tab in the toolbox. This tab contains a list of templates arranged by type, available within EMDeveloper. A template is a well-formed, schema-validated part of an equipment model stored for use in future models. It defines a single model node and all of its children. Templates are stored in the basedir\Templates directory.

Templates are generally grouped with other similar templates.

To add a template to the model:

1. Select 'Templates' tab in the toolbox
2. Find the desired template by opening the tree view of the type you wish to add.
3. Drag the single item template with the mouse from 'Templates' toolbox and drop it onto a node in the model tree view. The node upon which the template is dropped becomes the parent for the template node.

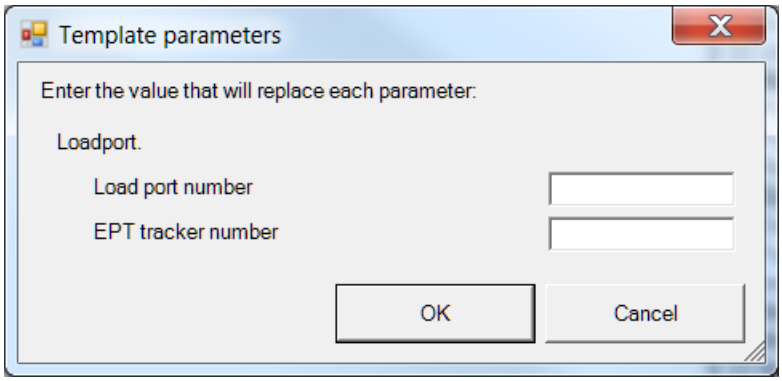
   -or-

   Drag the group template from the 'Templates' toolbox and drop it onto a node in the model tree view. The node upon which the template is dropped becomes the parent for each template in the group.

2.5.4.5.2 Adding Parameterized Templates

It is possible to create a template that can be used multiple times in the model but with different settings. For example, you may wish to create a loadport template that is used for more than one loadport. You can add a parameter to the template that allows you to change the loadport number when adding the template to the model.

When adding a parameterized template to the model, the user will be prompted to enter information for the parameterized portion of the template. As an example, look at the loadport template. The load port and EPT tracker numbers are parameterized. When the user attempts to add the loadport template to the model, the following prompt will be displayed:



The user then enters the value for the Load port number and EPT tracker number. Each time a load port template is used, different numbers should be entered. This allows a single template to be used for multiple objects in the model.

2.5.4.5.3 Template DCP Objects

When a template contains parameters, events or exceptions, the DCIM information (DCIMInstance and attributes) will have been removed. This allows the template to be used with multiple DCIMs. After a template has been added to the model, the user will need to fill in the DCIM information for all parameters, events and exceptions in the template. This can be done automatically.

1. Open the DCIM you wish to use by selecting it in the DCIM toolbox.
2. Select Match Empty DCIM from the Tools menu.
3. Select Match DCP Reference from the Tools menu

2.5.4.5.4 View a template's contents

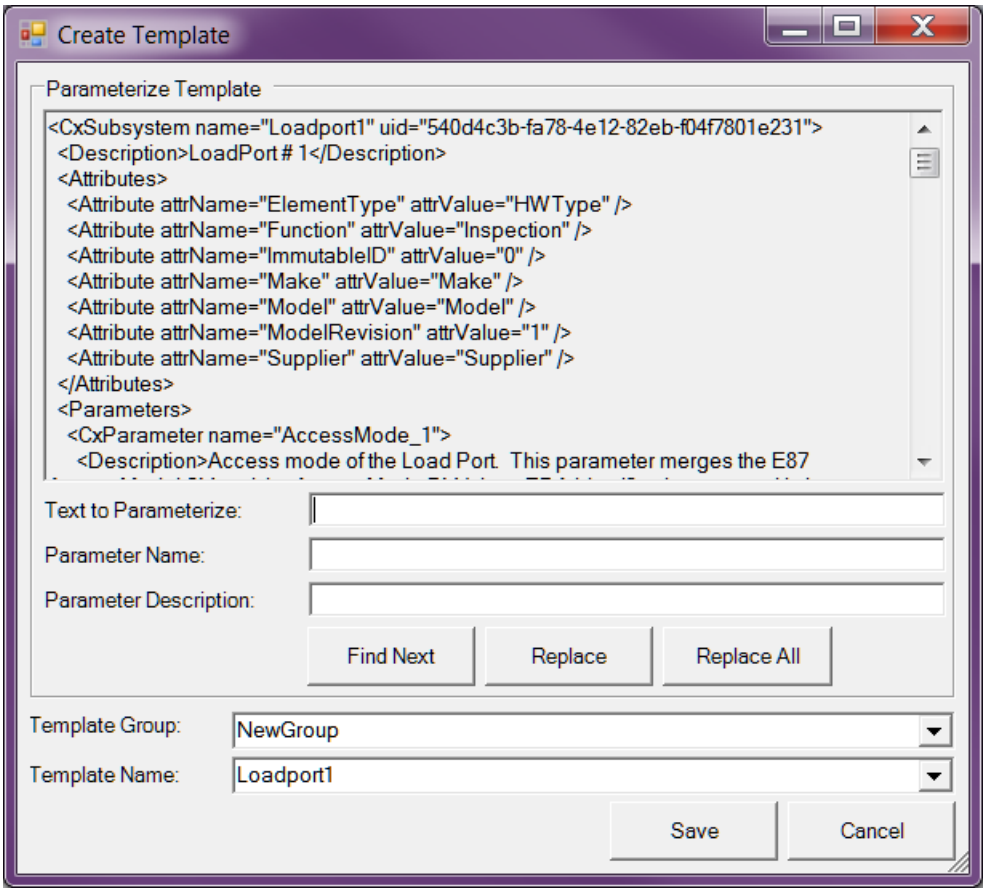
1. Select the 'Templates' tab in the toolbox
2. Find the desired template
3. Right click with the mouse on the desired template and select 'View...' from the context menu
4. The Template view form will display the XML of the selected template

2.5.4.5.5 Creating Templates

2.5.4.5.5.1 Create a Template from a Model Node

To create a template from a node in the model tree, right click the desired node and select 'Create Template' from the context menu. Note that not all node types can be used to make templates. If a node type would not make a valid template, the Create Template option is not visible in the context menu.

Selecting Create Template from the context menu displays the Create Template Dialog.



The XML for the template is displayed in the top text window.

When a template is created from a model, the UID attribute for hierarchy nodes and the DCIM information (DCIMInstance and attributes) for parameters, events and exceptions are cleared. When the template is added to a model, the UID for hierarchy nodes is automatically generated. The Match Empty DCIM and Match DCP References tools can be used to assign the DCIM information to the items in the model that came from the template. This flexibility allows templates to be used in different models with different DCIMs.

2.5.4.5.5.2 Parameterizing a Template

Before the template is saved, the user has the opportunity to parameterize the template. In the image above, the node to be made into a template is called Loadport1. Throughout the template, parameters and other objects associated with this load port have a 1 appended to them. For example, there is a parameter called AccessMode_1. It would be good to use the template for all load ports rather than just one. This is done by parameterizing the template. In other words, all instances of the string "1" are replaced with a variable. To do this, the user can either select the text to be parameterized from the template XML or type the text to parameterize in the "Text to Parameterize" field. The name of the parameter that will replace the selected text is then entered. In the above example, the user would enter “1” in the Text to Parameterize field and “LoadportNumber” in the Parameter Name field.

Each parameter can have an optional description that is displayed to the user to describe the parameter when the template is used. If the description is empty, then the parameter name will be shown to the user. If the description is not empty, it will be shown instead. If the same name is used with different descriptions, then the description becomes a type of subscript for the parameter name. This allows the multiple values within a single template for a single logical parameter. View the LoadLock2Location template for an example use of this feature.

Three buttons are provided to assist in the parameterization process.

- Replace - replaces the current instance of the text with the parameter name and locates the next instance.
- Replace All - replaces all instances of the text with the parameter name.
- Find Next - finds and highlights the next instance of the text in the template XML

When text is replaced with a parameter name and description, the parameter information is bracketed by at the beginning by %# and at the end by #%. In this case, the parameter AccessMode_1 will now read AccessMode_%#LoadportNumber#%.

2.5.4.5.5.3 Saving the Template

When the user is ready to create the template (parameterized or not) the group and name fields are entered on the bottom of the dialog. These values are filled in with defaults based on the source node. These values can be modified.

To assign a template to a group, the user may either enter a group name in the Template Group field or select an existing one from the drop down list. The group name is used as a subdirectory name for the template.

Templates are stored in the basedir\Templates directory. Groups are stored in subdirectories of this one. In the example above, the group name is entered as NewGroup the template name is Loadport1 and the template will be stored in basedir\Templates\NewGroup\Loadport1.xml.

Pressing the Save Template button writes a template file to the hard drive. In our example, the template would be stored as basedir\Templates\NewGroup\Loadport1.xml.

2.5.4.6 Loading a Model

When a model is initially loaded, the model is optionally converted to the current schema. Then the file is parsed and the internal model objects are built, and the model is displayed.

2.5.4.6.1 Mismatches between DCIM and Model

When building the model objects from the XML, if the DCP object is not found in the DCIM, then an error message will be generated. This error indicates that the DCIM has changed since the model was created and no longer supports DCP objects stored in the model. The error should be resolved to ensure that CIMPortal can report DCIM parameters, events, and exceptions to the client at runtime.

The error message states “DCP object named in model is not supported by the DCIM” will be displayed. The error will contain the name of the DCP object, the name of the DCIM and have a clickable locator to navigate to the node with the problem.

Either the node should be changed in the model to use a DCP object currently provided by the DCIM or the DCIM should be changed to provide the DCP object.

- In the case of AppDCIMs, this involves using the ADC editor to modify the ADC file.
- In the case of CIMConnectDCIMs, this involves editing the EPJ file.
- Other DCIMs will have their own methods of supplying this information.

2.5.5 Model Validation

Before a model can be packaged and deployed in CIMPortal Plus, final validation must be done on the model to ensure that CIMPortal Plus will be able to use it. Final validation does not guarantee the model is fully standards compliant, only that it passes certain sanity checks and doesn’t contain any gross mistakes. Final validation is accessed from the Packaging menu in EMDeveloper.

Final validation runs several types of validations including:

- Schema validation
- Node validation
- DCIM validation

- Reference validation
- State Machine validation
- SEMIObject validation
- Exception validation

If any of these validations fail, final validation fails.

If the model has any unsaved changes, the model is saved before final validation is run.

After a successful final validation, the model is marked as validated. A deployment package can then be made. If any changes are made to the model after final validation, final validation must be performed again before the model can be re-packaged.

2.5.5.1 Schema Validation

Schema validation ensures that the model adheres to the CxEquipmentModel.xsd schema file defined by Cimetrix. Schema validation should just be a formality during validation. EMDeveloper will not allow the creation of a model that does not adhere to the schema.

Schema Validation is done during final validation. If the model has unsaved changes, a prompt asks the user if he wishes to save the file. If no or cancel is selected, the schema validation is aborted. After the file is saved, the schema validation is performed. The output of the validation is written to the Validation report tab in the Output window.

Error and comment messages give the user a description of the problem and the node where it occurs. After fixing the issues, schema validation may be performed again.

2.5.5.2 Node Validation

Node validation checks:

UIDs are set.

All events have non-empty EventNames.

No duplicate EventNames exist.

No duplicate DCP objects.

2.5.5.2.1 Empty UIDs

Empty UIDs on the root equipment node is an error; it must be assigned. Empty UIDs on all other hierarchy nodes produce comments. These can be fixed by using the Tools | Generate Model UIDs menu item.

2.5.5.2.2 Empty Event name

This is an error. To fix, click on the error message to move to the event in error. In the EventName property, enter the appropriate value.

2.5.5.2.3 Duplicate event name

Within a hierarchy node, there cannot be any sibling events with the same name. This is an error. To fix, change one of the event's EventName properties to not conflict or delete one of the events.

2.5.5.2.4 Duplicate DCP object

Within a hierarchy node, there cannot be any events with the same DCP object. This is an error. To fix, drag and drop a different event DCP object from the DCIM instance to update the offending event or delete one of the events to remove the duplicate.

2.5.5.3 DCIM Validation

The purpose of DCIM validation is to ensure that every Data Collection Point object (Event, Exception and Parameter) in the model has a DCIM Instance associated with it and that the DCIM is reporting acceptable data.

2.5.5.3.1 Fixing DCIM comments

One of the common comments that may be generated is that GEM IDs are not set (this includes Set and Clear events for exceptions). Earlier versions of EMDeveloper allowed direct editing of these values. However, this caused the DCIM (the publisher of the data) to be out of sync with the copy of the data in the model generated by EMDeveloper so this facility was removed. Instead, the accepted way to set GEM ID fields is to update the DCIM so it is reporting the IDs correctly and then run the Resync DCIM/DCP References tool. For App DCIM based DCIMs, this involves editing the ADC file (see section 2.6 of this document). For CIMConnect based DCIMs, this involves updating the associated EPJ file. For other DCIMs, refer to that DCIM's documentation for further details.

2.5.5.3.2 Fixing DCIM Validation Failure

If DCIM validation fails, all failures in the model are reported. The user may go to the failed Event, Exception or Parameter in the model by clicking on the line in the validation report. A DCIM instance may be applied to the element as outlined in Data Collection Elements.

2.5.5.3.3 Match Empty DCIM

EMDeveloper provides a tool for filling in DCIM Instance information for all Events, Exceptions and Parameters that are missing this information. The tool is called Match Empty DCIM and can be activated from the Tools menu.

Match Empty DCIM searches the model for elements with empty DCIM Instance information. When it finds one, it searches the list of DCIM Instances that have been activated in EMDeveloper. If the element type (Event, Exception or Parameter) and name match, the element with empty DCIM Information in the model is assigned the DCIM Instance information from the DCIM Instance in which it finds the element. Note that for parameters, the parameter type and transient type are not updated to match the DCP object. The values that are set in the model are preserved. If these are different from the model's current values and should be updated, the Resync DCP object tool can be subsequently run with the two "Preserve" options unchecked.

If a match is made, a success message is logged to the output tab. If a matching element could not be found in any of the active DCIM instances, an error message is logged instead.

If more than one DCIM is instantiated that contains a match for the parameter, event or exception in question, the user is prompted to select the DCIM to use for the match.

If the user wishes to use the same DCIM for all found duplicates, then check the "Use setting for all duplicates" box. Pressing OK assigns the selected DCIM to the item in question.

It may be that no match is found. There are several ways to resolve this issue.

1. The offending element can be deleted.
2. A DCIM Instance with a matching element can be activated (see section 2.5.3, Using DCIM Instances) and the Match Empty DCIM tool re-run.
3. The name of the element can be changed in the model to match a name of an object in an active DCIM instance.
4. An object from a DCIM can be manually selected by pressing the button next to the DCPObject property in the node's property box and selecting the DCIM, category and object from the dialog's drop down lists.
5. The DCIM can be changed to support the needed DCP objects and the Match Empty DCIM tool rerun.

If all empty DCIM are matched or the output window does not show any empty elements, the DCIM validation succeeded.

When the tool finishes and there were errors reported, the user will be prompted to save the missing DCP objects to a file. This is to help facilitate item 5 above. Some DCIMs support importing the DCP objects into their internal object system. If the user does not wish to do this, simply hit Cancel for the save dialog and the file will not be saved.

2.5.5.4 Reference Validation

The model may contain several types of references:

- Parameter References
- Event References
- Parameter Type References
- Unit References

Reference validation traverses the model to ensure that all such items reference a valid object in the model.

If reference validation fails, all failures in the model are reported. Clicking the error message will take the user to the node on which the failure occurs. This section discusses methods for correcting reference validation failure.

2.5.5.4.1 Fixing Parameter and Event Reference Failures

The easiest way to correct parameter and event reference failure is to use the Match DCP References tool from the Tools menu. This tool searches the model for all event and parameter references. If any reference is invalid, it will search the model for an element of the correct type (event or parameter) and name. If it is found, the reference is resolved. If a matching element is not found, the tool then searches all active DCIM instances for an element with the correct type and name. If one is found, the element definition either replaces the invalid reference or is added to the nearest hierarchy node parent and the reference is resolved. If a matching element is not found in an active DCIM instance, an error message is reported to the output tab indicating the reference could not be matched.

If EMDeveloper is unable to match a DCP reference, the user may:

1. Delete the unmatched element.
2. Activate a DCIM Instance with a matching element (see section 2.5.3, Using DCIM Instances) and re-running the Match DCP References tool.
3. An object from a DCIM can be manually selected by pressing the button next to the DCPObject property in the node's property box and selecting the DCIM, category and object from the dialog's drop down lists.

2.5.5.4.2 Fixing Unit Reference Failures

ByteType, ShortType, IntegerType, LongType, FloatType and DoubleType parameter types require a reference to a unit. If this reference is null, or the unit referenced does not exist in the model, unit reference validation will fail. To fix this failure:

1. Add Unit to be referenced if necessary.
2. Go to the parameter type that failed validation by clicking on the message in the validation report.
3. In the properties window for the parameter type, select the desired unit type in the UnitRef drop down list.

2.5.5.4.3 Fixing AvailableUnit Reference Failures

To be valid, AvailableUnits:

- Must have at least one UnitConfig.

- All UnitConfig unitIDs must reference an existing unit in the model.
- Must refer to an enumerated string parameter type for the settings.
- All UnitConfig settings must reference a string in the enumerated string parameter type.
- unitParameter must be set.

If any of these conditions are not met, reference validation fails. Correct the items listed in the validation report and run reference validation again.

2.5.5.4.4 Fixing ParameterType Reference Failures

Another type of reference failure may be a parameter type reference failure. Parameter types can be referenced in four places:

- Parameters
- SEMIObject E39Attributes
- ArrayType parameter types
- StructureType structure elements

If any of these locations reference a parameter type that doesn't exist or have a null reference, reference validation will fail. To fix the reference validation failure in these cases:

1. Add ParameterType Definition to be referenced if necessary.
2. Go to the element that failed validation by clicking on the message in the validation report.
3. In the properties window for the problem element, select the desired parameter type to be referenced from the drop down list.

After fixing all reference failures, reference validation will succeed for the model.

2.5.5.4.5 Transient Parameters

Transient parameters should only be added to E39 Objects (SEMIObject types). Reference validation traverses the model and displays a comment if any transient parameter is found outside of one of these objects. Reference validation succeeds if only comments are present.

2.5.5.5 State Machine Validation

State machine validation checks all state machines and state machine instances in the model for validity.

A state machine is valid if:

- It has at least one state.
- It has at least one transition.
- Transitions reference valid source and target states in the state machine.
- Transitions do not reference the top level state as a source or target state.
- All transitions within the state machine have unique transition numbers.
- It has only one top level state.

Fix state machine validation failures by supplying part missing that causes the failure.

2.5.5.6 State Machine Instance Validation

A state machine instance is valid if:

- The state machine from which it was created is defined in the model.
- All events in the state machine instance are defined in the state machine from which it was created.
- All event references reference defined events.

- Location ID value string is valid.
- Hierarchy of State Machine Instances matches the corresponding State Machine hierarchy.

State machine validation traverses the model to make sure that all state machines and state machine instances in the model are valid.

2.5.5.6.1 Fixing Invalid State Machine Instances

If validation detects that a state machine instance references a state machine that is not in the model, this should only be fixed by deleting the state machine instance. It is possible to add a state machine with name referenced by the state machine instance, but the event map may not be correct for the state machine instance. It is recommended to remove the state machine instance and add it again after the state machine is created.

2.5.5.6.2 State Machine Validation Comments

Two other issues can be uncovered during state machine validation. These are not necessarily critical issues, but they are issues that the user needs to be aware of.

- State machine does not have a state machine instance in the model.
- State machine instance event map has no available parameters.

State machine validation will succeed if only comments are found.

2.5.5.6.2.1 Missing State Machine Instance

In order for event data to be meaningful, the state machine must have at least one instance in the model. The instance provides context for what the collected data means. It is perfectly legal to have a state machine definition without a state machine instance in the model, but in order to collect data for the state machine, an instance must exist.

If a state machine is found in the model without a corresponding state machine instance, a comment message is logged. This situation can be resolved by either deleting the state machine definition or adding a state machine instance for it. If nothing is done, validation will still succeed.

2.5.5.6.2.2 Missing Available Parameters

During state machine validation, all state machine instances are checked to see if any event map has zero available parameters. Available parameters are not required for event maps, but if any empty ones are found, a comment is logged. A path to the event map with no available parameters is also listed.

Available parameters can be added to event maps as described in Adding Parameters.

2.5.5.7 SEMIObject Validation

SEMIObject validation checks that:

- SEMIObject type has at least one E39Attribute
- SEMIObject type has E39Attributes called ObjID and ObjType
- SEMIObject type stores the name of the DCIMInstance that services the object
- SEMIObject type has a valid InstParameter
- SEMIObject type has a valid, non-empty Standard string
- E39Attributes do not have null reference to a parameter type
- E39Attributes reference valid parameter types defined in the model
- SEMIObject Instances reference a SEMIObject type defined in the model
- SEMIObject Instances have a valid Location ID Value string

2.5.5.7.1 Fixing SEMIObject Validation Failures

SEMIObject validation must succeed before a model can be deployed for use with CIMPortal Plus. This section describes how to fix each type of validation failure.

2.5.5.7.1.1 Missing E39Attribute

1. Click the error message in the validation report.
2. Add at least one E39Attribute to the SEMIObject type as described in Add SEMIObject.

2.5.5.7.1.2 Missing DCIMInstance Name

1. Click the error message.
2. Select the desired DCIM instance name from the drop list in the properties grid.

-or-

Run Match Empty DCIM from the tools menu. If there is only one DCIM instantiated, it will automatically be assigned to the SEMIObject.

2.5.5.7.1.3 Empty Standard string

SEMIObject types must have a SEMI standard string specified for the object.

1. Click the error message in the validation report.
2. Enter the desired standard string for the object.

Note: The standard string must start with an alphabetic character. This must be followed by a number a dash and another number. An example of a valid standard string is E87-0305.

2.5.5.7.1.4 Null or Invalid Parameter Type Reference

1. Add a parameter type definition to the model if necessary.
2. Click the error message.
3. Select the desired parameter type from the drop down list in the ParameterType field of the properties window for the E39Attribute.

2.5.5.7.1.5 SEMIObject not defined for SEMIObject instance

If validation detects that a SEMIObject instance references a SEMIObject type that is not in the model, this should only be fixed by deleting the SEMIObject instance. It is possible to add a SEMIObject type with name referenced by the SEMIObject instance, but this is not recommended. It is better to remove the SEMIObject instance and add it again after the SEMIObject type is defined.

2.5.5.8 Exception Validation

Exception validation checks that:

- Definitions have at least one instance.
- Definitions have a severity assigned.
- Instances have definitions.
- Severity in the instance's DCP object is the same as the definition's severity.
- Instance has parameters for each of the exception parameters in the definition.
- SetEventGemID and ClearEventGemID in the instance's DCP object are defined.

2.5.5.8.1 Definitions have at least one instance

This is a comment only. To fix this comment, create an instance for the indicated definition.

2.5.5.8.2 Definitions must have a severity

This is an error and must be fixed prior to package creation. To fix, click on the error message and select a severity from the drop down list.

2.5.5.8.3 Instances have a definition

This is an error and must be fixed prior to package creation. To fix, create a definition for the instance if one does not already exist. Click on the error message to move to the instance with the missing definition. Select a definition from the drop down list in the DefinitionName property.

2.5.5.8.4 Severity mismatch between the definition and the instance

This is a comment and should be fixed prior to package creation. The fix can be done either in the definition or the instance, depending on which one is wrong.

If the definition is wrong, click the error message to move to the instance and then right click and select "Go To Definition" to move to the exception definition. Then select the correct severity from the drop down list in the Severity property.

If the instance is wrong, select the DCP object from the DCP object selection dialog to update the DCP object properties. If the DCP object property is wrong, follow the instructions for editing the DCP object properties as outlined in section 2.5.5.9, changing the severity as appropriate.

2.5.5.8.5 Missing parameters

This is an error and must be fixed prior to package creation. To fix, click on the error message to move to the instance with the missing parameters.

There are two types of parameters: absolutely referenced parameters and relatively referenced parameters. The type of reference is determined by the Source property of the Exception Parameter in the Exception Definition. For more information, see section 2.5.4.1.5 on model building exceptions, instances and parameters.

If Source is empty, then each exception instance must have a parameter that matches the name of the Exception Parameter. In this case, create a parameter in the exception instances' parents as needed to fix any errors.

If the Source has a locator path to a model node in it, add a parameter to that model node with the same name as the Exception Parameter.

2.5.5.8.6 SetEventGemId and/or ClearEventGemId fields are not set

This is a comment. To fix, follow the instructions for editing the DCP object properties as outlined below, changing the SetEventGemID and ClearEventGemID properties as appropriate.

2.5.5.9 Editing DCP object properties

Earlier versions of Equipment Model Developer allowed DCP properties to be edited from a dialog in the DCIM Administrator. This did not work too well and so has been removed. (See section 2.5.5.3.1.) The ADC Editor is the accepted way to update the DCP properties for App DCIM based DCIMs. Other DCIMs will have their own ways of defining and editing DCP objects.

For App DCIM based DCIMs, to update DCP object properties, open the DCIM Instance's ADC file in the ADC Editor and set the field's values as appropriate. Save the updated ADC file and reopen Equipment Model Developer.

The DCP objects in the model nodes can be updated one of two ways. The first way is to use the property editor for the DCPObject on each node that needs to be updated. The second and easier way is to select the Tools | Resync DCIM/DCP reference menu item. This tool will update all the DCP objects in the

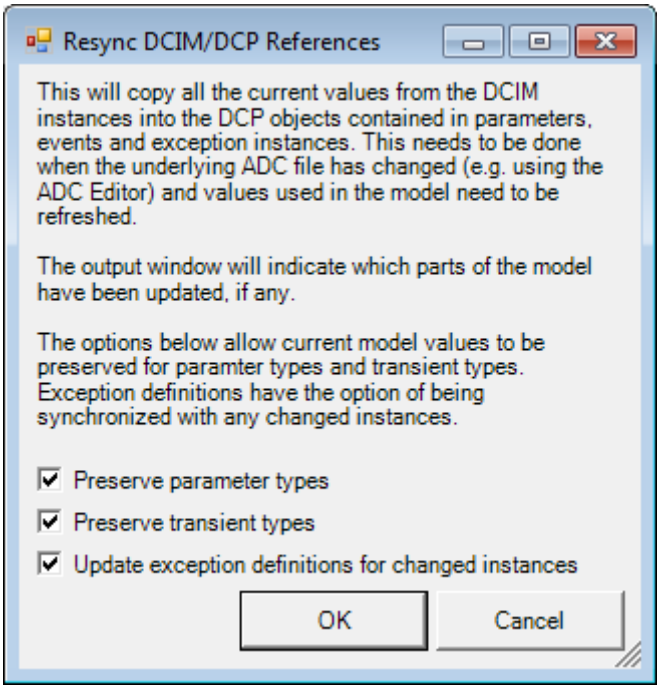
model with the current values from the DCIM. Any model nodes that were changed will be displayed in the output window.

There are three model properties that are impacted by the DCP object. During the course of model developerment, the user may change these from the values provided by the DCP object. When the resync operation is performed, the user may or may not want these values to change. Therefore the resync tool has options to save these values during the resync operation. Because the user has made these changes specifically from the default values, the tool defaults to preserve the user's work.

The DCP object contains a field containing the parameter type. Equipment Model Developer allows the user to change this. When this is changed, it's changing the value contained in the DCP object stored by the model.

The DCP object contains a Boolean field indicating if it's transient or not. The model contains an enumeration as to whether a parameter is transient, restricted or unrestricted. When a DCP object is assigned to a parameter, the parameter is set to transient or unrestricted based on this Boolean flag. The user may change this to a different value, based on the needs of their model that does not match the DCP object's isTransient flag. The most common example would be a change from Unrestricted to Restricted, both considered non-transient (false in the DCP object's context).

The DCP object contains a field containing exceptions' severity. When an exception instance is created and a corresponding exception definition does not exist, the exception definition may be created automatically. When this is done, the exception definition's severity property is set to match the severity contained in the DCP object for the exception instance. In general, these should always match. If they don't match an exception comment is created during final validation.



The "Preserve parameter types" and "Preserve transient types" options, when checked will keep the model's parameter types and transient types from being updated by the DCP object's sync operation. If unchecked, when a DCP object is sync'd, the parameter types and transient types in the model will be updated to match the DCP object.

The "Update exception definitions for changed instances" will update the severity in the exception definition to match the updated exception instances. This will keep the final validation from emitting a comment message.

The Resync tool will only update DCP objects if the value in the model and the value provided by the DCIM do not match. Any nodes where they have been updated will be detailed in the Output tab at the bottom of the screen.

2.5.6 Model Package Creation

Once a model is validated, a deployment package can be created through the 'Packaging | Create Deployment Package...' menu item. Create Deployment Package is used to create Runtime Deployment Package files for later use by CIMPortal Plus. By default, the equipment name is the name of the validated model file (without extension) currently loaded in EMDeveloper.

1. Change the equipment name if desired. The equipment name from this form will be the name of the deployed equipment in CIMPortal Plus.
2. Enter a password used to encrypt the runtime deployment package. This password must be used during the deployment from the CIMPortal Plus Control Panel.
3. Confirm the password.
4. Press the Create Package button. A Save File Dialog will be displayed to allow the user to specify the file to which the package will be saved. The package should be saved in the basedir\Deployment Packages directory.

The Runtime Deployment Package contains the Equipment Model XML file, DCIM Packages and DCIM Instances with all the supporting files that are defined in the Equipment. The Runtime Deployment Package file is encrypted with a password during creation. The password is required by the CIMPortal Plus Control Panel to deploy the package on a runtime CIMPortal Plus installation.

During deployment, the Equipment Model XML file is extracted into an encrypted file with .encxml extension that only CIMPortal Plus can read. All of the DCIM and DCIM Instance Files are extracted from the Runtime Deployment Package. DCIM Package files are registered using their Install_XXX.bat files. CIMPortal Plus updates the CIMPortal.cfg file and the running instance of CIMPortal Plus with the new equipment.

2.5.7 Creating ISMI Common Metadata-compliant model

The purpose of the E164 specification is to encourage and promote those companies using EDA/Interface A connections to use a more common representation of equipment metadata that is based upon the SEMI E125 Specification for Equipment Self-Description. This will help establish more consistency from tool to tool and from fab to fab, making it easier for equipment vendors to provide a consistent EDA interface and for fabs to develop EDA clients.

The standard was developed because semiconductor equipment suppliers were developing equipment models that were compliant with the E125 standard, but very different from one piece of equipment to the next. Even similar types of equipment had different models, which produced different metadata sets.

Cimetrix does not provide tools for validating equipment models for compliance with either Equipment Data Acquisition (EDA) Version 1105 and 0710 Guidance documents or SEMI E164 Specification for EDA Common Metadata. This section provides a guideline on creating equipment model that is compliant with E164 or the ISMI EDA Guidance documents.

- Download and read Equipment Data Acquisition (EDA) Version 0710 Guidance document from ISMI.
- Download ISMI/NIST Metadata Conformance Analyzer (MCA) tool.
- Start building of the CIMPortal Plus model using EMDeveloper's New Model Wizard. The New Model Wizard was designed to be a good starting point for creating 0710 Common Metadata-compliant models.
- Following EDA Guidance document finish building the CIMPortal Plus model

- Equipment model should contain only one equipment node (at the top of the model) and the UID for the Equipment node must be unique amongst the equipment in the fab. This requires a unique deployment package (or hardware.xml file) for each shipped piece of equipment.
- Names or ids for nodes should not embed information about the owner or type. PM1GasBoxSubsystem should be GasBox.
- Supplier, make, model, modelRevision, function and immutableId are conditional. They should be specified if the part has a serial number or equivalent.
- For hierarchy nodes, the elementType should be one of the strings in section 8.5.3.2.2 of E120.
- Metadata for identical equipment should be the same except for UID & Immutable ID.
- All collection parameters defined for the SECS interface should be available for collection by EDA.
- All numeric sensor data to be reported as floating point values with defined units.
- Any alarms reported to the operator and GEM host should be modeled as Exceptions.

- Validate the model and create a CIMPortal Plus deployment package file.
- Deploy equipment in CIMPortal Plus.
- Use EDAConnect sample application or ECCE2 to connect to CIMPortal Plus and create 7 metadata files.
- Run MCA tool and load metadata set specifying folder where 7 metadata files were created in previous step.
- MCA will load and analyze the equipment model displaying any errors and warnings it found in the model.
- Compare reported issues to the reference standards. The MCA tool sometimes reports false positives (things flagged as errors that really are not). For any true problems, fix them in EMDeveloper and repeat the steps starting with model validation until desired conformance level is achieved in MCA.

For more details on validating and deploying models and creating the metadata, please see the section called Collecting Metadata toward the end of this document.

2.5.8 Modeling Best Practices

This section gives a quick overview of modeling rules and conventions useful in building equipment models. It is not intended to be an exhaustive training guide. Rather, it is intended as a quick reference guide for different types of modeling decisions. It does not cover all node types. For detailed modeling information please see the CIMPortal Plus Developers Guide and the E120 and E125 standards documents.

2.5.8.1 E120 Common Equipment Model

- Equipment – Top level node that must contain at least one Module, Subsystem or IODevice but may contain zero or more of each.
  - For E164 the Equipment must contain a few pre-defined Parameter nodes
  - Must have all valid EquipmentElement properties
  - Must contain a LogicalElement that defines GEM 300 standard state machines and SEMIObjects.
- Module – capable of processing material. The module or one of its sub-modules or subsystems must contain at least one MaterialLocation.

- Subsystem – capable holding material (can have MaterialLocation elements), but is not capable of processing material. Examples would be Robots, LoadLocks, LoadPorts. If a subsystem only defines parameters, events, exceptions and state machines, it is recommended that an IODevice be used to model the element rather than a subsystem. A subsystem may not define a sub-module.
- IODevice – models sensors, actuators and devices. If an IODevice must have sub-IODevices, it should be modeled as a subsystem.
- MaterialLocation – provides the ability of equipment components to hold material. If a module or subsystem is capable of holding material (Carrier, Substrate, ProcessDurable, Consumable, Other) it should contain at least one MaterialLocation.
    - For E164, MaterialLocation types with material specified as Substrate must also implement the E90 SubstrateLocationStateModel.
- LogicalElement – Contains non-structural elements such as State Machines and SEMI Objects with their associated events and parameters that describe a complete element. Examples of logical elements are MaterialManagers, JobManagers, etc.

2.5.8.2 E125 Equipment Metadata

- StateMachine – a collection of states and transitions that comprise a finite state machine.
    - State machines shall have a single top level state that has the same name as the StateMachine. The name may be different from the ID.
    - All defined states, except the top state, should be used in at least one transition in the state machine.
    - For E164 compliance, Transition names should be an integer followed by an optional letter (ex: 1, 2, 3a, 3b).
    - Events should only describe one transition
    - Most state machines will have a no-state state that can be used as an initial or final state in the state machine.
    - It is NOT recommended that each device (valve, pump, etc.) have a state machine.
    - Parameters may be added to state machines, but may make more sense to be added to the defined state machine instance.
    - Parameters associated with StateMachine Events should generally be Transient parameters – or parameters that only have valid values when the event is triggered.
- StateMachineInstance – used to declare that an equipment node supports a specific state machine. Each node that supports a given state machine may report different data for those events.
- SimpleEvents – A SimpleEvent is a new equipment metadata item introduced in Freeze II (E125 0710). A SimpleEvent is an event generated by the equipment that is not within the context of a state machine, but is associated with an equipment HierarchyNode. This type of event was not permitted in Freeze I (1105), which required all events to be associated with a state machine.
    - Parameters associated with SimpleEvents should generally be Transient parameters – or parameters that only have valid values when the event is triggered.
    - If the SimpleEvent has a corresponding CEID (Collection Event ID for a SECS/GEM Interface), then the GemId property must be set to the CEID value.
    - In CIMPortal Plus' EMDeveloper, an equipment model can define SimpleEvents and still be compliant with both Freeze I and Freeze II. The two versions of CIMWeb, i.e. CIMWeb1105 and CIMWeb0710, each exposes the SimpleEvent definitions to the EDA client in a compliant format. Because SimpleEvents are actually not permitted in Freeze I, CIMWeb1105 will automatically generate a StateMachine and StateMachineInstance for each SimpleEvent under each HierarchyNode in the model. All SimpleEvents for a HierarchyNode are added to one StateMachine. The automatically generated StateMachines

will be named SimpleEventX where X is an incrementing number starting at 0. In Freeze II where SimpleEvents are permitted, CIMWeb0710 will expose these events as SimpleEvents under the HierarchyNode.

- ExceptionDefinition/ExceptionInstance – All exceptions may be defined once and only once in the model. ExceptionDefinitions may be referenced multiple times by using ExceptionInstances in the model. ExceptionInstances should have unique underlying definitions. For example an ExceptionDefinition may define AccessModeViolation alarm for load ports. Each load port could have an ExceptionInstance that references the definition, but the underlying object for the instance should be unique. For example load port 1 may have an exception instance object of AccessModeViolation_1.
- ExceptionDefinitions may also have ExceptionParameters. ExceptionParameters are not actual parameter definitions, but must reference parameters that already exist in the model. If an exception can reference multiple parameters with the same name, the SourceID field should be left blank.

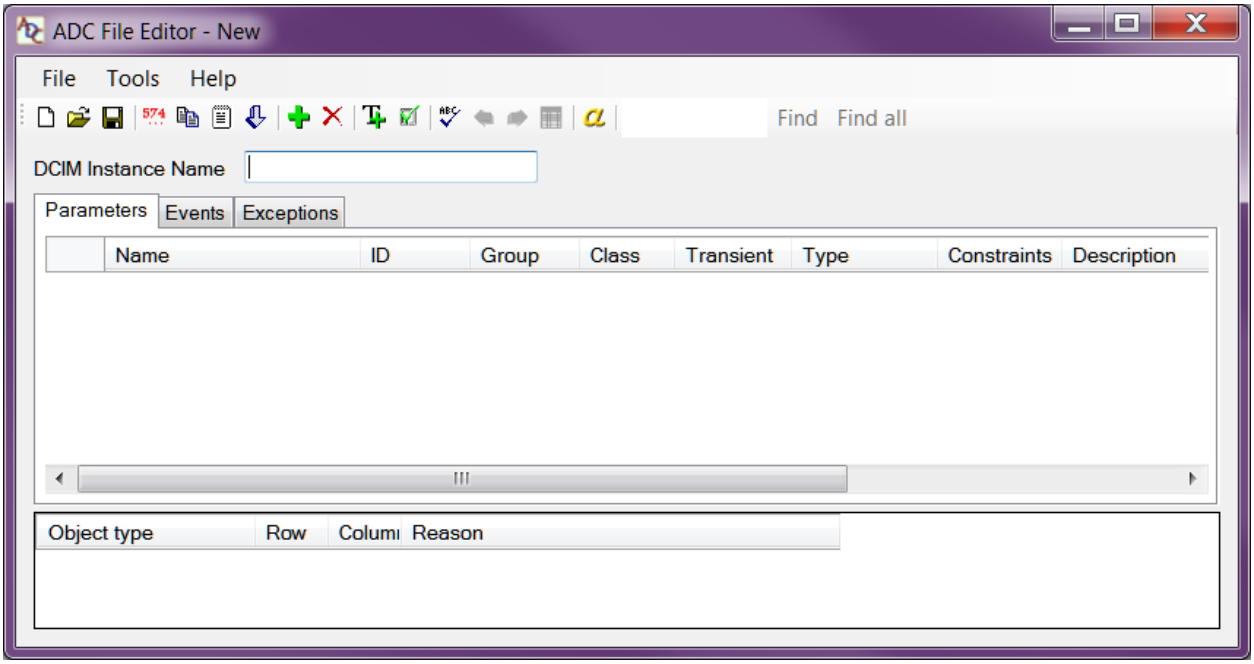
2.6 ADC Editor

The ADC Editor allows the user to conveniently edit the ADC file used by variants of the AppDCIM. In the past, the AppDCIM accessible through Equipment Model Developer's DCIM Administration dialog provided an ADC editor, however this editor was cumbersome to use and did not support the features needed for the new 0710 standards. Therefore, that method of editing the information was removed and the ADC Editor updated. It does the following:

- Shows and allows editing of all user-defined information.
- Makes it quick and convenient to add multiple new events, exceptions and parameters.
- Groups of events, exceptions and parameters.
- Enforces data integrity by allowing the user to select only defined parameter type definitions that are unit-specific.
- The ADC Editor runs as a stand-alone application, outside of Equipment Model Developer.
- Provides active validation mechanism for the current file. When an error condition is created, a clear error message is immediately displayed. The user can easily move to lines with problems by double-clicking the error message.
- When the file is saved, it is validated. If an error is found, it will ask the user if it should be saved anyway, thus allowing partial work to be saved for later editing.
- The following tools are available to the user:

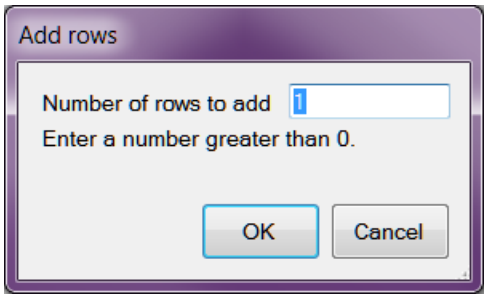
    - **Validate** – Show problems with the current contents. The current file is checked for known types of problems and any issues shown at the bottom of the screen. Double clicking on reported errors will navigate to the offending line. Typically, this tool does not need to be manually run as it is auto-run after any changes so the list is always up to date.
    - **Generate IDs** – Select rows, input a start number and step. The selected objects will have their ID set based on the input. If no rows are selected, it will generate IDs for the entire file.
    - **Duplicate** – Select a set of rows, and copy them to the bottom of the list.
    - **Remove items with duplicate…** – This tool will delete lines that have duplicate names or IDs, depending on which sub-menu is selected. This only works on the selected tab, so, for example, events will not be removed if parameters are shown.
    - **Replace Range** – Select rows or cells, input a search string, and replace with a new string.
    - **Import ADC file** – Merge an ADC with the opened file.
    - **Fill** – Select rows or cells, and put any valid string into selected fields. ID fields cannot be filled with this tool. Parameter Classes, Parameter Types, and Transient properties have alternative Fill mechanisms.

- **Add Rows** – Allows the user to enter any number of new empty objects to add to the bottom of the list.
- **Add Type** – Allows the user to add custom Parameter Types.

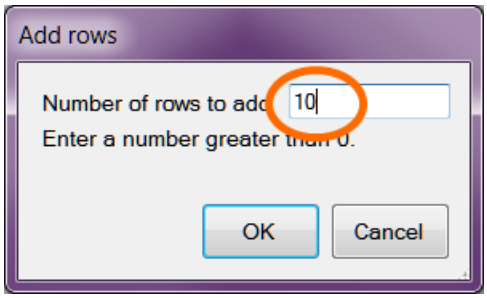


2.6.1 Add Rows

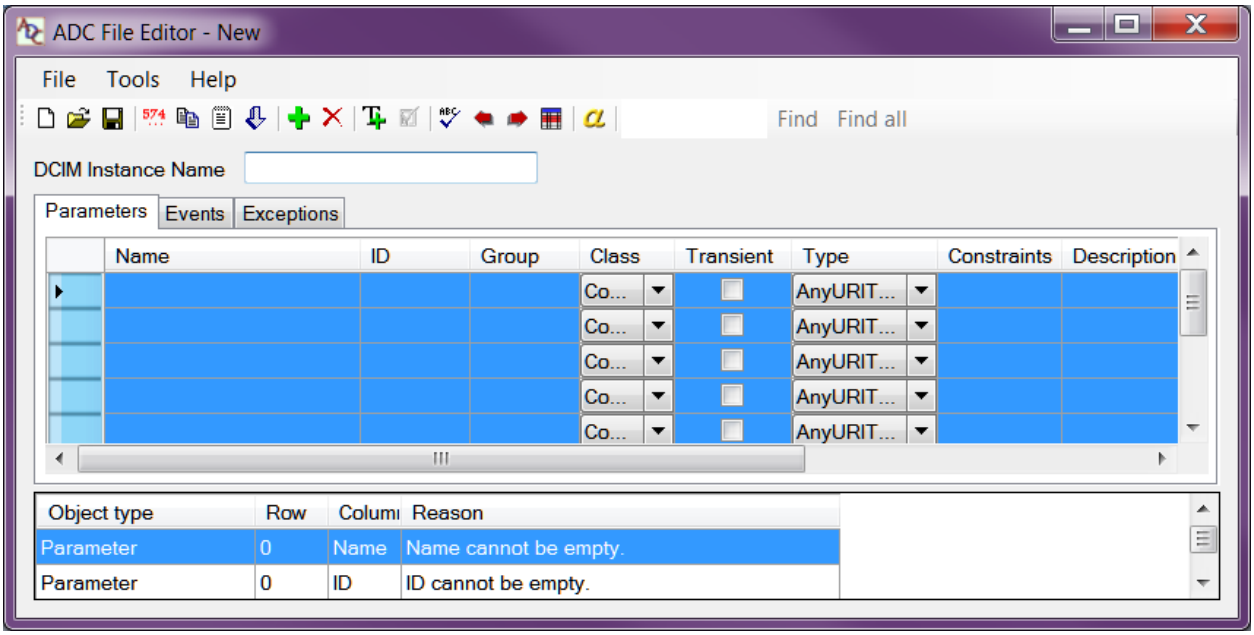
Press the Add Rows button.



Input the number of new rows you wish to add, and press OK.

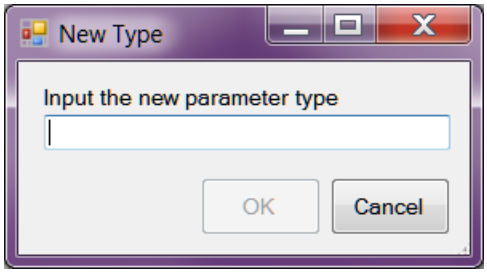


The new empty rows will be added and highlighted at the bottom of the list. Note the errors indicated in the error list at the bottom. As they problems are corrected, they will automatically be removed from the list.

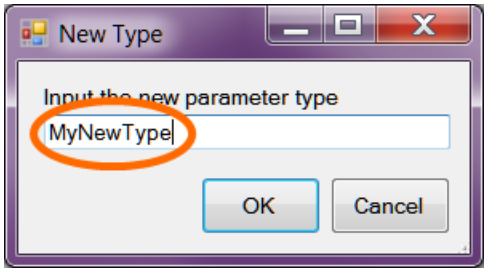


2.6.2 Add Parameter Type

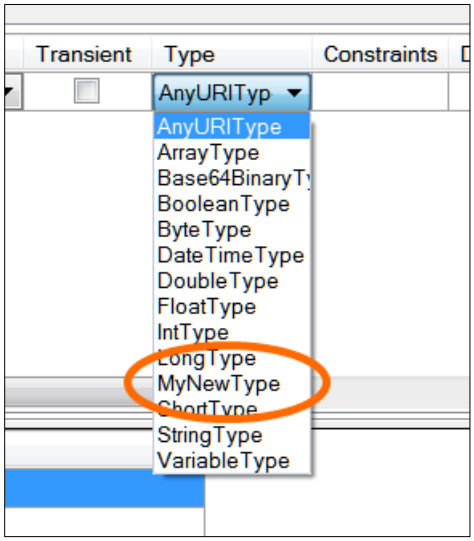
Press the Add Type button.



Input the new parameter type, and press OK.

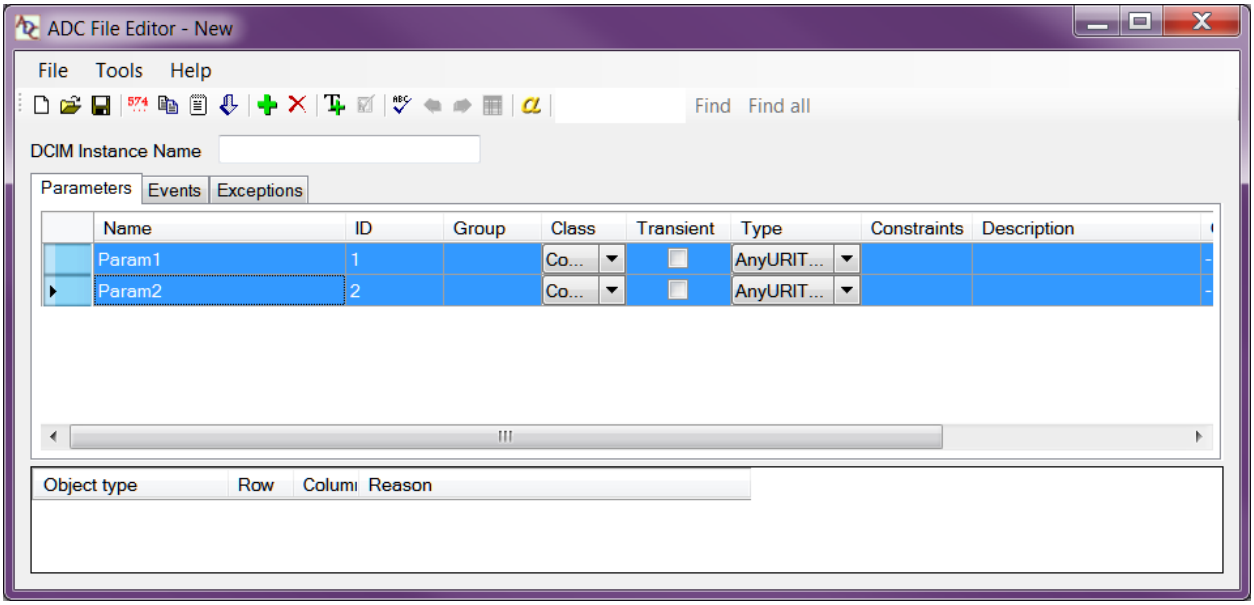


The new type will be added to the possible choices here.

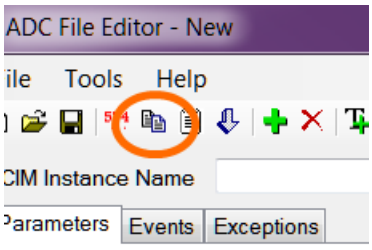


2.6.3 Duplicate Rows

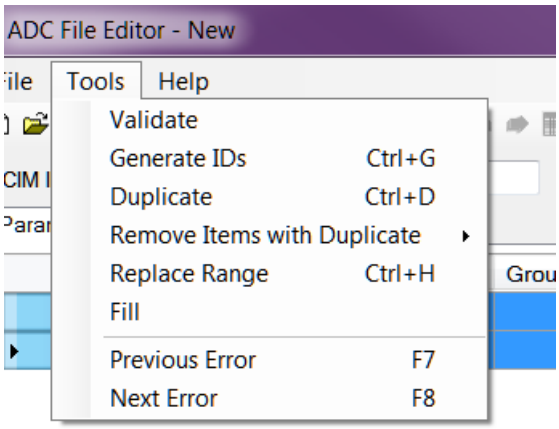
Select the Rows you want to duplicate. Note: to perform this function, you must select complete rows, individual cells are not sufficient.



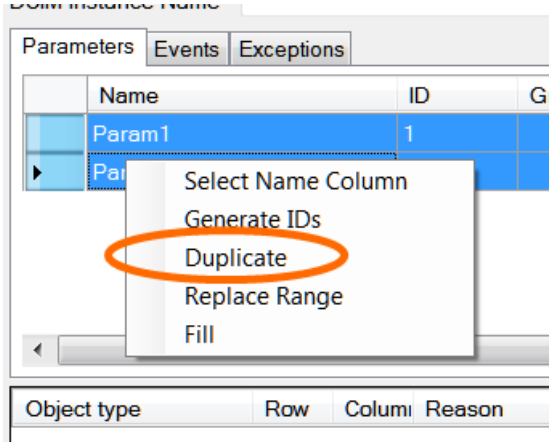
Press the Duplicate Button:



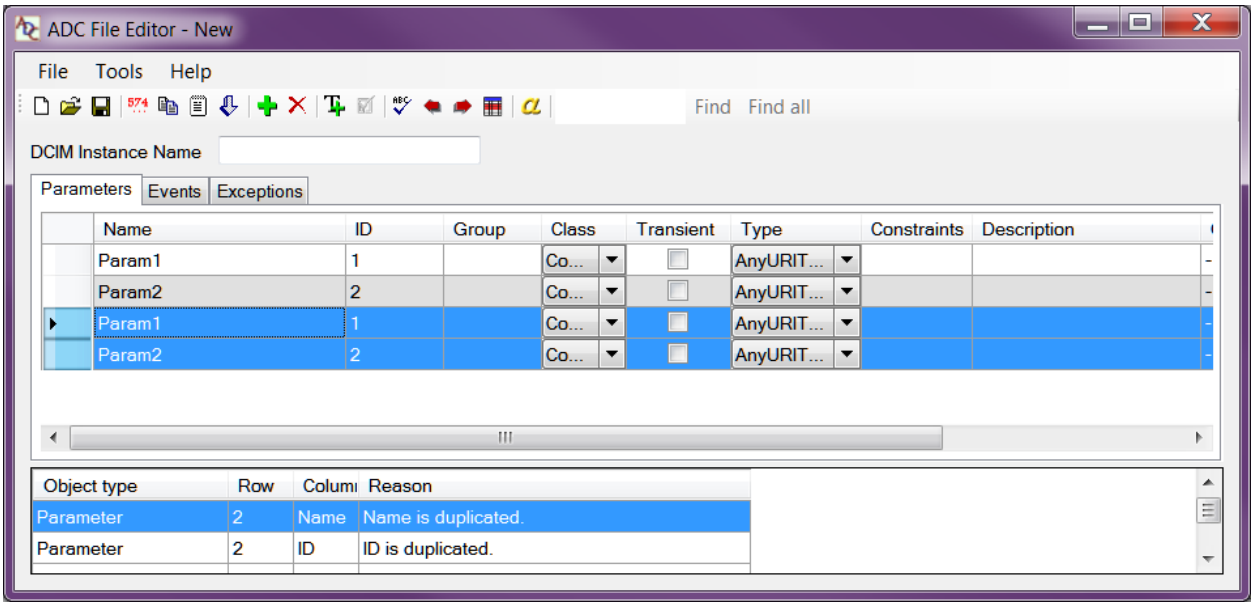
Or use the Tool menu option:



Or you can use Context Menu (right-click):



The selected rows will be added to the bottom and highlighted. Validation errors will be generated since the names and IDs are duplicated. Once these have been changed, the validation errors will be removed.

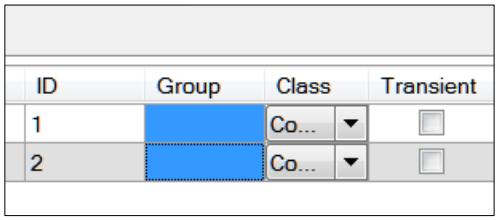


2.6.4 Fill Data

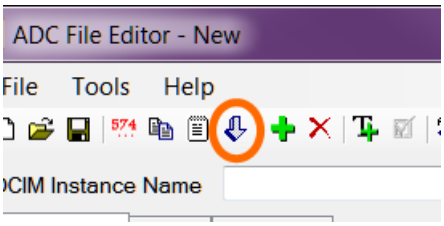
Fill table contents with a specific string, or preset value. There are different ways to fill data depending on the field type. For text fields, follow the generic fill instructions. Parameters have classes, transient and type fields that have constrained data with their own fill procedures.

2.6.4.1 Generic Fill

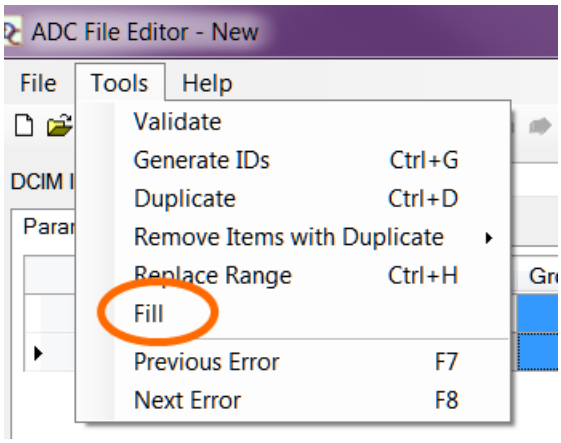
Select the cells (or rows) you want to fill.



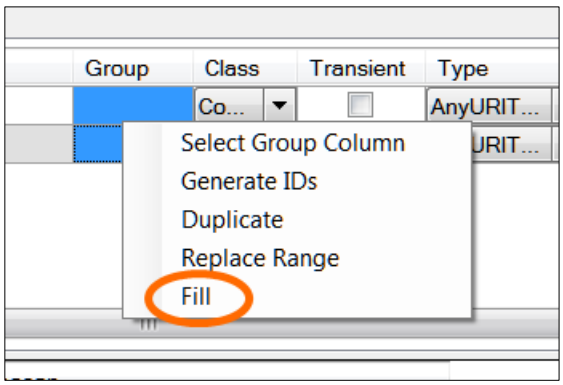
Press the Fill button on the toolbar:



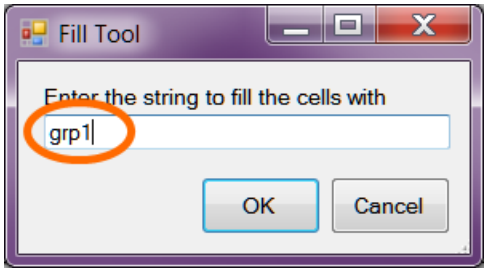
Or select Fill from the Tools menu:



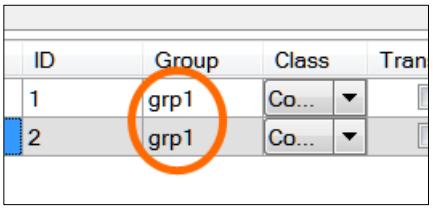
Or you can use Context Menu (right-click):



Input the desired string, and click OK.

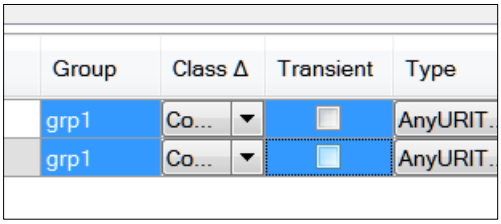


The cells will be filled with the input data.

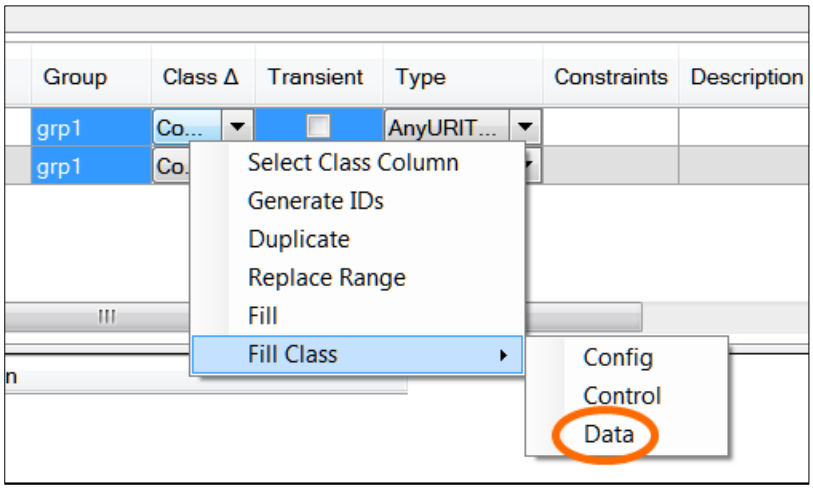


2.6.4.2 Fill Class

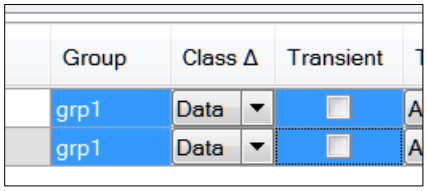
Select the Class cells that you wish to change



Use the Context Menu (right-click)

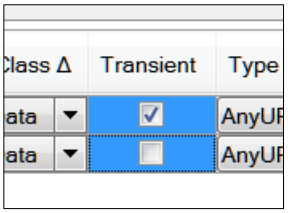


The Classes all have the new value

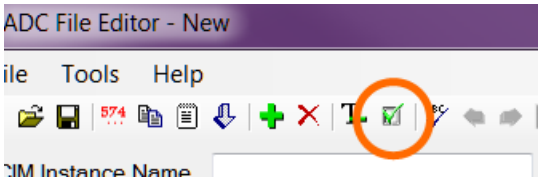


2.6.4.3 Toggle Transient

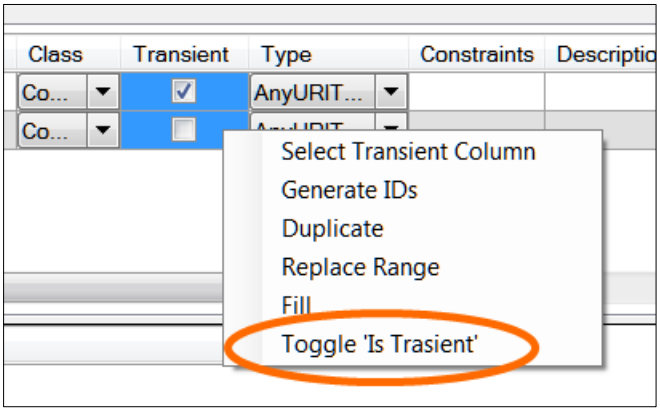
Select the Transient cells you want to change.



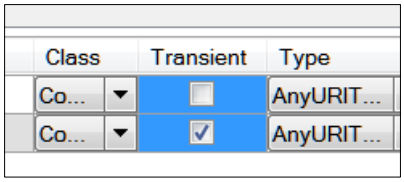
Use the button on the toolbar:



Or use the Context Menu (right-click):

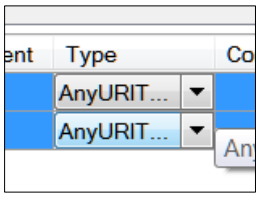


The values will have changed:

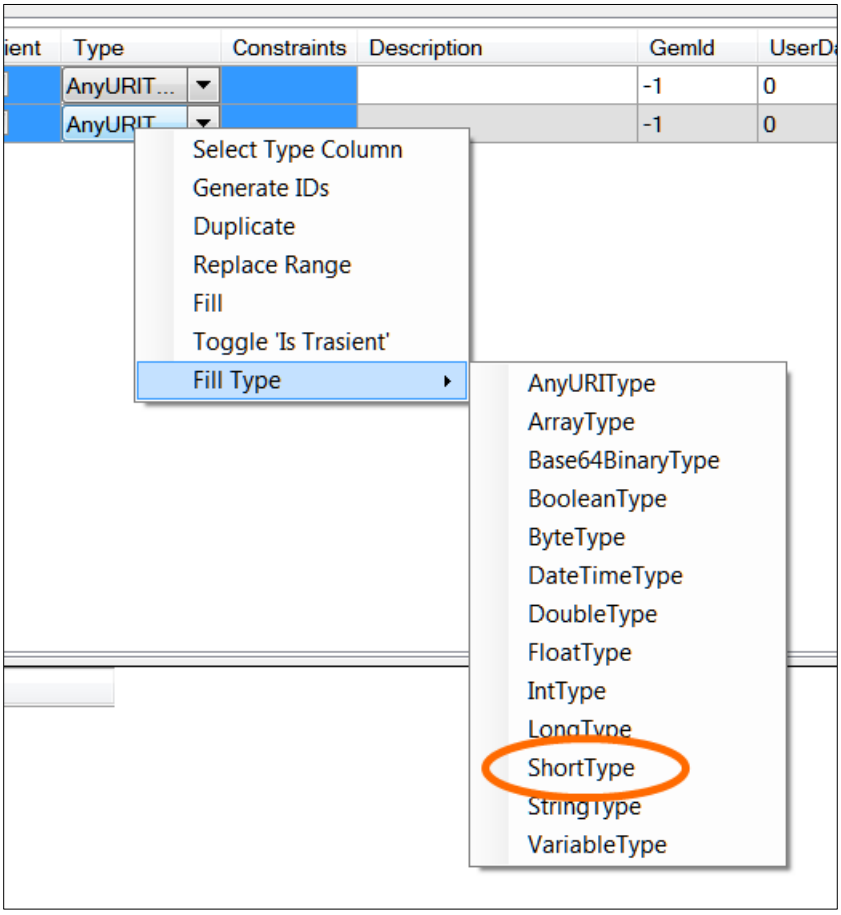


2.6.4.4 Fill Type

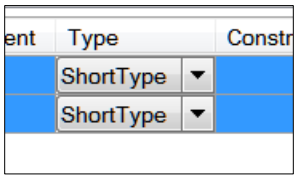
Select the Parameter Types you want to modify.



Use the Context Menu (right-click).

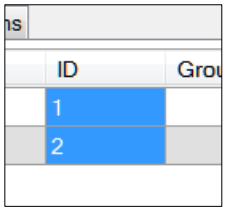


The values will have changed:

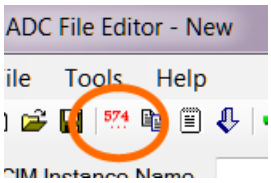


2.6.5 Generate IDs

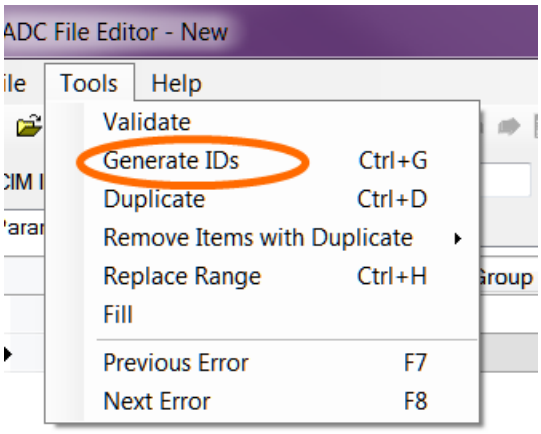
Select the ID cells you want to change. Alternatively, you could select the whole row. If you select no rows or any ID cells, you will generate new IDs for the entire set.



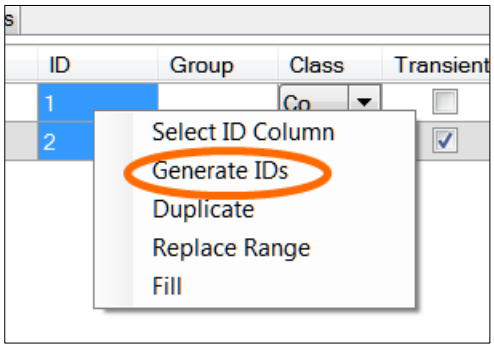
Press the Generate IDs button:



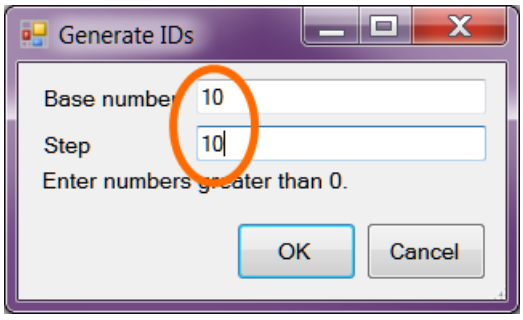
Or select Generate IDs from the Tools menu:



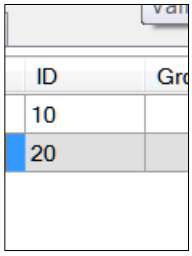
Or you can use Context Menu (right-click):



Input the starting number and step and click OK.

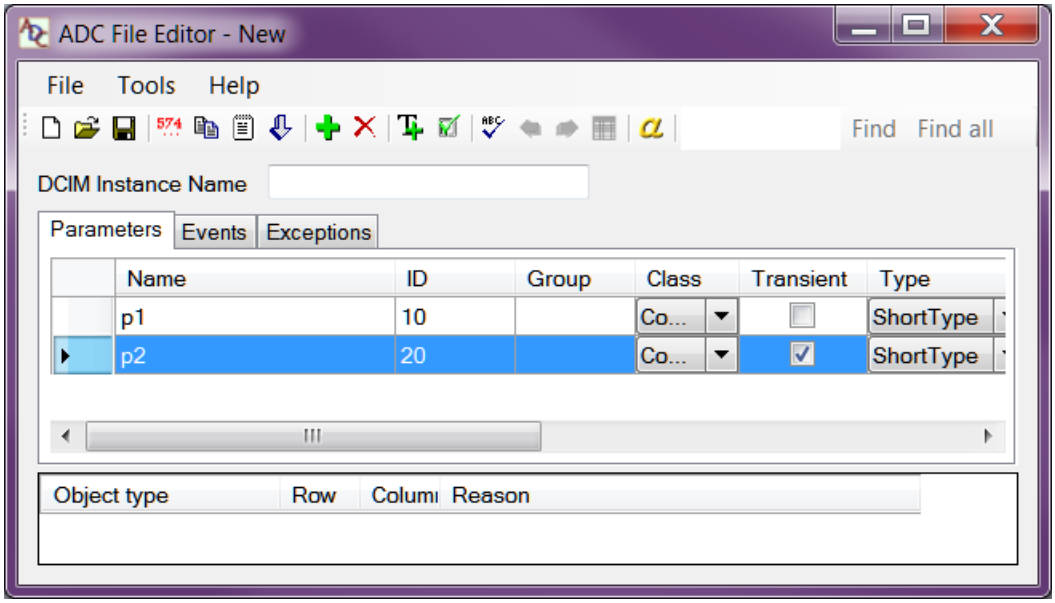


The IDs will be generated based the input data

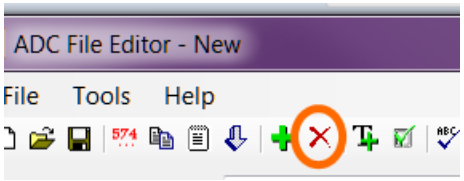


2.6.6 Remove Rows

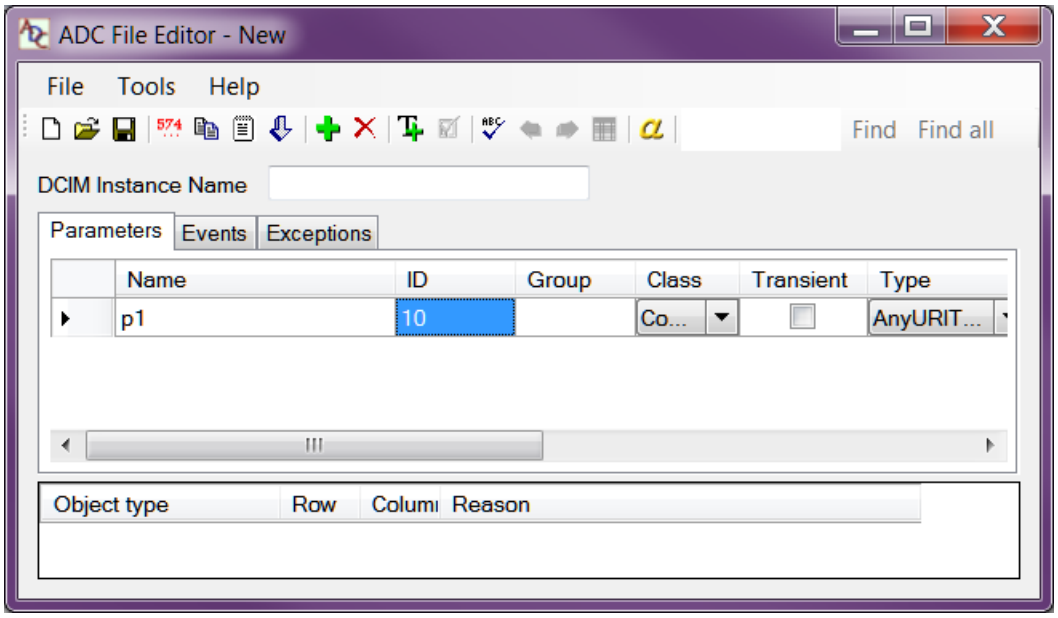
Select the rows you want to remove.



Press the Remove button, or simply press the Delete key.

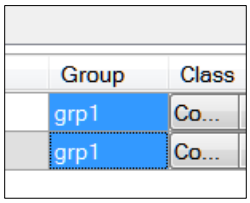


The rows have been removed.

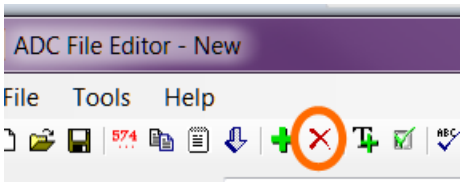


2.6.7 Remove Cell Data

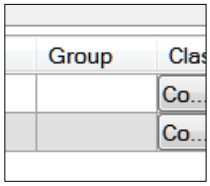
Select the cells containing the data you want to remove:



Press the Remove button.



The data in the cells has been removed.

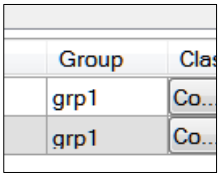


2.6.8 Import ADC file

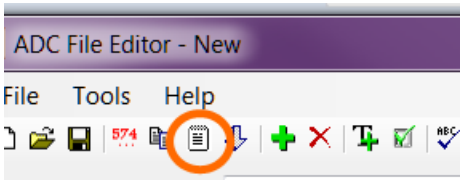
This will display a standard open file dialog to allow selecting an ADC file. The selected file will be merged with the currently open file. Any duplicate DCP objects (based on having the same name within the type) will be ignored during the import process.

2.6.9 Replace Range

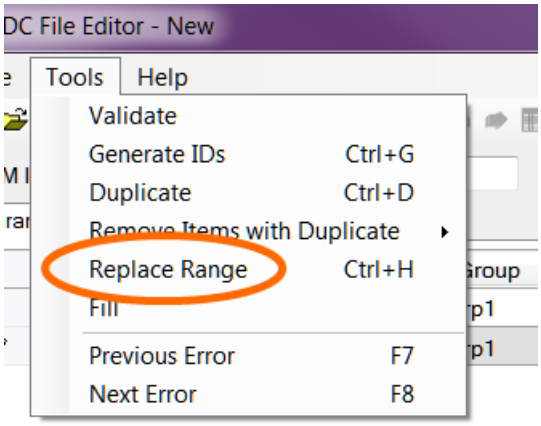
Select the cells you want to change.



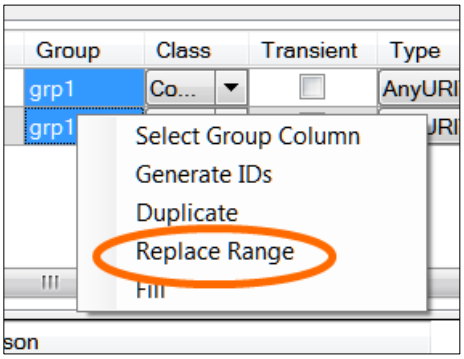
Press the Replace Range button:



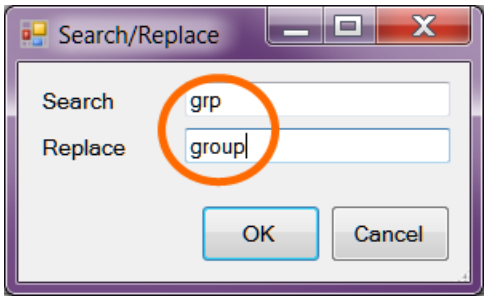
Or select Replace Range from the Tools menu:



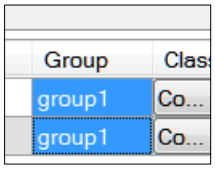
Or you can use Context Menu (right-click):



Input the strings to search for and replace and click OK.

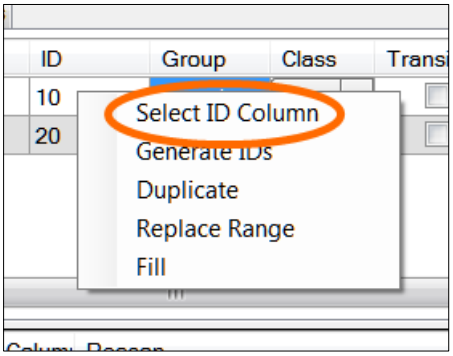


The fields will now reflect the changes you requested

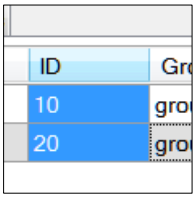


2.6.10 Select Column

Use the Context Menu (right-click) in any column

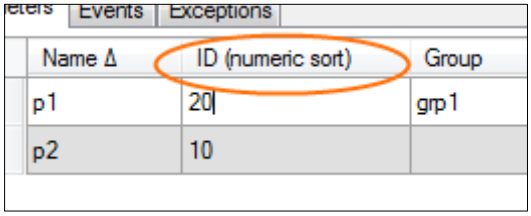


The requested column will be selected

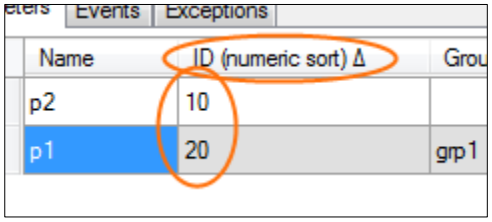


2.6.11 Sort by Column

Click the header of the column you want to sort.



The rows will be sorted based on the selected column in ascending order. And an indicator will show which column it was sorted by.

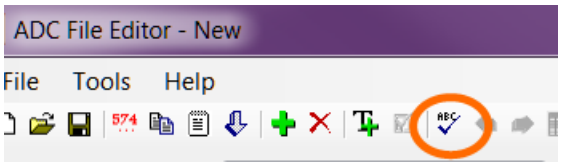


If all the contents of the ID column is numeric, it may be toggled between alpha and numeric sort orders by use of the tool bar button with an “a” on it.

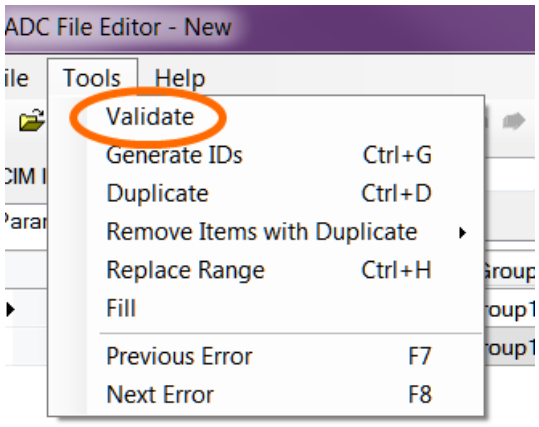
If there is any non-numeric content in the ID column, then a message will be given stating the sort order cannot be changed.

2.6.12 Validation

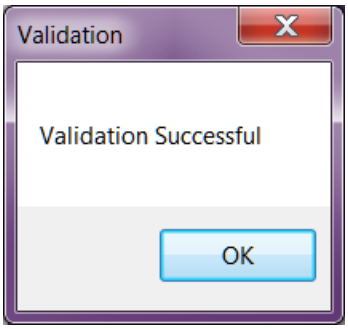
Generally, validation will happen on an as needed basis while editing the file. Any problems found will be displayed in the grid in the lower part of the window. Double clicking on the error will shift focus to the line containing the problem. Automatic validation will be turned off if the Find All feature is used since the results of the find will be placed in the same lower part of the window. Turn automatic validation back on, press the Validate Button:



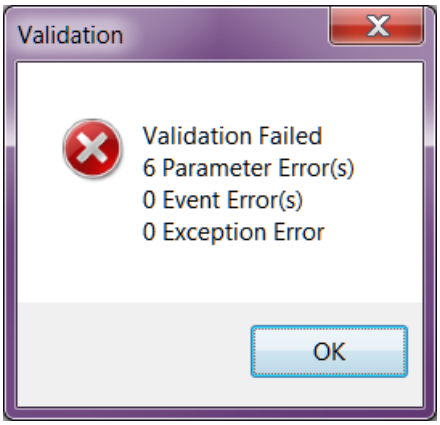
Or select Validate from the Tools menu:



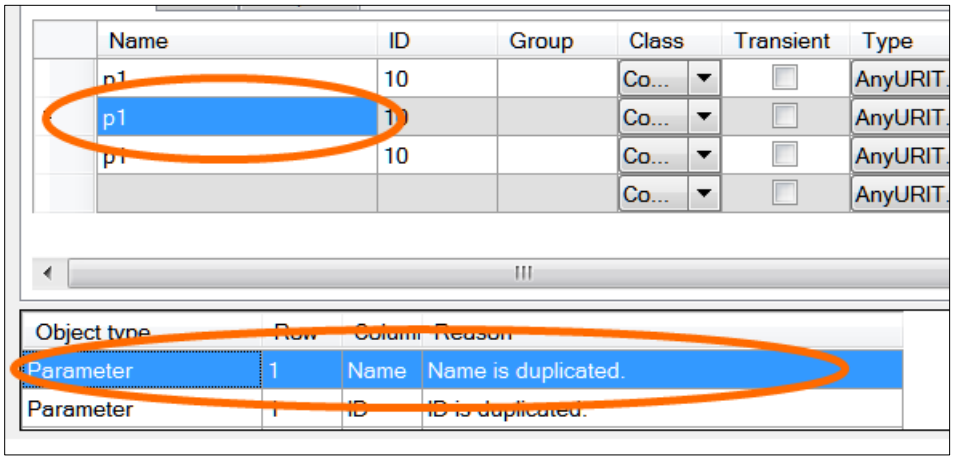
If there are no errors, you will see the following message:



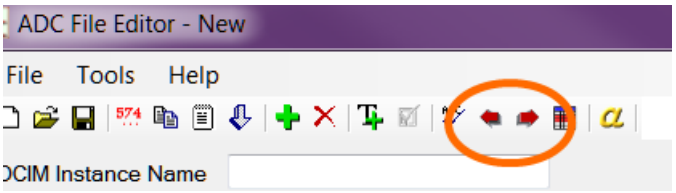
If there are errors, you will see the following message. Note that the number of errors will be displayed for each object type:



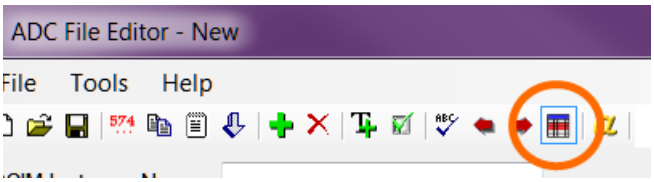
After pressing OK, the first error will be selected



You can now use the navigation buttons to go the Next or Previous error



Note the Select Errors button. As you navigate, it is the natural disposition of the component to unselect all rows except the one that it is currently on. If you want to show all errors all the time, click this button. It will reselect all errors as you navigate. But if this is a large set of data, it will impact performance.



2.6.13 Fixing Validation Errors

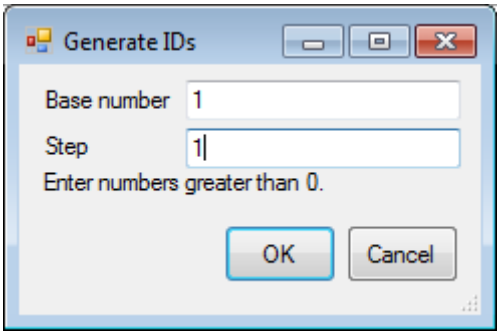
2.6.13.1 Fixing Name validation errors

Names must be assigned and cannot be duplicated within an object type. To fix either of these validation errors, double click on the error. The focus will be moved to the name field that does not conform. Change the value to a correct value.

2.6.13.2 Fixing ID validation errors

IDs must be assigned and cannot be duplicated within an object type. To fix either of these validation errors, double click on the error. The focus will be moved to the name field that does not conform. Change the value to a correct value.

Alternatively, new IDs can be generated automatically. To use this, select either the Generate IDs tool on the menu or press the corresponding toolbar button. The following dialog box will be displayed:



The Base number is the first number that will be used. The Step is how much to increment to the get the next ID number. Press OK to update the IDs. This will update all the IDs for the displayed object types, including ones already assigned. Press Cancel to not make any changes.

2.6.13.3 Fixing exception severity validation errors

Exceptions' Severity fields cannot be undefined; they must be set to a valid value. To fix these validation errors, double click on the error. The focus will be moved to the severity field that does not conform. Change the value to a correct value.

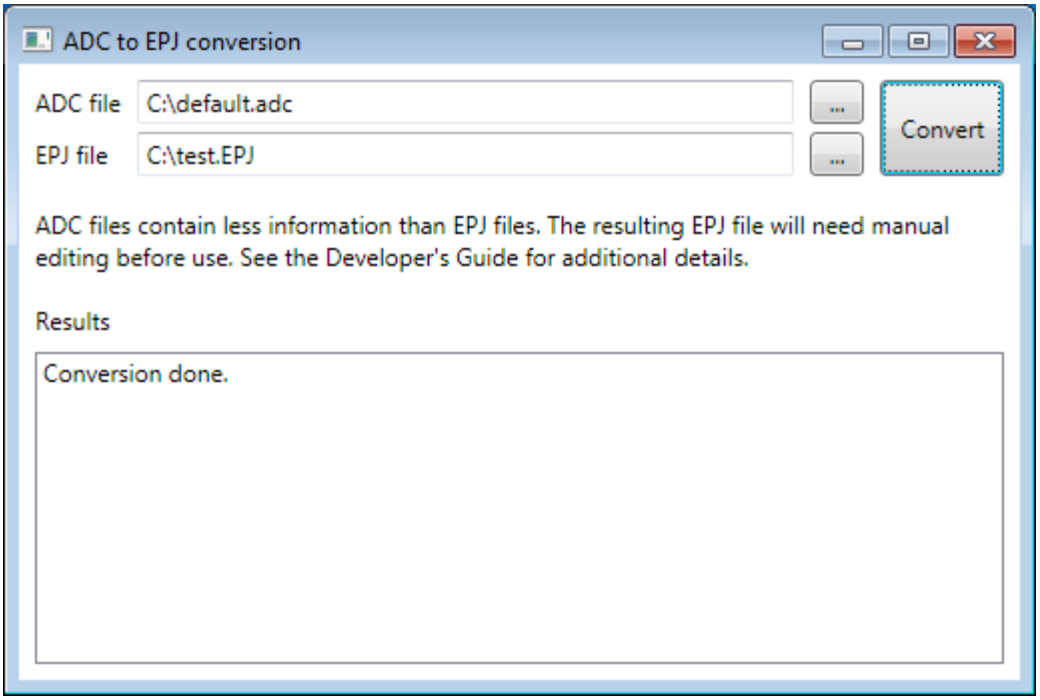
2.7 ADC file/EPJ file conversion

During development and trouble shooting, sometimes it is convenient to work with both CIMConnect DCIMs and AppDCIM-based DCIMs. To help facilitate this, there are conversion utilities to create an ADC file from an EPJ file and an EPJ file from an ADC file.

There is not a one-to-one relationship between these two formats with regard to their data contents. EPJ files contain more information than an ADC file, so generating an ADC file from an EPJ file will result in data loss and generating an EPJ file from an ADC file will result in many values containing default values.

2.7.1 ADC to EPJ file conversion

When the ADC to EPJ file conversion utility is selected, the following window will appear:



The Convert button will be disabled if the ADC file does not exist or the EPJ file is not specified.

The browse buttons, labeled with "…" can be used to select files for the two filename fields.

When the "Convert" button is pressed, the ADC file will be read and an EPJ file generated from it. If there are any problems, they will be detailed in the Results area. When completed, the "Conversion done" message will be shown.

The ADC file contains less information than the EPJ file contains. Because of this, the EPJ file will probably need to be manually edited before it can be used.

Specifically, the following items need to be addressed:

- All Events and Variables are put in Common. If any items contained in these two items need to be connection specific, then they should be moved from the common areas into new, connection specific sections.
- DCP exceptions' severity is more limited than EPJ alarms' AlarmCodes. They will be converted according to the following chart. Note the holes in the DCP Severity column. This indicates the associated EPJ Alarm code (codes 2, 5, 6 and 8) will never be assigned and some that are assigned may be assigned incorrectly (for types 1, 3, 4 and 7).

DCP Severity	Description	EPJ Alarm code
Undefined		0
Fatal	Personal Safety	1
	Equipment Safety	2

Warning	Parameter Control Warning	3
Error	Parameter Control Error	4
	Irrecoverable Error	5
	Equipment Status Warning	6
Informational	Attention	7
	Data Integrity Warning	8

- DCP parameters' parameter type is only loosly related to the EPJ variables' value type. They will be converted according to the following chart. Note the holes in the DCP Parameter Type column. This indicates the associated EPJ value types (types U1, U2, U4, U8, W, J) will never be assigned and some that are assigned may be assigned incorrectly (for types I1, I2, I4, I8 and A). Any parameter type that is not in the following table (e.g. enumerated types) will be assigned a value type valueAny.

DCP Parameter Type	EPJ Value Type
ArrayType	valueArray
VariableType	valueAny (default)
ByteType	I1
ShortType	I2
IntType	I4
LongType	I8
	U1
	U2
	U4
	U8
FloatType	F4
DoubleType	F8
Base64BinaryType	Bi
BooleanType	Bo

	W
StringType	A
	J
StructureType	L

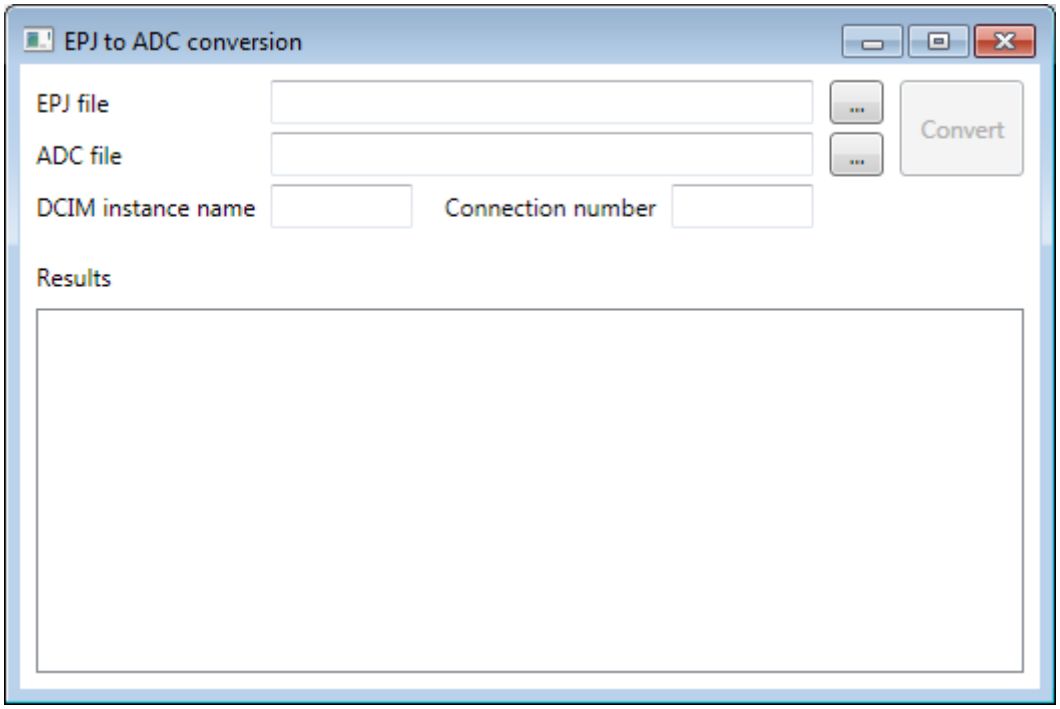
- DCP isTransient and parameter class fields are loosly related to the EPJ variable type. They will be converted according to the following chart. In general this should work, but a review might be advisable.

DCP isTransient	DCP parameterClass	EPJ varType
True	Data	DV
False	Data	SV
False	Config	EC

- The following EPJ variable fields are never set: default value, event ID, maximum value, minimum value, persist, private and units.
- The following EPJ event fields are never set: associated DVs.
- The well known name field is always set to be the same as the name field. These should be reviewd for accuracy.

2.7.2 EPJ to ADC file conversion

When the ADC to EPJ file conversion utility is selected, the following window will appear:



The Convert button will be disabled if the EPJ file does not exist or any of the ADC file, DCIM Instance name or Connection number are not specified.

The browse buttons, labeled with "…" can be used to select files for the two filename fields.

The DCIM Instance name is used to specify the name of the AppDCIM. It should be the same as the DCIM that uses the new ADC file.

The Connection number should refer to a connection section in the EPJ file that will be used to populate the ADC file. The ADC file will contain all the alarms, events and variables from common and this connection.

When the "Convert" button is pressed, the EPJ file will be read and an ADC file generated from it. If there are any problems, they will be detailed in the Results area. When completed, the "Conversion done" message will be shown.

2.8 CIMPortal Plus Service

This section describes different issues regarding administration of the CIMPortal Plus service:

- CIMPortal Plus Control Panel
- Configuration file
- Data collection
- Disabling model nodes
- Preserving configuration when uninstalling or upgrading
- SEMIObjType (E39) Data Collection

2.8.1 Data Collection

The purpose of the CIMPortal Plus Service is to gather data from one or more Data Collection Interface Modules (DCIMs) and make this data available to CIMWeb, CIMStore and other data collection applications. This section describes the different types of data collection that are possible in the CIMPortal Plus Service. Data is collected through the ICxCPEstablishedSession interface. Both CIMStore and CIMWeb use this interface. Trace, exception and event requests require the client using the interface to implement the ICxCPEquipmentDataCB interface.

Built-in data collection plans (DCPs) may be established by the equipment vendor to serve as default or common or recommended plans for use by their customers. This allows the customer to collect data without having to create their own plans. More information on managing these plans can be found in Section 2.11.

2.8.1.1 Trace Request

A Trace Request will report a set of data on a periodic basis. CIMPortal Plus uses the Microsoft waitable timers to implement trace data collection. Therefore the performance of a trace request is the closest multiple of 1/64 or 1/100, depending on the hardware. For more details and information, please search Microsoft MSDN for articles on ClockRes and High Resolution Timers.

Internally, CIMPortal Plus first identifies the DCIM instance for each parameter in the trace request. Then CIMPortal Plus asks each participating DCIM instance to report the data. Each DCIM instance is expected to setup its own autonomous timer and send the data in a timely manner before the TraceRequestTimeout timeout expires.

2.8.1.2 Event Request

CIMPortal Plus triggers events whenever the corresponding DCIM instance reports the event's occurrence. The DCIM that owns the event is also expected to send all of its requested data with the event report to CIMPortal Plus. If data is required from one or more additional DCIM, then CIMPortal Plus Server simultaneously requests the additional data from any corresponding DCIMs. Each DCIM must respond with data before the EventRequestTimeout timeout expires.

There is one exception to this rule. If the event is a part of a State Machine with a Location Parameter that corresponds to another DCIM (different than the event) then CIMPortal Plus will request the data from this DCIM first before requesting data from other DCIMs simultaneously.

2.8.1.3 Exception Request

CIMPortal Plus triggers exception state changes whenever corresponding DCIM instance reports a state change. The DCIM that owns the exception is also expected to send all of its requested data with the exception report to CIMPortal Plus. If data is required from one or more additional DCIM, then CIMPortal Plus Server simultaneously requests the additional data from any corresponding DCIMs. Each DCIM must respond with data before the ExceptionRequestTimeout timeout expires.

2.8.1.4 Data Requests

CIMPortal Plus will return a set of requested parameters at any time. If data is required from one or more additional DCIM, then CIMPortal Plus Server simultaneously requests the additional data from any corresponding DCIMs. Each DCIM must respond with data before the specified timeout value expires.

2.8.1.5 Data Collection Configuration

CIMPortal Plus's configuration file settings determine CIMPortal Plus tolerance for slow DCIM data reporting and for situations where the CIMPortal Plus cannot keep up with the requested data collection. If DCIMs do not send data fast enough, CIMPortal Plus will send empty values. If CIMPortal Plus cannot keep up with the requested data collection, then CIMPortal Plus will shut down the offending data collection request(s). The settings in the CIMPortal Plus Configuration File are critical to refining CIMPortal Plus's behavior.

2.8.2 Update Hardware

When CIMPortal Plus Service loads an equipment model file (either at startup or during deployment), it also tries to read a file named "Hardware.xml". If it finds the "Hardware.xml" file, it will update the model with the information found in this file.

2.8.2.1 Steps to complete:

1. Edit the "Hardware.xml" file so it contains the correct information. The file may be edited using a text editor, HTML editor, or an application using the Hardware Update Active X control.
2. Save the "Hardware.xml" file in the same directory as the deployed equipment model .encxml file (see Applying Updated Information to the Model about the location of the file). It must be named "Hardware.xml".
3. Restart CIMPortal Plus Service. The Hardware.xml file is loaded.
4. If "Stopped on CIMPortal Plus startup until Equipment Application started" option is set then Hardware.xml can be updated during startup sequence using the following steps:
    a. Connect to CIMPortal Plus service by creating an ICxCPService COM object.
    b. Establish a session with the equipment using the ICxCPService::EstablishSession method.
    c. Update "Hardware.xml" file
    d. After the model changes are done, the equipment should be enabled by calling the ICxCPEstablishedSession::Enable method at which point the updated "Hardware.xml" file is re-loaded.
    e. Close the session using the ICxCPEstablishedSession::CloseSession method.
5. If the log level is set to collect CIMPortal debug information, you should see in the new values in the log.

2.8.3 CIMPortal Plus Configuration File

CIMPortal Plus uses a configuration file called CIMPortal.cfg in the bin directory. It has the basically same format as a typical .ini file. The CIMPortal.cfg file has a main section designated by the heading `[CIMPortal]` and an `[Equipment]` section for each deployed equipment. Each `[Equipment]` section has a `<DCIM>` section for each DCIM instance in the equipment.

Changes made in the .cfg file do not take effect immediately. You must restart CIMPortal Plus in order to put the changes in effect. Create a backup of the CIMPortal.cfg file before making any changes. Syntax errors in this file could have an adverse effect on the CIMPortal Plus Service startup.

It is not required to have all settings in the file, if a setting is missing, the default value is used.

2.8.3.1 `[CIMPortal]` Settings

2.8.3.1.1 CfgVersion

The configuration version is used by Cimetrix to identify when the file format version to allow for backward compatibility in future versions of CIMPortal Plus.

2.8.3.1.2 BaseStorageDir

The CIMPortal Base Storage Directory is displayed and can be changed in the CIMPortal Plus Control Panel. It determines the base directory for all DCIM Packages, DCIM Instances, CIMStore configuration, templates, deployment packages and deployed equipment files and directories. Typically, this is not altered.

2.8.3.1.3 EquipmentDeployTimeout

When an equipment .pkg file is deployed in CIMPortal Plus, then CIMPortal Plus waits for the deployment to complete. This setting determines how long CIMPortal Plus will wait before assuming an error has occurred. The timeout value is in milliseconds. If not specified, the default value is 60000 ms.

2.8.3.1.4 MaxRequestQueueSize

The CIMPortal Plus Service has a data reporting queue. This setting determines the maximum number of trace reports allowed to be queued. This prevents DCIMs from sending trace reports faster than CIMPortal Plus can process them. If this value is exceeded, the offending data collection request is terminated and CIMPortal Plus issues a PerformanceWarning to the client application. The client, of course, can take further action like CIMWeb does. The default value is 250.

2.8.3.1.5 CustomGUI

This identifies a GUI ActiveX Control to display in the CIMPortal Plus Control Panel to configure CIMWeb. This should only be present when using CIMPortal Plus Interface A or Bundle, but not for CIMPortal Plus DB.

2.8.3.1.6 CPPriority

The CIMPortal Plus process priority specified as an integer (not a hexadecimal). Set this value from the CIMPortal Plus Control Panel. Typically it is desirable to increase the priority of the CIMPortal Plus service process to Above Normal or possibly High to ensure that trace data collection can run consistently with minimal interruption.

CIMPortal Plus Control Panel Setting	Value
Idle	64
Below Normal	16384

Normal	32
Above Normal	32768
High	128
Real-Time	256

See Microsoft MSDN documentation on function SetPriorityClass for a description of the values and for more information on how this setting affects the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.7 LoggingAndGuiThreadPriority

Thread priority of CIMPortal Plus's logging thread specified as a signed integer. Set this value from the CIMPortal Plus Control Panel. Typically this will have a lower priority such as Below Normal or Lowest.

CIMPortal Plus Control Panel Setting	Value
Idle	-15
Lowest	-2
Below Normal	-1
Normal	0
Above Normal	1
High	2
Real-Time	15

See Microsoft MSDN documentation on function SetThreadPriority for a description of the possible values and for more information on how this setting affects the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.8 DataCollectionThreadPriority

Thread priority of CIMPortal Plus's data collection thread(s) specified as a signed integer. Set this value from the CIMPortal Plus Control Panel. Typically this will have a higher priority such as High. See the table in LoggingAndGuiThreadPriority for possible values.

See Microsoft MSDN documentation on function SetThreadPriority for a description of the possible values and for more information on how this setting affects the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.9 DcimPingThreadPriority

Thread priority of CIMPortal Plus's ping DCIM thread(s). Typically this can be low priority. Set this value from the CIMPortal Plus Control Panel. Typically this will have a lower priority such as Lowest. See the table in LoggingAndGuiThreadPriority for possible values.

See Microsoft MSDN documentation on function SetThreadPriority for a description of the possible values and for more information on how this setting affects the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.10 PerfMonThreadPriority

Thread priority of CIMPortal Plus's performance monitor thread(s). Set this value from the CIMPortal Plus Control Panel. Typically this will have the default priority. See the table in LoggingAndGuiThreadPriority for possible values.

See Microsoft MSDN documentation on function SetThreadPriority for a description of the possible values and for more information on how this setting affects the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.11 TimerResolution

Resolution of the timers created, specified in milliseconds. This directly affects all timer-based features like trace data collection. If not specified, the default value is 1 ms. Microsoft's default timer resolution is 15.625 ms. This is a global Windows setting which affects other processes. See Microsoft MSDN documentation on function timeBeginPeriod for more details on how this affects the operating system, the CIMPortal Plus process and other processes on the same Windows system.

2.8.3.1.12 CustomCOMHandler

A COM object created by CIMPortal Plus Service at the end of the CIMPortal Plus Service startup. Multiple instances of the CustomCOMHandler setting can be specified. The CustomCOMHandler value must be a valid COM Prog ID registered on the Windows system where CIMPortal Plus is installed. CIMPortal Plus Service uses the CustomCOMHandler feature to startup the CIMWeb component, where the setting is specified like this "CustomCOMHandler=CIMWeb.CIMWebCreator".

2.8.3.1.13 ThreadPoolThreadCount

The number of threads allocated in a single thread pool used by all equipment and all App DCIM instances inside of the CIMPortal service process. The default value of this count is 0. Zero is the default value if the setting is not present. When the count is 0, CIMPortal calculates the number threads to allocate by the formula: 16 x the number of logical processor cores. For example, for a four-core processor system, the number of threads allocated for the pool would be 16 x 4 = 64 threads. When the count is nonzero, the value specified is used as the number of threads allocated for the pool, with a minimum of 32 and a maximum of 1024. The count value may need to be determined empirically on a particular Windows system according to available memory and performance given that approximately 1 MB of memory is allocated for each thread in a Windows process.

A condition under which the ThreadPoolThreadCount may be considered for optimization is one where an EDA client demands multiple trace reports at the same rate and triggered off of the same event, which would cause some trace data reports to be delayed in processing until threads in the pool became available. A characteristic of this condition would be a trace data report whose collection timestamp appears randomly delayed versus the expected trace timestamp.

2.8.3.2 `[Equipment]` Settings

2.8.3.2.1 Name

The equipment name. This can be viewed in the CIMPortal Plus Control Panel.

2.8.3.2.2 BaseDir

The equipment's base directory for storing all of its data collection setup, equipment model file and other files. This can be viewed and changed in the CIMPortal Plus Control Panel Configuration GUI. Typically it is not changed.

2.8.3.2.3 ModelFile

The name of the encrypted equipment model file. This can be viewed and changed in the CIMPortal Plus Control Panel. Typically it is not changed.

2.8.3.2.4 DeploymentDate

The deployment date. This can be viewed in the CIMPortal Plus Control Panel.

2.8.3.2.5 StartState

The initial state of the equipment. 1-enabled, 2-stopped until Eq. App starts it, 0-disabled. It is a good idea to have the StartState be 0 or 2 and to have a software application program notify CIMPortal Plus when the equipment is ready for data collection. The default is 1 so that data collection will be immediately available on CIMPortal Plus startup. This can be viewed and changed in the CIMPortal Plus Control Panel Configuration GUI.

2.8.3.2.6 EventRequestTimeout

When more than one DCIM must provide data with the Event Request, CIMPortal Plus requests the additional data from the other DCIM each time the event occurs. This timeout determines the maximum number of milliseconds allowed for the DCIM to respond with values. The default is 10000 milliseconds while the maximum is 60000 milliseconds. If DCIM do not respond before the timeout expires, then empty values are sent in the report and a warning is produced in the log.

2.8.3.2.7 ExceptionRequestTimeout

When more than one DCIM must provide data with the Exception Request, CIMPortal Plus requests the additional data from the other DCIM each time the exception state change occurs. This timeout determines the maximum number of milliseconds allowed for the DCIM to respond with values. The default is 10000 milliseconds while the maximum is 60000 milliseconds. If DCIM do not respond before the timeout expires, then empty values are sent in the report and a warning is produced in the log.

2.8.3.2.8 TraceRequestTimeout

When more than one DCIM must provide data with the Trace Request, each DCIM must send the data independently. This timeout determines the maximum number of milliseconds allowed for the DCIM to send its values. The default is 10000 milliseconds while the maximum is 60000 milliseconds. If DCIM do not send data before the timeout expires, then empty values are sent in the report and a warning is produced in the log.

2.8.3.2.9 MaxConcurrentReportsPerTrace

The maximum number of trace time periods within which each DCIM has to report the trace data. If a DCIM doesn't report the trace data within this allowed period, the CIMPortal Plus will send report to client application with Empty values and a warning will appear in the CIMPortal Plus logging.

2.8.3.3 <DCIM> Settings

2.8.3.3.1 Name

This is the DCIM Instance name. This can be viewed in the CIMPortal Plus Control Panel.

2.8.3.3.2 ProgID

This is the COM Prog ID for the DCIM Package. This can be viewed in the CIMPortal Plus Control Panel.

2.8.3.3.3 CfgDir

This is the DCIM Instance's configuration directory where the DCIM instance files are stored. Typically it is not changed.

2.8.3.3.4 ModelCfg

This is the DCIM modeling configuration string. This can be viewed and changed in the CIMPortal Plus Control Panel. Typically it is not changed.

2.8.3.3.5 RuntimeCfg

This is the DCIM runtime configuration string. This can be viewed and changed in the CIMPortal Plus Control Panel. Sometimes this is changed to configure the DCIM after deployment, such as timeouts for the CIMConnect DCIM.

2.8.3.3.6 Enabled

This is the default state of the DCIM. 1 = enabled (typical). 0 = disabled.

2.8.4 Preserving settings when CIMPortal Plus is uninstalled and reinstalled

When upgrading CIMPortal Plus, the preferred method is to install the new version without uninstalling the existing version of CIMPortal Plus. However, sometimes it is necessary to uninstall the existing copy of CIMPortal Plus first. When CIMPortal Plus is uninstalled, all built-in DCPs, configuration, and settings are lost. The first step in this section describes how to save the built-in DCPs, configuration and settings prior to uninstalling CIMPortal Plus. The second step in this section describes how to restore the built-in DCPs, configuration, and settings after the CIMPortal Plus upgrade is installed. Once these two steps are completed, CIMPortal Plus will operate with the same configuration and settings and the same built-in DCPs will be available as the original version of CIMPortal Plus.

2.8.4.1 Save the Configuration and Settings Prior to Uninstalling CIMPortal Plus

Prior to uninstalling CIMPortal Plus, the built-in DCPs, configuration, and settings must be saved or else they will be deleted. The following steps will save the built-in DCPs, configuration, and settings:

1. Copy these files

To save the configuration and settings, copy these files from the Cimetrix bin folder (typically \\Program Files\Cimetrix\Comm Products\bin) to another location:

- CIMPortal.cfg - contains CIMPortal Plus settings and information about the deployed equipment and DCIMs
- CMXConfig.xml – contains logging settings and privilege information
- CMXUsers.cfg – contains access control information

2. Copy any built-in DCPs

The built-in DCPs are stored as XML files in the equipment folders. Typically the files are located in "\\Program Files\Cimetrix\Comm Products\bin\Storage\[Equipment Name]\Runtime\Plans". All built-in plan file names will start with "urn_semi-org_equipment". Locate all files with this name structure and copy them to another location.

3. Restore the configuration and settings after upgrading CIMPortal Plus

After installing the CIMPortal Plus upgrade, the built-in DCPs, configuration, and settings can be restored. The following steps will restore the built-in DCPs, configuration, and settings:

4. Restore the saved files

To restore the configuration and settings, copy these files from the saved location to the Cimetrix bin folder (typically \\Program Files\Cimetrix\Comm Products\bin):

- CIMPortal.cfg
- CMXConfig.xml

- CMXUsers.cfg

If there are new features in CIMPortal Plus that are handled in the CIMPortal.cfg or CMXConfig.xml files, these new features will not be configured when the original file is restored. When the CIMPortal.cfg or CMXConfig.xml files do not contain entries used by new features, default values for those features are used. Manually merging the new features into the original files will allow the new features to be configured.

5. Restore any built-in DCPs

After installing CIMPortal Plus and deploying equipment, restore any built-in DCPs by copying the saved xml files to the equipment folders (typically "\\Program Files\Cimetrix\Comm Products\bin\Storage\[Equipment Name]\Runtime\Plans"). If the Runtime or Plans folder does not exist, create it.

6. Restart Services

For the changed files to be restored, the following services need to be restarted:

- CxConfigSvc
- CxCIMPortalSvc
- CxCIMStoreSvc (if installed)

2.8.5 SEMIObjType (E39) Data Collection

Interface A uses SEMIObjType data to make E39 object data available. Standard SEMIObjTypes include E40 process jobs, E87 carriers, E90 substrates and substrate locations, E94 control jobs and E116 Equipment Performance Tracking (EPT) modules. Equipment suppliers can also implement custom object types. This section describes how to implement SEMIObjType data collection in CIMPortal Plus. This includes how to define the SEMIObjType in the equipment model, how to define various instances in the equipment model and how to provide data using different DCIMs.

The basic steps to implement SEMIObjType objects are:

1. Create SEMIObject Definitions in the Equipment Model
2. Declare SEMIObject Instances in the Equipment Model
3. Implement SEMIObject Instance Identifiers (Interface A GetObjTypeInstanceIds)
4. Implement Object Attribute Data Collection

2.8.5.1 Create SEMIObject Definitions in the Equipment Model

You must first define SEMIObject types in the equipment model. Follow these steps to create each SEMIObject definition. As an alternative to creating the SEMIObject definitions manually, Equipment Model Developer has templates for adding them to the Cimetrix implementations of E40, E87, E90, E94, E99, and E116 standards and related state machines.

1. Drag and drop a SEMIObject into the equipment model. Typically they are defined at the equipment root level. Enter the ObjectType appropriate, such as "Substrate" for an E90 substrate and provide a Description.
2. Assign the DCIM that is responsible for the SEMIObject data collection. The assigned DCIM will be responsible to provide the list of all object identifiers for this SEMIObjectType and the values for each object's attribute. By default, Equipment Model Developer adds the E39 mandatory ObjID and ObjType attributes.
3. Drag and drop an E39Attribute for each attribute required in the definition. Assign accurate names to match the E39 object implementation and assign an appropriate parameter type.
4. Drag and drop all application parameters as object EventData. For example, GEM Data Variables related to the SEMIObjType are typical EventData.
5. Select an InstParameter for the SEMIObjType, a parameter that contains the instance information for the SEMIObject type. For example, typically for E87 Carrier objects there is a parameter "CarrierID" whose value is an object's name when related events occur.

6. Drag and drop any State Machine Instances applicable to the static SEMIObjType. The State Machines must be previously defined. As an example, the E90 Substrate object implements the Substrate object state model and therefore should have an instance of this state machine.

2.8.5.2 Declare SEMIObject Instances in the Equipment Model

There are two general types of SEMIObjType (or E39) objects that could be implemented: static and dynamic.

- Static objects are always present including internal (non carrier) E90 substrate locations and E116 EPT modules. For each static object, drag and drop a SEMIObjectInstance onto the related E120 hardware element. Specify the ObjectID, the actual object identifier. In Interface A, this ObjectID will be included in the object identifier list for operation GetObjTypeInstanceIds. Note that a DCIM still has to provide this ObjectID to the CIMPortal Plus.
- Dynamic objects are not always present including E87 carriers, E90 substrates, E90 carrier substrate locations, E40 process jobs and E94 control jobs. For the dynamic objects no SEMIObjectInstance nodes should be added the equipment. The dynamic objects always reported to the EDA Clients with SourceID of the root equipment node.

2.8.5.3 Implement SEMIObject Instance Identifiers (Interface A GetObjTypeInstanceIds)

The DCIM instance assigned to handle SEMIObjType data collection must provide CIMPortal Plus with the list of object instances IDs as requested. For example, if there are three process jobs in the equipment, the assigned DCIM should return three process job object instance IDs. The list of dynamic objects is expected to change during production. In the CIMPortal Plus help file, see the DCIMs section of documentation for more details on the developer's responsibilities for each particular DCIM.

2.8.5.4 Implement Object Attribute Data Collection

The DCIM instance assigned to handle SEMIObjType data collection must provide CIMPortal Plus the values of object attributes as requested. Of course, because of the dynamic nature of most objects, it is possible for clients to request the attribute values of objects that have expired and the DCIM returns an empty value. In the CIMPortal Plus help file, see the DCIMs section of documentation for more details on the developer's responsibilities for each particular DCIM

2.8.6 Disabling Model Nodes

Programmatically removing nodes from a deployed equipment model is possible. Removing nodes is useful when an OEM wants to deploy a single equipment model and then disable the model nodes which don't apply to a particular machine configuration. Disabling model nodes removes those nodes from the volatile model being used by the CIMPortal Plus service. It does not remove the nodes from the non-volatile equipment model stored on disk. Consequently, the nodes must be disabled each time CIMPortal Plus service is started.

Disabling model nodes must be done after the equipment model is created and before the equipment is enabled. The default behavior is to enable equipment when it is created by CIMPortal Plus service. This behavior can be changed after the equipment is deployed with the Equipment status and configuration dialog. Check both the "Enabled on CIMPortal startup" checkbox and the "Stopped on CIMPortal startup…" check box. With these settings, the equipment will remain in the "Initializing" state when CIMPortal Plus service is started. It will not be available for data collection until the equipment is enabled.

During subsequent startups, an application can attach to CIMPortal Plus service and modify the equipment model using the following steps:

1. Connect to CIMPortal Plus service by creating an ICxCPService COM object.
2. Establish a session with the equipment using the ICxCPService::EstablishSession method.

3. Disable a model node using the ICxCPEstablishedSession::DisableModelNode method. This can be done as many times as needed for different nodes. The CIMPortal Plus service client must have the Modify Model privilege to call the DisableModelNode method.
4. After the model changes are done, if the resulting model is different than the last time the application was run, then call ICxCPEstablishedSession::ChangeModelRevisionTimestamp to set the new revision timestamp.
5. After the model changes are done, the equipment can be enabled by calling the ICxCPEstablishedSession::Enable method.
6. Close the session using the ICxCPEstablishedSession::CloseSession method.

2.9 CIMWeb

2.9.1 CIMWeb Runtime Performance Monitoring

During runtime, CIMWeb is monitoring performance in three ways:

- Periodic Monitoring: The CIMWeb administration GUI has been configured for periodic monitoring. Periodic monitoring allows the automatic deactivation of DCPs.
- On Demand: Performance messages are triggered by the equipment DCIMs and sent immediately to an Interface A client. Automatic deactivation of DCPs is not done.
- Report queue size monitoring: CIMWeb is continuously monitoring current report queue size for each EDA session and if any session's queue grows above the configured maximum, the equipment is put in "Performance Warning" state – a PerformanceWarning message is sent out to all sessions and all DCPs are deactivated. The maximum report queue size is configured in the Control Panel (see section 2.12.4).

If the CIMWeb Performance Monitoring tab in the Control Panel (see section 2.2.5.3) has been used to specify a 'Check performance every X seconds' value greater than zero, then **Periodic Monitoring** is taking place.

When periodic monitoring is in process, PerformanceWarning and PerformanceRestored evaluations are done every N seconds (as specified by the configurable value in the Control Panel). Based on the performance monitoring tests in use, PerformanceWarning and PerformanceRestored messages are sent to an Interface A client only during the evaluation of the test(s). This means that if a CPU monitoring test has been configured to be performed and the CPU goes 'High' and then back to 'Normal' between tests, the periodic monitoring will not recognize that the CPU was high. In other words, the CPU must be 'High' as the monitoring test is evaluated.

The same is true of DCIM-generated Performance warnings. If a DCIM triggers a PerformanceWarning, the PerformanceWarning message is sent to CIMWeb and stored. This PerformanceWarning state is only realized at the occurrence of next periodic monitoring cycle. Hence, an Interface A client would be sent the PerformanceWarning message as part of the periodic evaluation cycle.

Word of caution: If the DCIM triggers a performance warning and then a performance restored message between the evaluation of a monitoring cycle, then CIMWeb will never see the performance warning state from the DCIM because the PerformanceRestored message will clear the PerformanceWarning message before CIMWeb has a chance to evaluate it and send a message to a client.

On-Demand monitoring is used when the Administrator has configured a value of zero on the Performance Monitoring administration tab. This means that PerformanceWarning and PerformanceRestored messages are initiated only by the Equipment DCIM(s). One major difference between these two types of monitoring is that in On-Demand monitoring, the PerformanceWarning and PerformanceRestored messages are sent to Interface A clients *immediately - as opposed to waiting for a Performance Monitoring cycle to occur.*

Note that if equipment has multiple DCIMs, when one DCIM is in the PerformanceWarning state, the entire

equipment is considered to be in that state and the PerformanceWarning message is sent to ALL clients of that equipment. In order *to return to the PerformanceRestored state, all of the DCIMs must return to the PerformanceRestored state before the interface A clients will be sent the PerformanceRestored message.*

Word of caution: One drawback of On-Demand monitoring is that no automatic deactivation of DCPs is done. The Administrator must configure and use Periodic Monitoring if automatic DCP deactivation is needed.

2.9.2 Conversion from Cimetrix values to EDA

Cimetrix CIMWeb uses the parameter's PTD as it is defined in the model when converting values to EDA. The conversion rules are described in the following table with a few exceptions:

- When CxValue set to uninitialized (empty) value EDA value is always converted to NoValueType with reasonCode = ValueNotAvailable and Description = "Value Not Available"
- When CxValue is set to a type not compatible with PTD EDA value is set to NoValueType with reasonCode = ValueNotAvailable and Description = "Invalid parameter type: cannot convert from CxValue type 'X' to parameter type Y", where X is the CxValue type and Y is EDA type
- When converting numeric values and CxValue is set to a valid type but its value is outside of range of PTD (for example PTD is ByteType and CxValue is set to I4 300) EDA value is set to NoValueType with reasonCode = ValueNotAvailable and Description = "Parameter value overflow"
- Variable PTD rules are described in the "Cimetrix Variable PTD Conversion" section

PTD	Valid CxValue type(s)	1105 EDA Value	0710 EDA Value
ByteType	I1,I2,I4,I8,U1,U2,U4,U8 with value (-128..127)	"I1" "xsd:byte"	"I1" "xsd:byte"
	Empty I1,I2,I4,I8,U1,U2,U4,U8	"Null"	"I1"
ShortType	I1,I2,I4,I8,U1,U2,U4,U8 with value (-32,768..32,767)	"I2" "xsd:short"	"I2" "xsd:short"
	Empty I1,I2,I4,I8,U1,U2,U4,U8	"Null"	"I2"
IntType	I1,I2,I4,I8,U1,U2,U4,U8 with value (-2,147,483,648..2,147,483,647)	"I4" "xsd:int"	"I4" "xsd:int"
	Empty I1,I2,I4,I8,U1,U2,U4,U8	"Null"	"I4"
LongType	I1,I2,I4,I8,U1,U2,U4,U8 with value (-9,223,372,036,854,775,808.. 9,223,372,036,854,775,807)	"I8" "xsd:long"	"I8" "xsd:long"
	Empty	"Null"	"I8"

	I1,I2,I4,I8,U1,U2,U4,U8		
FloatType	F4,F8 with value (-/+3.402823466E+38)	“F4” "xsd:float"	“F4” "xsd:float"
	Empty F4,F8	“Null”	“F4”
DoubleType	F4,F8 with value (-/+1.7976931348623158E+308)	“F8” "xsd:double"	“F8” "xsd:double"
	Empty F4,F8	“Null”	“F8”
BooleanType	Bo	“B” "xsd:bool"	“B” "xsd:bool"
	Empty Bo	“Null”	“B”
Base64BinaryType	Bi, single or array	“B64” "xsd: base64Binary"	“B64” "xsd: base64Binary"
	Empty Bi	“Null”	“B64”
DateTimeType	I8,U8 (use Win32 GetSystemTimeAsFileTime or .NET DateTime.Now. ToFileTimeUtc to get current time)	“D” "xsd: dateTime"	“D” "xsd: dateTime "
	Empty Bo	“Null”	“D”
AnyURIType	A, W	“URI” "xsd:string"	“URI” "xsd:string"
StringType	A, W	“S” "xsd:string"	“S” "xsd:string"
Enumeration (Integer)	I1,I2,I4,I8,U1,U2,U4,U8 with value (-2,147,483,648..2,147,483,647). CIMWeb doesn’t check the value against possible enumeration values in the model.	“EI” "xsd:int"	“EI” "xsd:int"
Enumeration (String)	A, W CIMWeb doesn’t check the value against possible enumeration values in the model.	“ES” "xsd:string"	“ES” "xsd:string"
ArrayType	Empty, single or array value of a valid (see corresponding PTD above)	"Arr" CompositeValueType with corresponding	"Arr" CompositeValueType with corresponding

		value and xsd type	value and xsd type
Structure	Empty L	“Su” xsi:nil="true"	“Su” xsi:nil="true"
	L with same number of items as fields in PTD	“Su” with elements for each field, which are converted using their PTD	“Su” with elements for each field, which are converted using their PTD

2.9.2.1 Cimetrix Variable PTD Conversion

Cimetrix CIMWeb follows the following rules when converting parameter values from Cimetrix CxValue to EDA value.

CxValue	1105 EDA Value	0710 EDA Value
Empty I1	“Null”	“I1”
Empty I2	“Null”	“I2”
Empty I4	“Null”	“I4”
Empty I8	“Null”	“I8”
Empty U1	“Null”	“I2”
Empty U2	“Null”	“I4”
Empty U4	“Null”	“I8”
Empty U8	“Null”	“I8”
Single I1	“I1” "xsd:byte"	“I1” "xsd:byte"
Single I2	“I2” "xsd:short"	“I2” "xsd:short"
Single I4	“I4” "xsd:int"	“I4” "xsd:int"
Single I8	“I8” "xsd:long"	“I8” "xsd:long"
Single U1	“I2” "xsd:short"	“I2” "xsd:short"
Single U2	“I4” "xsd:int"	“I4” "xsd:int"
Single U4	“I8” "xsd:long"	“I8” "xsd:long"
Single U8	“I8” "xsd:long"	“I8” "xsd:long"

I1 Array	"Arr" CompositeValueType with “I1” array of "xsd:byte"	"Arr" ArrayValueType with “I1” array of "xsd:byte"
I2 Array	"Arr" CompositeValueType with “I2” array of "xsd:short"	"Arr" ArrayValueType with “I2” array of "xsd:short"
I4 Array	"Arr" CompositeValueType with “I4” array of "xsd:int"	"Arr" ArrayValueType with “I4” array of "xsd:int"
I8 Array	"Arr" CompositeValueType with “I8” array of "xsd:long"	"Arr" ArrayValueType with “I8” array of "xsd:long"
U1 Array	"Arr" CompositeValueType with “I2” array of "xsd:short"	"Arr" ArrayValueType with “I2” array of "xsd:short"
U2 Array	"Arr" CompositeValueType with “I4” array of "xsd:int"	"Arr" ArrayValueType with “I4” array of "xsd:int"
U4 Array	"Arr" CompositeValueType with “I8” array of "xsd:long"	"Arr" ArrayValueType with “I8” array of "xsd:long"
U8 Array	"Arr" CompositeValueType with “I8” array of "xsd:long"	"Arr" ArrayValueType with “I8” array of "xsd:long"
Empty Bo	“Null”	“B”
Single Bo	“B” "xsd:bool"	“B” "xsd:bool"
Bo Array	"Arr" CompositeValueType with “B” array of "xsd:bool"	"Arr" ArrayValueType with “B” array of "xsd:bool"
Empty Bi	“Null”	“B64”
Single Bi	“B64” "xsd: base64Binary"	“B64” "xsd: base64Binary"
Bi Array	“B64” "xsd: base64Binary"	“B64” "xsd: base64Binary"
A,W	“S” "xsd:string"	“S” "xsd:string"
Empty F4	“Null”	“F4”
Single F4	“F4” "xsd:float"	“F4” "xsd:float"
F4 Array	"Arr" CompositeValueType with “F4” array of "xsd:float"	"Arr" ArrayValueType with “F4” array of "xsd:float"
Empty F8	“Null”	“F8”

Single F8	“F8” "xsd: double"	“F8” "xsd: double"
F8 Array	"Arr" CompositeValueType with “F8” array of "xsd:double"	"Arr" ArrayValueType with “F8” array of "xsd:double"
L	"Var" "CompositeValueType"	"Su" "StructureValueType"

NOTE: When converting U8 values, values greater than 9223372036854775807 (the largest signed integer value) are converted to “Null” value for 1105, and “NoValue” NoValueType with reasonCode = ValueNotAvailable and Description = "Value overflow".

2.9.3 State Machine and State Machine Instance Event names and IDs

To comply with E164, when CIMWeb is converting Cimetrix Model into EDA model to be sent to EDA Clients it uses the following rules when generating State Machine (SM) and State Machine Instance (SMI) Event names and IDs (these rules do not apply to Simple Events):

1. If a SM event in Cimetrix Model is referenced by multiple transitions (like in E116 EPT State Model)
    a. EDA Event Name = <EventName>
    b. EDA Event ID = <SMName>:<EventName>
2. Otherwise, if an event is referenced by a single transition in SM:
    a. EDA Event Name = <SourceState>-<TargetState>
    b. EDA Event ID = <SMName>:<tr#>:<SourceState>-<TargetState>

where

- <SMName> is “SMName” property of the State Machine
- <EventName> is “EventName” property of the event
- <tr#> is “ID” property of the transition referencing the event with letter modifier, if exists, removed
- <SourceState> is “SourceState” property of the transition
- <TargetState > is “TargetState” property of the transition

2.9.4 CIMWeb administration API

The way to access the CIMWeb API has changed significantly in CIMPortal Plus from prior versions. For details, please see the *CIMWeb Admin API Help*.

The ICIMWebAdmin interface directly contains all the global configuration properties for CIMWeb. It also contains the GetEquipmentConfigurations method which returns an array of type IEquipmentConfigurationAdmin. Each item in this array contains configuration information specific to an equipment/version combination. There are also properties to retrieve administration interfaces for performance monitoring (PerformanceTestAdmin) and session administration (SessionAdmin).

2.9.5 CIMWeb Error Codes

The following tables outline the possible error codes that may be returned for a given web service.

2.9.5.1 Error code ranges

The XML 3570 SEMI spec defines error codes as follows:

0001 – 0999 Backward Compatibility to Current SEMI Standards

5000 - 5999 Reserved -> Common Standardized Errors

6000 - 6999 Reserved -> SEMI E132 Errors

7000 - 7999 Reserved -> SEMI E125 Errors

8000 - 8999 Reserved -> SEMI E134 Errors

9000 - 9999 Reserved -> SEMI Exxx Errors

10000 - 10500 Equipment Specific Error codes

2.9.5.2 E125

2.9.5.2.1 GetTypeDefinitions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.2 GetEquipmentStructure

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.3 GetEquipmentNodeDescriptions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header

6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.4 GetUnits

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.5 GetStateMachines

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.6 GetSEMIObjTypes

Code	Message

5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.7 GetExceptions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.8 GetLatestRevision

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.2.9 NotifyOnRevisions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
5002	Null NotifyOnRevisionsRequest
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.3 E132

2.9.5.3.1 EstablishSession

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
5002	The consumer url needs to be populated on a establish session request
6000	ACL entry < subjectId> not found.
6006	Session limit exceeded error
10002	Client already has a session
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10010	The session id cannot be provided for establish session.
10016	Invalid CIMPortal Session

2.9.5.3.2 CloseSession

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>

6000	CloseSession Operation Error: Cannot close another user's session without the Security Admin privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.3.3 SessionPing

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.3.4 PersistSession 1105 only

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6005	Unrecognized session <session id>
10000	The persist request did not specify whether the session should be persisted
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.3.5 EnhancedEstablishSession 0710 only

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
5002	The consumer url needs to be populated on a establish session request
6000	ACL entry < subject> not found.
6006	Session limit exceeded error
6007	Unrecognized equipment capability
6007	Unrecognized equipment capability and Duplicate equipment capability request
6008	Duplicate equipment capability request
10002	Client already has a session
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10010	The session id cannot be provided for establish session.
10016	Invalid CIMPortal Session

2.9.5.4 E132 Security Admin

2.9.5.4.1 GetDefinedPrivileges

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.4.2 GetACL

Code	Message

5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.4.3 AddACLEntry

Code	Message
5001	The subject name must be specified.
5001	At least one privilege must be specified.
5001	An ACL entry must be submitted with the request.
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6000	When the urn:semi-org:auth:allPrivileges privilege is specified, no other E134 privileges are allowed.
6000	ACLEntry containing the urn:semi-org.auth:securityAdminPrivileges privilege already exists.
6001	Duplicate ACL entry for <acl id>
6002	Unrecognized role <role>
6003	One or more privileges in the incoming request are not recognized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10012	urn:semi-org:auth:anyPrincipal cannot be assigned the urn:semi-org.auth:securityAdminPrivileges privilege.
10012	urn:semi-org:auth:anyPrincipal cannot be assigned a

	role with the urn:semi-org.auth:securityAdminPrivileges privilege
10013	Error: urn:semi-org:equipment is a reserved id for built in plan management. It is not allowable to create an ACL Entry with this subject id.
10016	Invalid CIMPortal Session

2.9.5.4.4 DeleteACLEntry

Code	Message
5001	The subject name must be specified.
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6004	ACL entry <id> not found.
6005	Unrecognized session <session id>
6010	Deleting role <role> with principals <principals> assigned.
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.4.5 GetActiveSessions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.4.6 SetMaxSessions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
5002	The maximum number of sessions must be greater or equal to zero.
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.4.7 GetMaxSessions

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Could not complete <operation> due to insufficient privileges
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name> is currently offline
10016	Invalid CIMPortal Session

2.9.5.5 E134

2.9.5.5.1 DefinePlan

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.

6005	Unrecognized session <session id>
8000	DCP Plan on request is null
8000	Error: Invalid IntervalInMinutes: '<value>'. Value must be greater than zero.
8000	Error: Data collection plan must contain at least one data collection request.
8000	Error: Data collection plan has an invalid identifier.
8000	Error: DCP id '<id>' must be unique
8000	ERROR: Validation failure: TraceId: '<id>' does not have any parameter requests.
8000	ERROR: Validation failure: Invalid GroupSize: '<size>' for TraceId: '<id>'
8000	ERROR: Validation failure: Invalid CollectionCount: '<count>' for TraceId: '<id>'
8000	ERROR: Validation failure: Duplicate traceRequest has been found in PlanId: '<plan id>', TraceId: '<trace id>'
8000	Error: Unable to locate SourceId: '<source>' for Id: '<id>'
8000	Error: Exception Trace Trigger must specify both Source And ExceptionId
8000	Error: Source Id '<id>' does not have any reportable exceptions.
8000	Error: Source Id '<id>' is not valid.
8000	Error: Severity '<severity>' is not valid.
8000	Error: The equipment does not have any reportable exceptions.
8000	Error: Source Id '<source>' does not have exception '<id>'
8000	Error: Source Id '<source>' does not have exceptions with '<severity>' severity.
8000	Error: Exception '<id>' is not valid.
8000	Error: Exception '<id>' in Source Id '<source>' does not match specified severity '<severity>'.
8000	Error: There are no Exceptions with Id '<id>' and severity '<severity>'.
8000	Error: Data collection plan has an error. Please see response object structure for more details.
8000	Error: Data collection plan has a structural error such as missing fields. Details: '<details>'
8000	ERROR: Multiple validation failures have occurred on DcpId: '<id>'. Check the Invalid request object return value to see all failure information
10001	The maximum number of DCPs has been exceeded.
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10004	Failure in the NVS system: <details>

10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.2 GetPlanDefinition

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
8001	The equipment does not recognize the requested planId <id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.3 GetDefinedPlanIds

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.4 ActivatePlan

Code	Message
5002	From field is null or empty in header

5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
8001	The equipment does not recognize the requested planId <id>
8002	The dcp with plan id '<id>' is active.
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.5 GetActivePlanIds

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.6 DeactivatePlan

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
8001	The equipment does not recognize the requested planId <id>
8003	The dcp with plan id '<id>' is not active.
10003	Invalid value, value is null

10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.7 DeletePlan

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
8001	The equipment does not recognize the requested planId <id>
8002	The dcp with plan id '<id>' is active.
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.8 GetParameterValues

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.9 GetObjTypeInstanceIds

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.10 GetCurrentPerformanceStatus

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value
10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.9.5.5.11 GetCurrentDateTime 0710 only

Code	Message
5002	From field is null or empty in header
5002	Equipment name must be populated in header
6000	Session does not have required capability <capability>
6000	Operation not authorized.
6005	Unrecognized session <session id>
10003	Invalid value, value is null
10003	Value is not in valid range
10003	Invalid value

10005	An unhandled exception has occurred. Message: <exception>
10007	CIMPortal API <api> returned error: <error>
10008	Equipment < name > is currently offline
10016	Invalid CIMPortal Session

2.10 Using the AppDCIM

The AppDCIM and its derived DCIMs are the main objects for data publication from equipment applications.

2.10.1 Valid Data Values for Interface A Data Types

The table below shows which SEMI E5 or Cimetrix Value Object (CxValueObject) data types that can be used for each of the Interface A parameter base types in the left column.

Interface A base type	Valid Value types
StringType, AnyURIType, EnumeratedType (EnumeratedString)	A(ASCII) or W(WCHAR)
DateTimeType	U8 or I8 The format of the 64 bit value should be in the format defined by Microsoft typedef FILETIME. See Microsoft C++ function GetSystemTimeAsFileTime and typedef FILETIME for more details. To get the current timestamp in C# use DateTime.Now.ToFileTimeUtc() function.
BooleanType	Bo
ByteType, Base64BinaryType	I1,I2,I4,I8,U1,U2,U4,U8 for ByteType; Bi for Base64BinaryType Value must be between -128 and 127
ShortType	I1,I2,I4,I8,U1,U2,U4,U8, Bi Value must be between -32768 and 32767
IntType, EnumeratedType (EnumeratedInt)	I1,I2,I4,I8,U1,U2,U4,U8, Bi Value must be between -2147483648 and 2147483647
LongType	I1,I2,I4,I8,U1,U2,U4,U8, Bi Value must be between -9223372036854775808 and 9223372036854775807
FloatType	F4 or F8 Value must be 0, 1.175494351e-38F .. 3.402823466e+38F, -3.402823466e+38F .. -1.175494351e-38F
DoubleType	F4 or F8
CompositeValueType	Composite Values may be one of three types. The Value mapping differs depending on the type: • **Array** - Value must be a List value when referencing String or other Composite values. For other types it may be a List (L {U4 1} {U4 2} {U4 3}), Array (U4 1 2 3 4 5) or Single (U4 5) value object. • **Structure** - Must be a List value with every item in the list complying with the corresponding StructureElement of the Structure. Number of StructureElement's in the Structure must equal number of items in the List value. For example, a Structure with a StringType, LongType and DoubleType elements may be represented as L {A StringValue} {I8 12345} {F8 -123.456}.

- **Variant** - A Value object of any type may be used. Values are translated using the following table:

Value Object Type	Interface A Value Type
A, W	StringType
Bi, single or array	Base64BinaryType
Bo single	BooleanType
Bo array	Array CompositeValueType of BooleanType
F4, F8	DoubleType
F4 array	Array CompositeValueType of FloatType
F8 array	Array CompositeValueType of DoubleType
single I1, I2, I4, I8, U1, U2, U4 or U8	LongType
I1, array	Array CompositeValueType of ByteType
I2 or U1, array	Array CompositeValueType of ShortType
I4 or U2, array	Array CompositeValueType of IntType
I8, U4 or U8, array	Array CompositeValueType of LongType
L	Variant CompositeValueType

If Value Object is empty, i.e. has type but no value (like U4 or F8), it is translated into NullType value. This rule doesn't apply to strings - empty string value objects are always translated into one of StringType, AnyURIType, EnumeratedType (EnumeratedString) types with empty "" value.

If Value Object is uninitialized, i.e. has no type or value, it is translated into NoValueType value with "Value Not Available" description.

If Value Object can't be translated because it's value exceeds minimum or maximum value defined by the Parameter Type Definition, it is translated into NoValueType value with "Parameter value overflow" description. Example: when using U2 130 value for ByteType parameter.

If Value Object can't be translated because its type doesn't match the Parameter Type Definition, it is translated into NoValueType value with "Invalid parameter type" description.

If there were other problems during the Value Object translation, it is translated into NoValueType value with description set by exception.

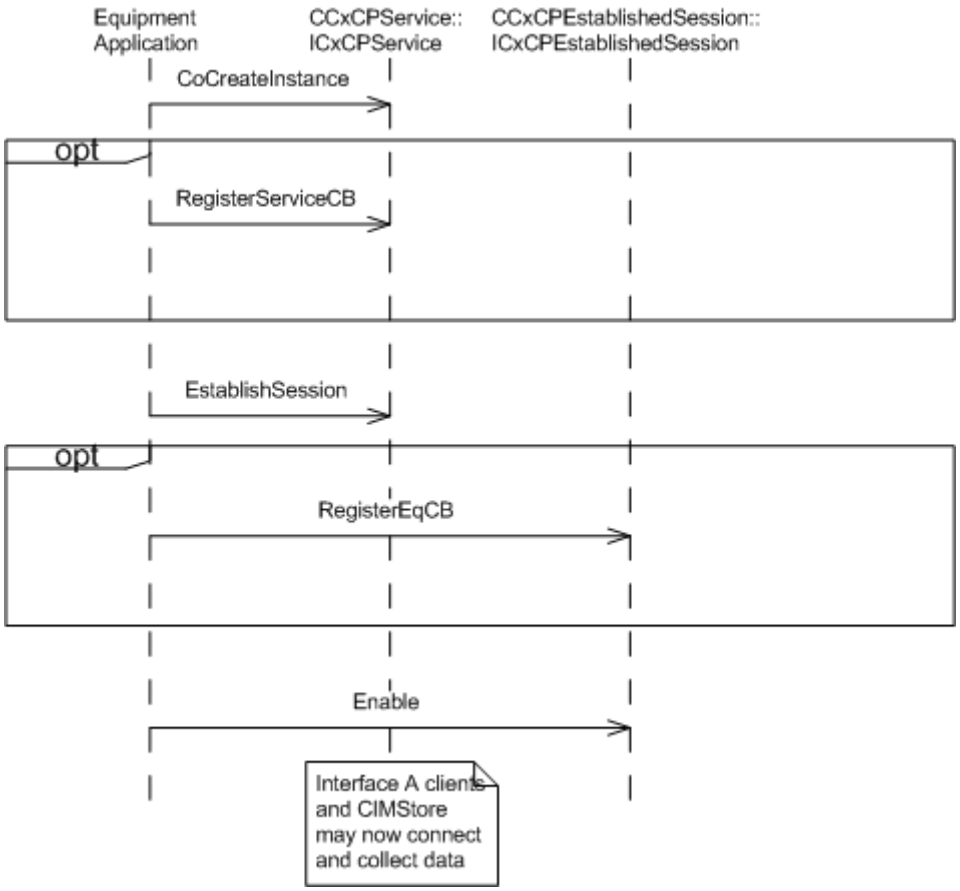
2.10.2 Connect to CIMPortal Plus

To connect to CIMPortal Plus, you first create an instance of the connection server CCxCPService and then request an equipment connection from it. If the user name and password you provide are valid, you will get back an ICxCPEstablishedSession pointer to use for operations on that equipment.

Enable should be called to notify CIMPortal Plus that the equipment application is ready and wants CIMPortal Plus to allow clients to collect data.

Optionally, you may register service or equipment callbacks to be notified of changes in the service or equipment states.

In the diagram below, CoCreateInstance and EstablishSession are required calls. RegisterServiceCB and RegisterEqCB are optional calls. Enable is a recommended call.



2.10.3 Data Publishing

Use GetDCIM to get the interface to the AppDCIM in CIMPortal Plus. Use the API to publish data, events, and exceptions.

In the diagram below, GetDCIM and QueryInterface are required calls.

RegisterCB is an optional call.

UpdateData, UpdateDataById, or UpdateDataByIdByteStream are recommended for updating parameters values.

If publishing alarms, the following API are recommended: TriggerException, TriggerExceptionByteStream, TriggerExceptionDataById, TriggerExceptionDataByIdByteStream, TriggerExceptionTS, TriggerExceptionByteStreamTS.

If publishing events, the following API are recommended: TriggerEvent, TriggerEventByteStream, TriggerEventDataById, TriggerEventDataByIdByteStream, TriggerEventTS, TriggerEventByteStreamTS.

When trigging events, the source of the event must be identified. The model may contain multiple state machine instances, each with an event by the same name. The new sourceID argument for the TriggerEvent family of API methods is used to distinguish between these events since the event names are not unique. This argument indicates which node is producing the event that is triggered and so should contain the locator for that node.

The sourceID is the path in the model to the parent hierarchy node of the event. For example: Equipment/EFEM/Loadport1 would be the sourceID for event Equipment/EFEM/Loadport1/CarrierIDReadFail.

For events in state machine instances, this is the parent hierarchy node of the state machine instance. For example: Equipment/EFEM/Loadport1/E99 would be the sourceID for state machine instance event Equipment/EFEM/Loadport1/E99/IDReaderSMInst/IDReaderAvailable.

2.10.4 Working with E39 SEMI Objects

According to E164, static E39 objects such as substrate locations and EPT Tracker objects should be published through normal parameters and events related to LogicalElements (or Subsystems for the 1105 freeze). Use the normal data publication API for these. The remainder of this section is for dynamic E39 objects.

Use AddSemiObjectInstance or AddSemiObjectInstanceByteStream to identify instances of E39 objects. Call one of these when carriers, substrates, or jobs are created.

Use RemoveSemiObjectInstance to remove E39 objects when they are deleted, such as when a carrier is removed or a job completes. Use RemoveAllSemiObjectInstances to clear the SEMI object cache.

Use UpdateSemiObjectInstanceAttributeValues or UpdateSemiObjectInstanceAttributeValuesByteStream to update SEMI object instance attributes when they change.

Use TriggerSemiObjectInstanceEvent or TriggerSemiObjectInstanceEventByteStream to publish events related to SEMI objects.

Use ListAllSemiObjectInstances to retrieve a collection of all SEMI object instances currently in the cache.

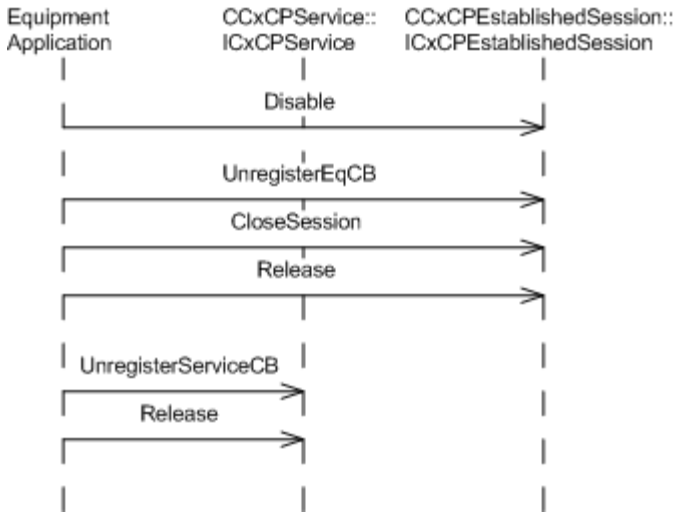
2.10.5 Disconnect from CIMPortal Plus

When an application is shutting down, it should follow the sequence below.

If callbacks were registered, they should be unregistered.

Disable tells CIMPortal Plus that the equipment application does not want any further data collection by Interface A clients to occur.

In the diagram below, CloseSession and Release are required calls. UnregisterServiceCB and UnregisterEqCB are optional calls. Disable is a recommended call.



2.11 Managing Built-In Plans

Built-in plans are DCPs that are stored persistently on the equipment, typically by the equipment vendor. They are available to all users who have permission to access DCPs. They can be enabled and disabled, but they cannot be changed or deleted except by a special user.

They are stored in the same directory as the DCPs in non-volatile storage. The Built-in DCPs are stored in [CIMPortal Base Storage]/[EquipmentName]/Runtime/Plans where [CIMPortal Base Storage] is the base storage directory for CIMPortal Plus and [EquipmentName] is the name of the equipment package which was deployed to the CIMPortal Plus Service.

Each Built-In DCP stores the following information in an XML file:

- E134 Definition: this includes the name, parameters, events, exceptions, etc.
- Date, Time, and Client name (`urn:semi-org:equipment`) when the Built-In plan was defined
- Data, Time, and Client(s) that activated the DCP. Note that if the 'IsPersist' flag is **not** set to 'true' when defining the DCP, then the activation information will not be stored and used during an equipment restart.

When CIMPortal Plus is started, the CIMPortal Plus Runtime engine reads the contents of the persisted DCPs directory and re-creates each DCP. If the DCP is a Built-In DCP, it is re-created just as regular DCPs are re-created.

To manage built-in DCPs, an Interface A client must connect to CIMPortal Plus with the predefined SEMI client name `urn:semi-org:equipment`. CIMPortal Plus will only allow this client to connect if the "Manage built-in plans" option is checked in the Control Panel on the General tab on the equipment's CIMWeb page. (See the Control Panel section 2.2.2.2.) If this option is not checked, this user will not be allowed to connect. For security purposes, the "Manage built-in plans" setting is forced to off when CIMPortal Plus starts.

Once this special user is connected through Interface A, any DCPs defined with the IsPersistent flag set to true will be stored as a Built-In DCP. They can also be removed just like any other DCP by this user via the DeletePlan method.

These DCPs will be available to any user with sufficient privileges to see and activate DCPs. These other uses will be able to activate and deactivate the built-in DCPs. However, if they try to delete or change them, they will receive an error indicating they are not authorized to do so.

Built-in DCPs must be defined for each CIMWeb version, for each equipment that is desired to have the plan. Neither CIMWeb instances nor equipment deployed in CIMPortal Plus share built-in DCPs.

2.12 Hardware Metadata Update

Cimetrix provides an ActiveX control that can be used for updating the Hardware List file with appropriate information. This Hardware List file allows the E120 equipment model and software modules metadata information to be updated after the equipment is deployed. For example, it allows the actual hardware serial numbers and software versions to be specified in the model without requiring a unique equipment model for each piece of equipment.

2.12.1 Installed Files

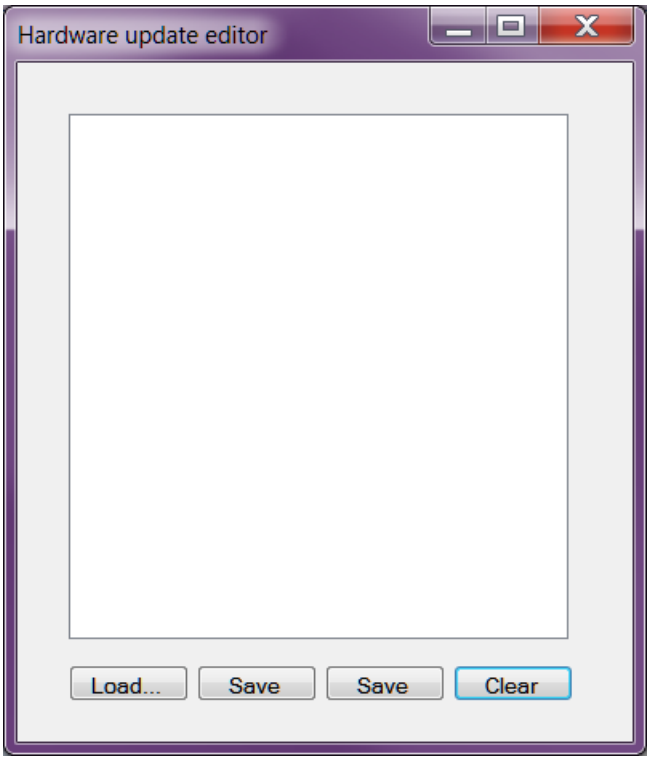
The CIMPortal Plus installation installs and registers a file called HardwareUpdate.dll. This file contains the ActiveX control that can be added to an end user application.

2.12.2 Adding to an Application

The name of the user control in the .dll file is HardwareUpdateCtrl

2.12.3 Updating Hardware Information

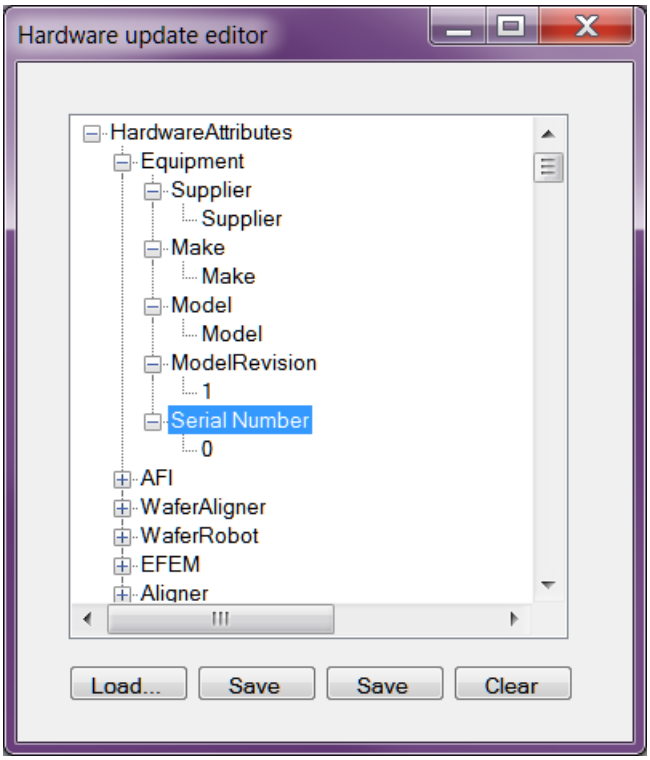
The ActiveX control appears as follows:



The large white area is a tree control that will display the hardware information that can be updated. The control contains the following buttons:

- Load - Open a hardware xml file created in Hardware List
- Save - Save the file with the current filename.
- Save As - Save the file as a name entered in a file selection dialog
- Clear - Clear the tree

When a file is loaded, the control appears as follows:



The user can modify the Supplier, Make, Model, ModelRevision and Serial Number of each E120 object and Supplier and SoftwareVersion of each Software Module object as needed. These fields are edited by either double-clicking the field to change or selecting the field and then clicking a second time on it. The field will be updated to show that it can be edited. The user enters the new information for the hardware or software. Clicking anywhere else in the application or hitting the Enter key will cause the edited change to appear in the tree.

Once all desired changes are made to the list, the user saves the file by clicking the Save or Save As button.

2.12.4 Applying Updated Information to the Model

When CIMPortal Plus Service loads an equipment model file (either at startup or during deployment), it also tries to read a file named "Hardware.xml". If it finds the "Hardware.xml" file, it will update the model with the information found in this file. Whenever a query is done on the E120 elements in the model, the information from this file would be returned. If the "Hardware.xml" file is changed, then CIMPortal Plus Service must be restarted before the changes will take effect.

The Hardware.xml file should be located with the model where it is deployed in the Program Files\Cimetrix\Comm Products\bin\Storage or Program Files(x86)\Cimetrix\Comm Products\bin\Storage\Deployed Equipment folder in a subfolder named the same as the name of the Equipment node in the model. For example: C:\Program Files\Cimetrix\Comm Products\bin\Storage\Deployed Equipment\Equipment.

2.13 Using DCOM to Collect Data from Multiple PCs

Using the AppDCIM and DCOM technology, CIMPortal Plus can collect data from a separate PC. However, for .NET programs, the WCFAppDCIM would be the preferred solution.

The following sections detail how to use the AppDCIM and DCOM.

2.13.1 Warning

DCOM protects PC and provides a safe and secure access from another PC. The procedure described in this section is uncomplicated usage of DCIM from another PC with CIMPortal Plus. This method is not for secure usage.

2.13.2 How to:

Collect Data from a PC with CIMConnect via CIMConnectDCIM

1. Configure XP DCOM Settings for CIMPortal Plus
2. Configure DCOM Settings of Equipment PC, from which data is collected and sent to CIMPortal Plus PC.
   Configure DCOM Settings of Win2K PC

   or

   Configure DCOM Settings of XP PC
3. Configure Equipment PC with CIMConnect
4. Implement CIMConnectDCIM on CIMPortal Plus PC

2.13.3 Configuring XP DCOM Settings for CIMPortal Plus

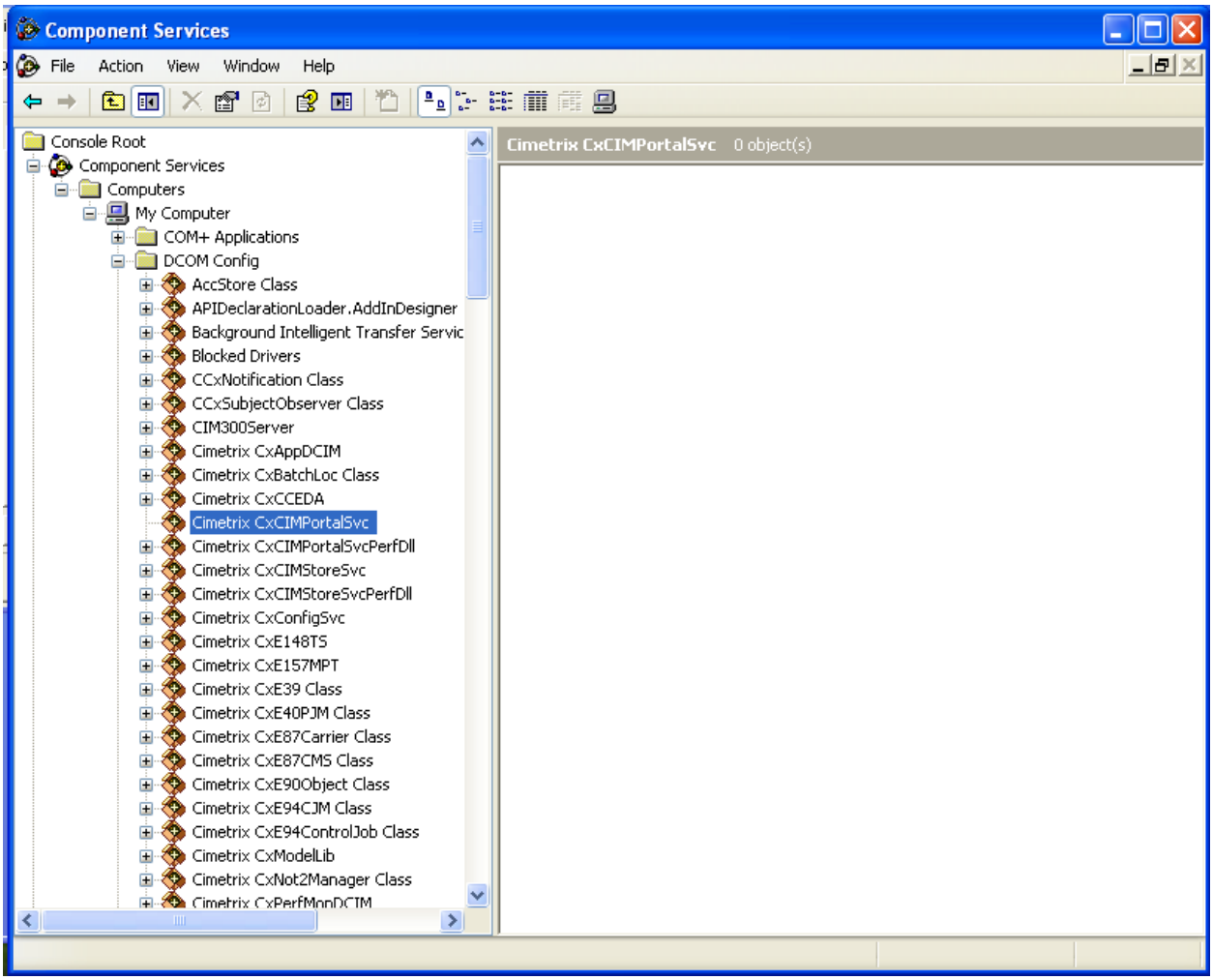
When users want to collect data from a different PC into a computer with CIMPortal Plus, user must modify DCOM setting of CIMPortal Plus. This document contains detailed configuration instruction.

2.13.3.1 Warning

DCOM protects PC and provides a safe and secure access from another PC. The procedure described here is for uncomplicated usage of DCIM from another PC with CIMPortal Plus. This method is not for secure usage.

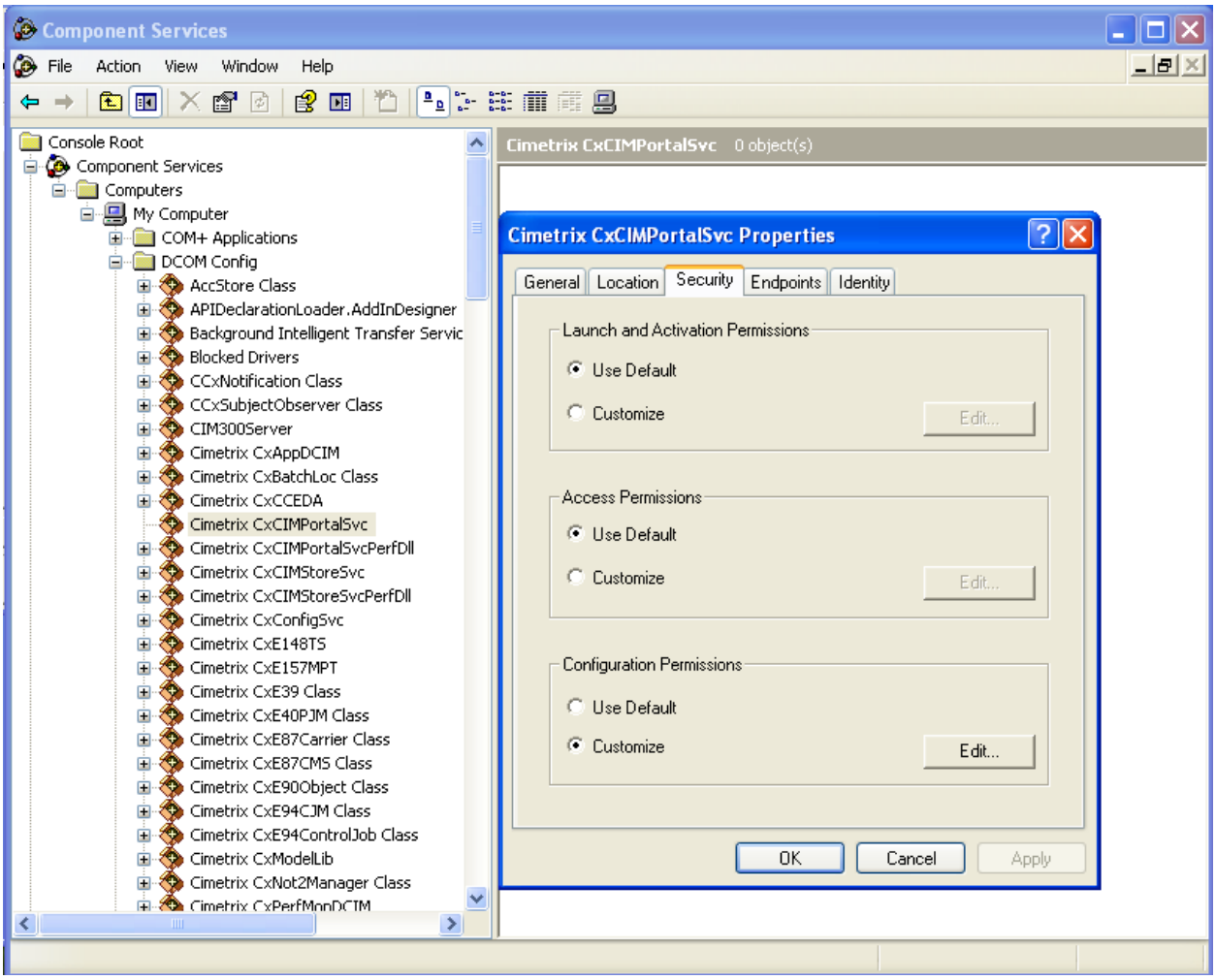
2.13.3.2 Do this:

1. Disable firewall of CIMPortal Plus PC. (If you don't want to compromise your CIMPortal Plus PC, instead of disabling firewall, add the designated RPC port ID to Equipment PC.)
2. Modify DCOM setting for CxCIMPortalSvc on CIMPortal Plus PC.
   At a command prompt, type **dcomcnfg**, and then press ENTER. Open Component Services. Select "Cimetrix CxCIMPortalSvc" and select its properties.

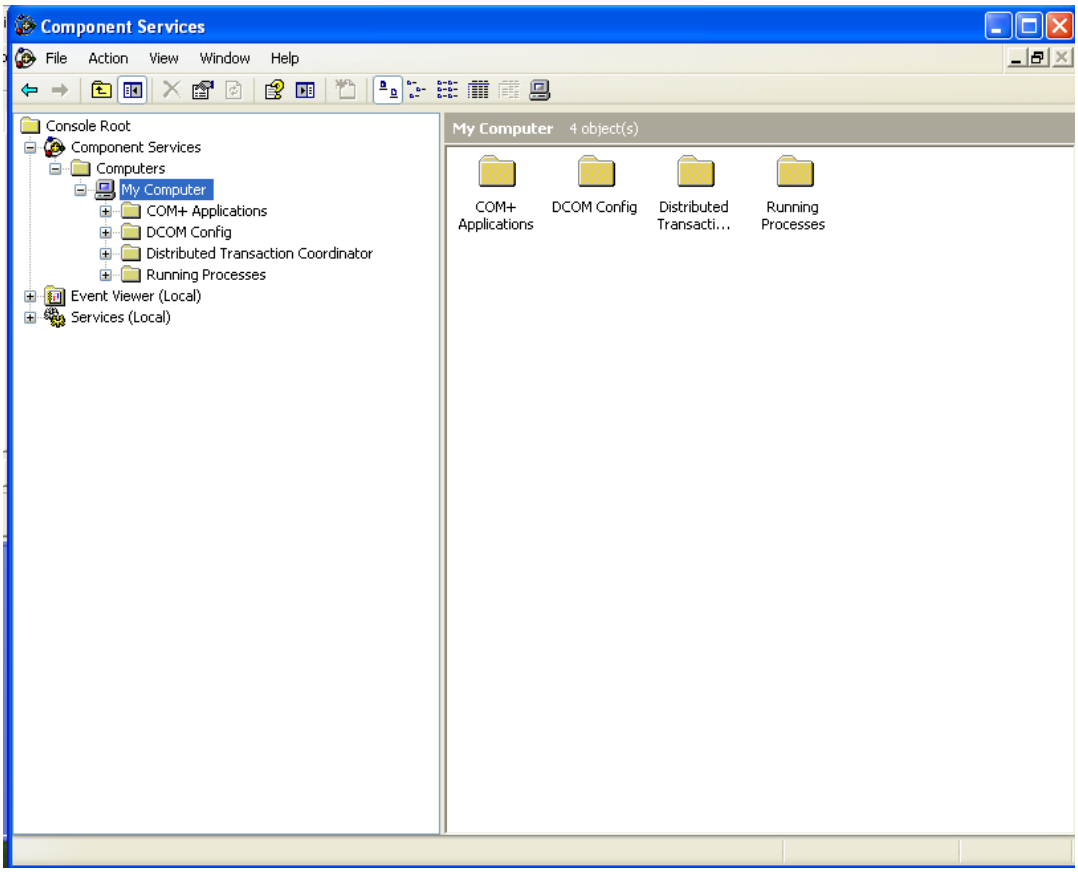


Select the "Security" tab from CxCIMPortalSvc Properties dialog window. Configure it as following:

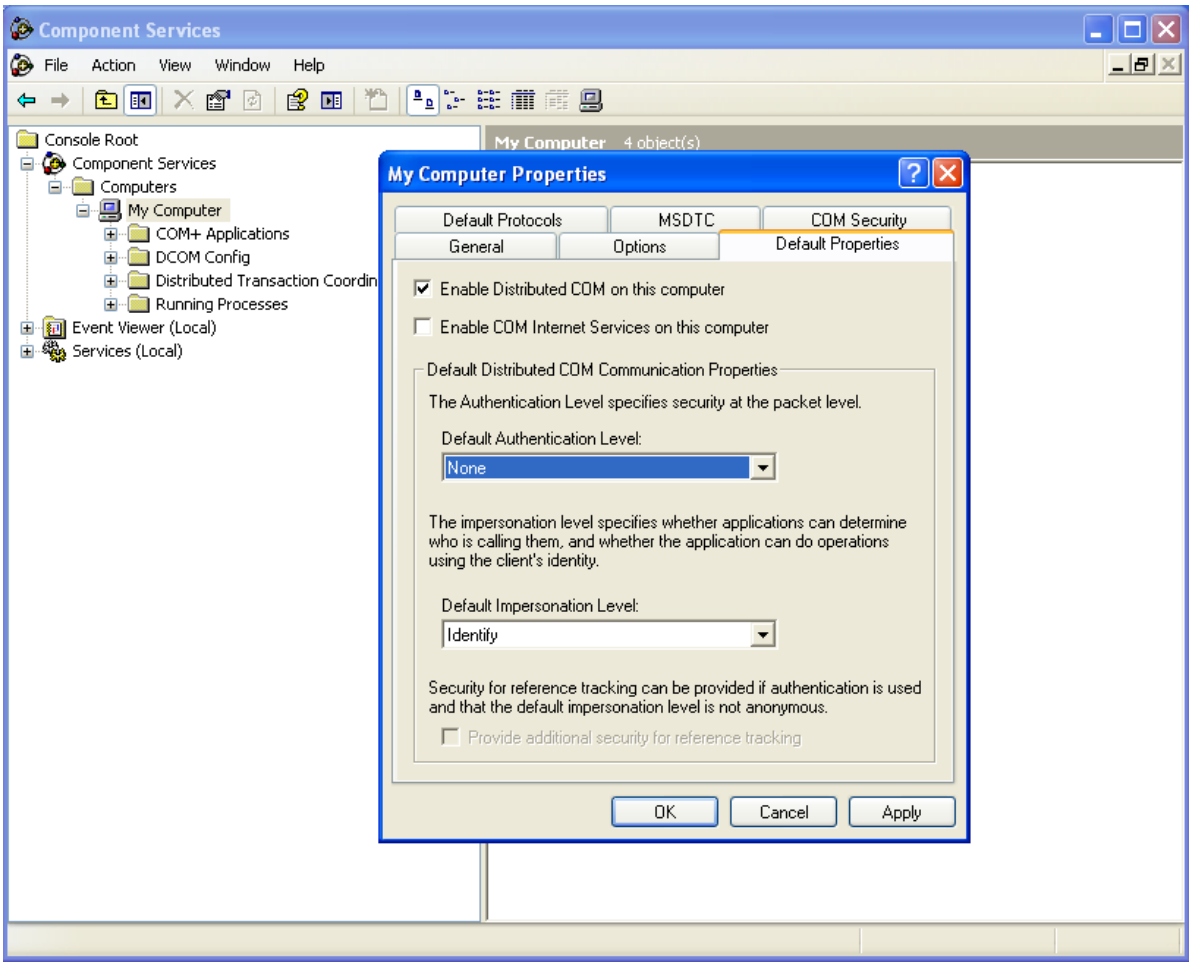
Group Name	Appropriate Setting
Launch and Activation Permission	Use Default
Access Permissions	Use Default



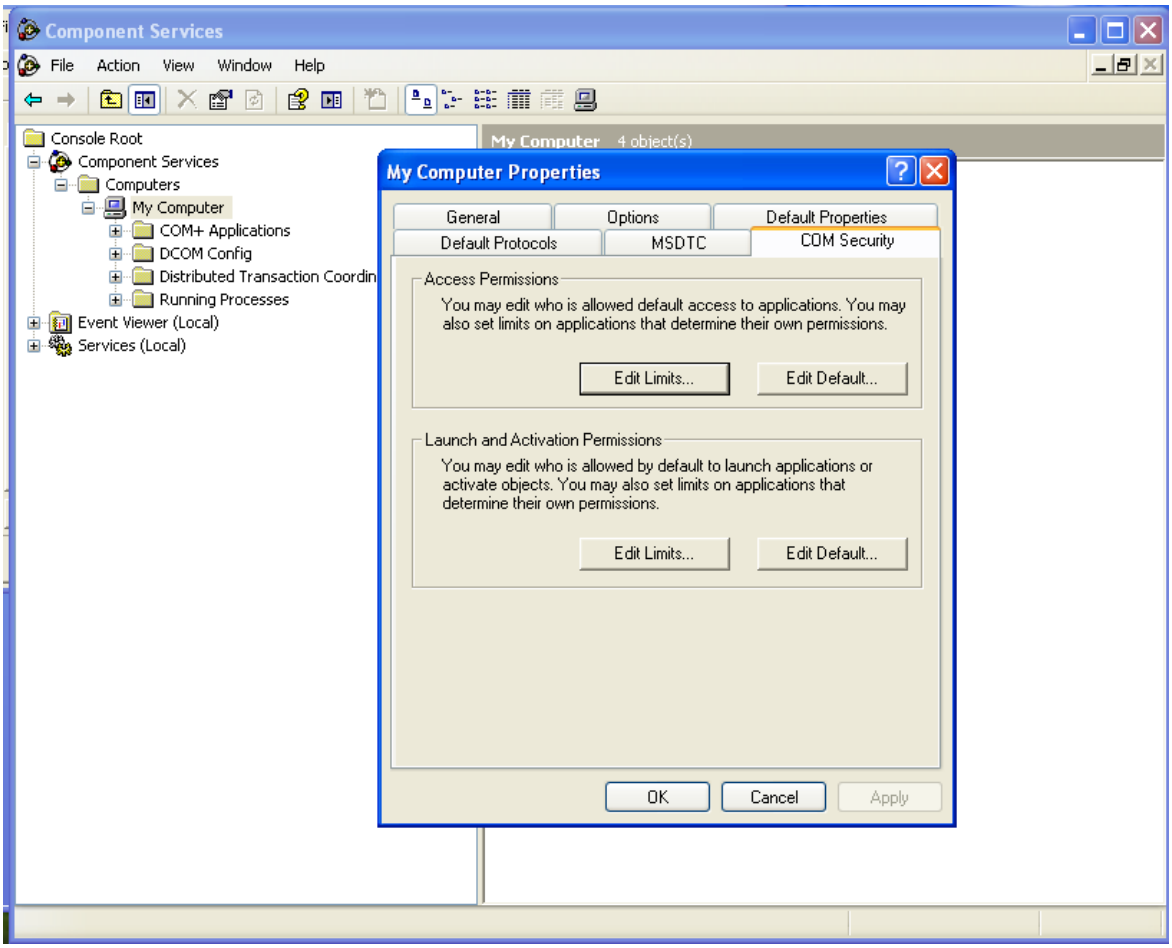
3. Set necessary DCOM setting for CIMPortal Plus PC.
   At a command prompt, type **dcomcnfg**, and then press ENTER. Open Component Services. Select "My Computer" under Computers. Select its Properties.



4. Select "Default Properties" tab. Configure as following to add DCOM and disable the authentication.
   - Check "Enable Distributed COM on this computer" checkbox.
   - Select "None" for "Default Authentication Level".
   - Select "Identify" or "Impersonate" for "Default Impersonation Level".

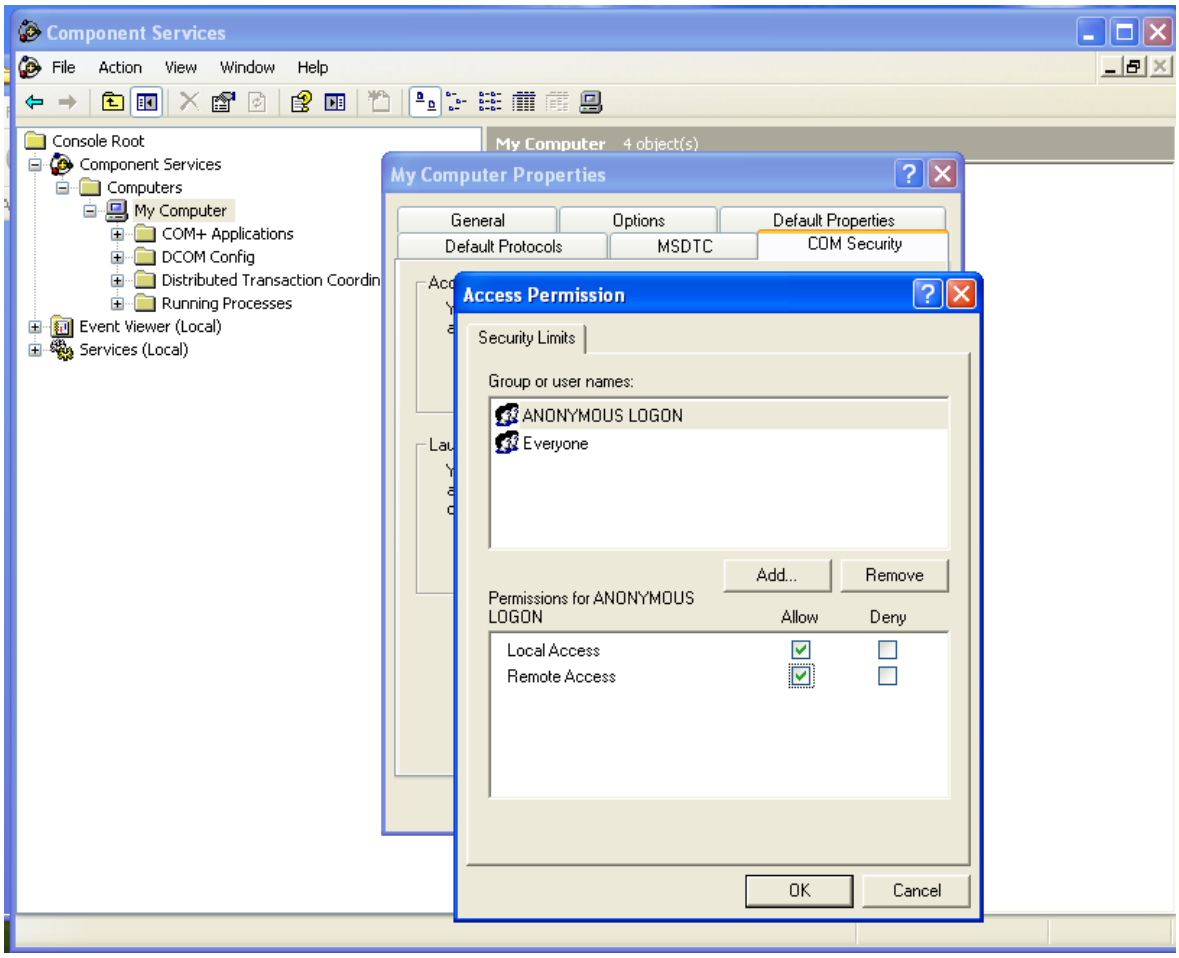


5. Select "COM Security" tab.



6. Click "Edit Limits..." for "Access Permissions" group. Make sure to have the following setting.

Group or User Names	Allow
ANONYMOUS LOGIN	Local Access Remote Access
Everyone	Local Access Remote Access



7. Click "Edit Default..." for "Access Permissions" group. Make sure to have the following setting.

Group or User Names	Allow
ASP.NET	Local Access
Everyone	Local Access Remote Access
SELF	Local Access Remote Access
SYSTEM	Local Access

8. Click "Edit Limits..." button in "Launch and Activation Permissions" group. Make sure to have the following setting.

Group or User Names	Allow
Administrators	Local Launch Remote Launch Local Activation Remote Activation
Everyone	Local Launch Remote Launch

	Local Activation Remote Activation

9. Click "Edit Default..." button in "Launch and Activation Permissions" group. Make sure to have the following setting.

Group or User Names	Allow
Administrators	Local Launch Remote Launch Local Activation Remote Activation
Everyone	Local Launch Remote Launch Local Activation Remote Activation
INTERACTIVE	Local Launch Remote Launch Local Activation Remote Activation
SYSTEM	Local Launch Remote Launch Local Activation Remote Activation

2.13.4 Configuring DCOM Setting of Windows Win2K PC

When users want to collect data from a different PC into a computer with CIMPortal Plus by CIMConnectDCIM, user must modify DCOM setting of the PC from where the data is collected. This document contains details instruction.

2.13.4.1 Warning

DCOM protects PC and provides a safe and secure access from another PC. The procedure described here is for uncomplicated usage of DCIM, collecting and sending data to a separate PC with CIMPortal Plus. This method is not for secure usage.

2.13.4.2 Do this:

1. Configure DCOM Setting for Equipment PC.
2. At a command prompt, type **dcomcnfg**, and then press ENTER. Open Distributed COM Configuration Properties.
3. Select "Default Properties" tab. Configure as following to add DCOM and disable the authentication.
   - Checkbox on "Enable Distributed COM on this computer".
   - Select "None" for "Default Authentication Level".
   - Select "Impersonate" or "Identity" for "Default Impersonation Level".

   (Warning: Do not select "Anonymous" for "Default Impersonation Level". This disables EMService.)
4. Select "Default Securities" tab.
5. Click "Edit Default…" buttons for "Default Access Permissions" group to add "Administrators", "Everyone", INTERACTIVE", "NETWORK", and "SYSTEM" with "Allow Access".

6. Click "Edit Default…" buttons for "Default Launch Permissions" group to add "Administrators", "Everyone", INTERACTIVE", "NETWORK", and "SYSTEM" with "Allow Launch" setting.
   Configuring DCOM Setting of Windows XP PC

2.13.4.3 Warning

DCOM protects PC and provides a safe and secure access from another PC. The procedure described here is for uncomplicated usage of DCIM, collecting and sending data to a separate PC with CIMPortal Plus. This method is not for secure usage.

2.13.4.4 Do this:

1. Disable Firewall on Equipment PC.
2. Configure DCOM Setting of Equipment PC
3. At a command prompt, type **dcomcnfg**, and then press ENTER. Open Component Services. Select "My Computer" under Computers. Select its Properties.
4. Select "Default Properties" tab. Configure as following to add DCOM and disable the authentication.
5. Checkbox on "Enable Distributed COM on this computer"
6. Select "None" for "Default Authentication Level".
7. Select "Identify" or "Impersonate" for "Default Impersonation Level"
8. Select "COM Security" tab.
9. Click "Edit Limits..." for "Access Permissions" group. Make sure to have the following setting.

Group or User Names	Allow	Deny
ANONYMOUS LOGIN	Local Access Remote Access	
Everyone	Local Access Remote Access	

10. Click "Edit Default..." for "Access Permissions" group. Make sure to have the following setting.

Group or User Names	Allow	Deny
ANONYMOUS LOGIN	Local Access Remote Access	
Everyone	Local Access Remote Access	
SYSTEM	Local Access	

11. Click "Edit Limits..." button in "Launch and Activation Permissions" group. Make sure to have the following setting.

Group or User Names	Allow	Deny

ANONYMOUS LOGON	Local Launch Remote Launch Local Activation Remote Activation	
Everyone	Local Launch Remote Launch Local Activation Remote Activation	

12. Click "Edit Default..." button in "Launch and Activation Permissions" group. Make sure to have the following setting.

Group or User Names	Allow	Deny
ANONYMOUS LOGON	Local Launch Remote Launch Local Activation Remote Activation	
Everyone	Local Launch Remote Launch Local Activation Remote Activation	
SYSTEM	Local Launch Local Activation	

2.14 Using SSL with Interface A clients

2.14.1 SSL Overview

SSL (Secure Sockets Layer) is cryptographic protocol that provides communication security (more info at http://wikipedia.org/wiki/Secure_Socket_Layer). SSL is application protocol-independent and transparently supports many higher-layered transport protocols such as SOAP over HTTP.

EDA requires mutual authentication between Equipment and Client(s) when SSL is used. Equipment (CIMPortal Plus) always sends its certificate and requests the client's certificate as a part of the SSL handshake. The Client (EDAConnect or other) has to send its own certificate back to the Equipment. CIMPortal Plus validates that the Client's ID ("From" field of the SEMI E132 header) against Client certificate's CommonName property and, if do they do not match, rejects the request from the Client.

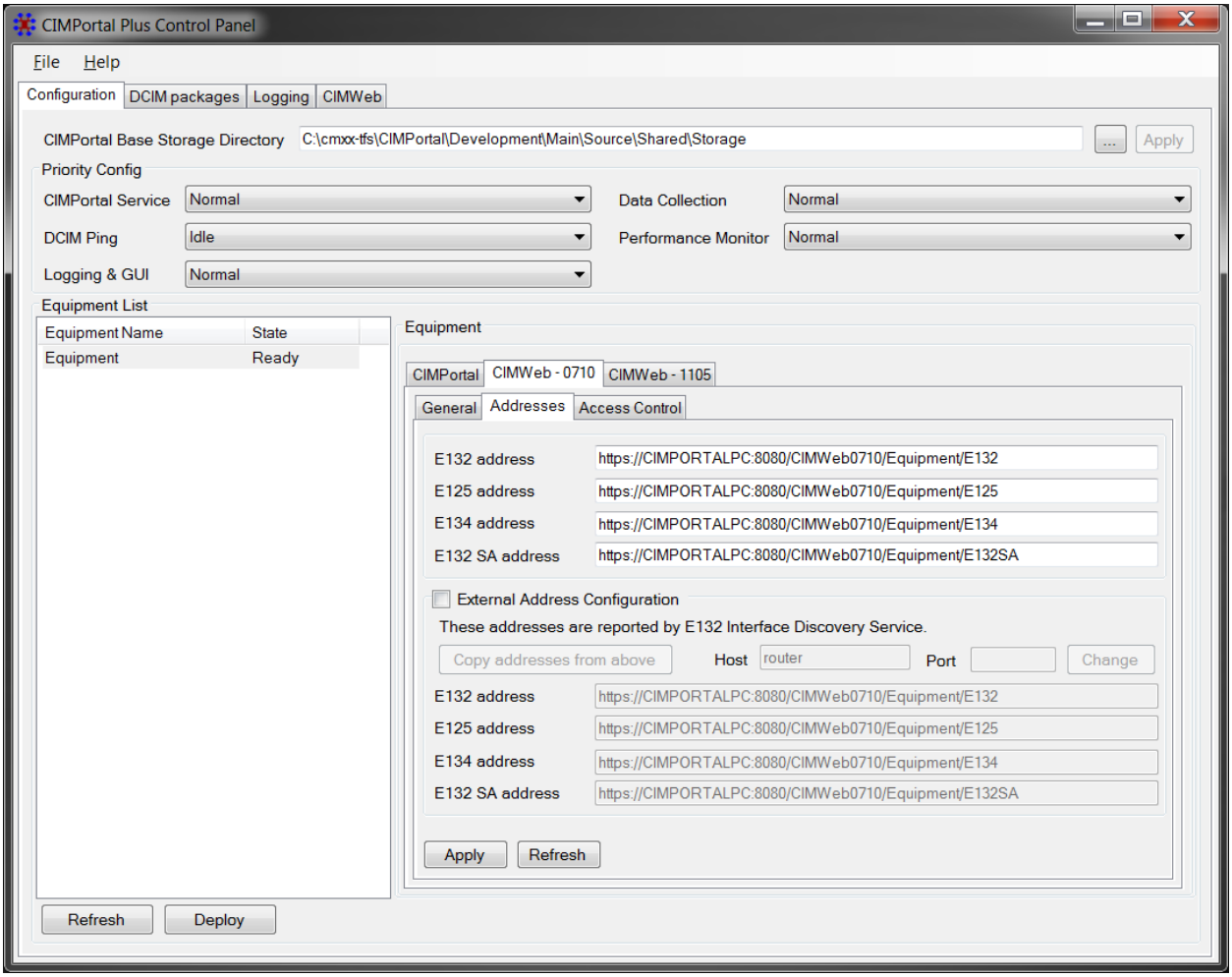
CIMPortal Plus does support SSL connection to the Client consumer, but does not support mutual authentication, i.e. it does not provide its own certificate to the Client consumer web service.

2.14.2 Setup

1. Install a valid certificate for CIMPortal Plus's use in Windows Local Machine Certificate Store (see http://msdn.microsoft.com/en-us/library/ms731899.aspx for more information). Note that the certificate's CommonName must match the computer name where CIMPortal Plus is running.
2. Configure a Port with the SSL Certificate (see http://msdn.microsoft.com/en-us/library/ms733791.aspx for more information). Note that the port cannot be shared with non-SSL connections once configured.

3. Changes to the certificates in Windows Certificate Store and to Port SSL configuration require CIMPortal Plus Service to be restarted.

4. Change CIMWeb EDA service URLs from "http" to "https" in CIMPortal Plus Control Panel application. The URLs must specify the port used in step 2. Note that CIMPortal Plus Service must be restarted for the changes to take effect.



5. If the server certificate is invalid (its CommonName does not match computer name or issued by an unknown CA, etc.), any client trying to establish a session will receive an error similar to the following explaining that the https connection cannot be established because of invalid credentials:

   a. Could not establish trust relationship for the SSL/TLS secure channel.
   b. The remote certificate is invalid according to the validation procedure.

2.15 CIMPortal Plus Redistribution

This section describes how to distribute CIMPortal Plus on manufacturing equipment to the end users.

2.15.1 General Information

CIMPortal Plus has only one installation for all versions, including for both Software Development Kits and runtime versions. Therefore, the same installation file can be used for development and redistribution. Customers are encouraged to use sound configuration management strategies. For example, customers are strongly recommended to do the following:

- Create a master hard drive with CIMPortal Plus runtime installed and all other equipment software. Remove the license file from the master hard drive so that it is not illegally distributed to equipment.
- Before upgrading to new CIMPortal Plus service releases or patches on production systems in the field, first upgrade a test system in-house and perform sufficient testing.
- Deploy the equipment in CIMPortal Plus
- Make any necessary changes to the CIMPortal Plus configuration file.
- Use the Cimetrix Configuration Manager to configure proper access for any applications
- Include any additional files or applications for data collection
- When distributing & upgrading CIMPortal Plus, it is important to save the CIMPortal Plus.cfg, CMXUsers.cfg and CMXConfig.xml files to quickly restore the system to its correct setup. Backup the new CIMPortal Plus files before overwriting them and test this carefully to ensure file formats have not changed in the new version.

2.15.2 CIMPortal Plus Interface A (CIMWeb)

Follow these steps to redistribute CIMPortal Plus Interface A.

- Request and install a CIMPortal Plus Interface A Runtime License from Cimetrix
- Use the Cimetrix Configuration Manager to create E132 administrative session name with E132 admin privileges
- Use the Cimetrix Configuration Manager to predefine any additional desired E132 ACL entries
- Create any desired built-in data collection plans. See section 2.11 for details.
- If using this feature, include the configured metadata (hardware) file
- After everything is setup, use ECCE to validate the CIMPortal Plus and CIMWeb work properly at the target data rates

2.15.3 CIMPortal Plus DB

Follow these steps to redistribute CIMPortal Plus DB

- Install the selected database
- Request and install a CIMPortal Plus DB Runtime License from Cimetrix
- Add the equipment to CIMStore using CIMStore Manager
- Setup the desired data collection requests

2.15.4 CIMPortal Plus Bundle

Follow the instructions for both CIMPortal Plus Interface A and CIMPortal Plus DB. Request a CIMPortal Plus Bundle Runtime license which will include both Interface A and DB versions.

2.16 COM Interface programming

The Windows' COM Interface standard provides a means for cross-language interoperability. Most programming languages in the Windows environment support some means of calling COM interfaces. Because of this common factor, all programmatic interfaces to CIMPortal Plus use COM interfaces. It is beyond the scope of this document to explain how to use COM interfaces in any particular language. This section aims to address some issues that are common to most languages. For specific details regarding a particular language, it's recommended reading the language's reference documentation about using COM interfaces and searching the web for specific issues.

2.16.1 Memory allocation

Memory allocation and deallocation is handled automatically by many languages. One notable exception is C/C++. In this language, the user must be sure to free any string values returned from CIMPortal Plus COM interfaces using the SysFreeString to ensure there are no memory leaks. For example:

```
// Sample API call populating a CxCPReturnCode
CxCPReturnCode myCxCPReturnCode = {0};

// Access the equipment's session in CIMPortal Plus
hr = m_pICxCPService->EstablishSession( bstrEQName, bstrClientName,
    bstrPasswords, &myCxCPReturnCode, &m_ICxCPEstablishedSessionPtr );

// If there is any error, free sErrorString of CxCPReturnCode to avoid memory leak.
If (NULL != myCxCPReturnCode.sErrorString)
    SysFreeString(myCxCPReturnCode.sErrorString);
```

2.16.2 .Net references

For .Net languages to work with COM, the .Net framework requires Interop files to handle the marshalling between .Net types and COM types. These interop files may be generated in a number of ways. One way these files may be generated is by a direct reference in a .Net project to the COM library. In this case, the build system will create an Interop file based on the registration details in the registry for that library when that project is built. This works OK as long as all other assemblies in the process space are built against this Interop file.

To follow Microsoft COM/.Net guidelines, CIMPortal Plus provides Interop files for all COM libraries it supplies. (See MSDN article "Producing Primary Interop Assemblies" at http://msdn.microsoft.com/en-us/library/hfac4fky%28v=vs.110%29.aspx for more details.)

Early releases of CIMPortal only contained Interop files for COM interfaces that CIMPortal referenced directly. Because of this, some customer applications may reference the COM library directly and generate their own Interop files. To ensure consistency in COM calls between different .Net assemblies, customers are recommended to replace their Interop files with the ones provided by CIMPortal Plus.

If a project references a different Interop file or has a direct reference to a COM library, the reference should be removed and a new reference added pointing to the correct Interop file. The Interop files are typically found in the CIMPortal Plus install location (i.e. "%ProgramFiles%\Cimetrix\Comm Products\bin").

2.17 Troubleshooting

This is a collection of known issues and how to resolve them.

2.17.1 Windows Server 2003

Windows License Manager has a potential issue where it does not release licenses associated with stale connections. These licenses can count against the total licenses. We recommend disabling Windows License Manager.

2.17.2 EDA Client Connectivity Test

If an EDA client (like ECCE or some other EDA client software) is unable to establish an Interface A session with the equipment using CIMPortal Plus with CIMWeb, then please use this checklist **in sequence** to help determine the cause of the problem. Do not proceed to further steps until the previous steps are completed successfully. This guide does not purport to propose resolutions to any problems. Instead, it simply tries to identify the most descriptive symptom so that an expert can resolve the issue more effectively.

In these tests, there are typically only two computers, the client computer and the equipment computer. The equipment computer is where CIMWeb is running. The client computer could potentially be two computers,

although typically it will be just one computer. EDA client software must implement a consumer web service to receive the equipment-initiated operations including the data reports. This client consumer computer and its web service URL are identified in the EstablishSession operation as the HTTPEndPoint value. A client consumer web service could conceivably run on another computer separate from the software that initiates operations to the equipment's web service. The instructions below distinguish between the client computer and client consumer computer, even though they could and generally will be the same.

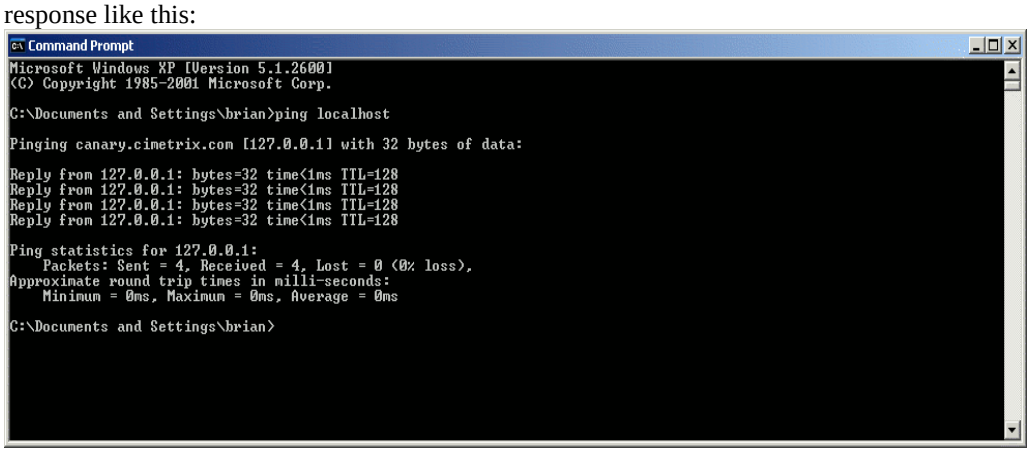
These tests can be run using ECCE (version 1) or ECCEv2. ECCEv2 is recommended. The client consumer URL listed is based on the default for ECCEv2.

2.17.2.1 Verify Basic TCP/IP on the Client Computer

On the client computer, open a command prompt window. In this window type

```
ping localhost
```

to verify TCP/IP is installed and running correctly on this computer. If it succeeds, then you will see a response like this:



If this fails, then please contact your System Administrator for computer setup assistance.

2.17.2.2 Verify Basic TCP/IP on the Equipment's Computer

On the equipment's CIMWeb computer, open a command prompt window. In this window, type

```
ping localhost
```

to verify TCP/IP is installed and running correctly on this computer.

If this fails, then please contact your System Administrator for computer setup assistance.

2.17.2.3 Ping from Client Computer to Equipment Computer

On the client computer, open a command prompt window. In this window use the **ping** command to verify TCP/IP communication with the equipment's computer where CIMWeb is running. You may use the equipment computer name or IP address depending on what the EDA client is actually using to establish a session. For example, type something like this:

```
ping 192.160.0.10
```
or
```
ping equipmentComputerName
```

and verify 0% packet loss.

If this fails, then contact your System Administrator for network setup assistance. It could be a hardware or software issue. The client's computer must have TCP/IP access to the equipment's CIMWeb computer.

2.17.2.4 Ping from Equipment Computer to Client Consumer Computer

On the equipment's computer where CIMWeb is running, open a command prompt window. In this window use the **ping** command to verify TCP/IP communication with the client consumer computer. You may use the computer name or IP address depending on what the EDA client is using in the HTTPEndPoint URL address specified in the client's EstablishSession operation. For example, type something like this:

```
ping 192.160.0.1
or
ping edaComputerName
```

and verify 0% packet loss.

If this fails, then contact your System Administrator for network setup assistance. It could be a hardware or software issue. The equipment's CIMWeb computer must have TCP/IP access to the client's computer. When the client establishes a session, it must specify the HTTPEndPoint of a computer that is accessible.

2.17.2.5 Verify Local Access to the CIMWeb Web Services

On the equipment computer, open an internet browser like Internet Explorer. Enter the E132 URL into the internet browser and verify access. The URL should be something like this:

```
http://localhost:8080/CIMWeb0710/EquipmentName/E132
or
http://equipmentComputerName:8080/CIMWeb0710/EquipmentName/E132
```

If successful, the internet browser will not show an HTTP error. Run the same test for each of the CIMWeb URLs; E125, E132, E132SA, and E134. If running both 1105 and 0710 version of the web services, then be sure to test both sets of URLs. Also check the E132 Interface Discovery web service.

If this fails, then CIMPortal Plus's CIMWeb module is getting blocked. CIMPortal Plus might have been installed incorrectly.

2.17.2.6 Verify Remote Access to the CIMWeb Web Services

On the client computer, open an internet browser like Internet Explorer. Enter the E132 URL into the internet browser and verify access. The URL should be something like this:

```
http://192.160.0.10:8080/CIMWeb0710/EquipmentName/E132
or
http://equipmentComputerName:8080/CIMWeb0710/EquipmentName/E132
```

If successful, the internet browser will not show an HTTP error. Run the same test for each of the CIMWeb URLs; E125, E132, E132SA, and E134. If running both 1105 and 0710 version of the web services, then be sure to test both sets of URLs. Also check the E132 Interface Discovery web service.

If this fails, then something is specifically blocking CIMWeb. Please contact your system administrator for assistance.

2.17.2.7 Verify Local Access to the Client Consumer Web Service

On the client's consumer computer, open an internet browser like Internet Explorer. Enter the HTTPEndPoint address using a localhost computer name. For example, ECCE uses IIS with this address:

```
http://localhost:8080/EDAConnect/ConsumerV1105
```

You should see a web page for the consumer.

If this fails, then the client consumer is not configured correctly for its web service. If using ECCE as the client, then the ECCEConsumerService is not installed or configured correctly.

IMPORTANT: Note that in ECCEv2 the client consumer web service is not running until the first time you connect to an equipment with the specified Consumer Location. So this is expected to fail until an attempt to connect is made.

2.17.2.8 Verify Remote Access to the Client Consumer Web Service

On the equipment's CIMWeb computer, open an internet browser like Internet Explorer. Enter the client consumer (HTTPEndPoint) URL into the internet browser and verify access. The URL should be something like the one ECCE uses:

```
http://192.160.0.1/ECCEConsumer/ECCEConsumerService.asmx
```
or
```
http://edaComputerName/ECCEConsumer/ECCEConsumerService.asmx
```

If this fails, then something is specifically blocking access to the consumer web service on the equipment computer which will prevent CIMWeb from sending any data to the client consumer and prevent it from sending a ping to the client consumer. Please contact your system administrator for assistance.

IMPORTANT: Note that in ECCEv2 the client consumer web service is not running until the first time you connect to an equipment with the specified Consumer Location. So this is expected to fail until an attempt to connect is made.

2.17.3 Errors when installing multiple Cimetrix connectivity products

There are known problems when installing multiple Cimetrix products. Notably, in some cases, earlier products released before later products may overwrite files that are common to both products. This may cause a later product to not function correctly. The incorrect behavior may be anything from a minor functional difference to the product crashing.

An example of this is if some releases of CIMPortal Plus are installed first and then earlier versions of CIMConnect are installed. CIMConnect will install common files that are incompatible with the newer CIMPortal Plus.

It is strongly recommended, when multiple Cimetrix products are installed, to install them in date order based on the release date documented at the top of the various release notes, earliest date first.

2.17.4 Reporting Issues

When requesting support for issues in CIMPortal Plus, please include the following information with your request.

2.17.4.1 All Issues:

For all support requests, please include the following information:

1. The symptoms of the issue including what you are seeing and how you are seeing this.
2. The expected behavior.
3. Any screenshots showing the issue.
4. Step by step instructions on how to reproduce the issue. If possible, Cimetrix Customer Support will want to reproduce the issue at Cimetrix so please include everything Cimetrix will need to do so.

5. The SupportTool report.
6. If applicable, Windows System, Application, and Security Event logs.

2.17.4.2 Installation Issues

For installation issues, please include the information for all issues and the following additional information:

1. MSI logs
2. Windows System, Application, and Security Event logs

2.17.4.3 Equipment Model Developer Issues

For issues with developing models and DCIM instances, please include the information for all issues and the following additional information:

1. The equipment xml file.
2. The DCIM instance directories for each DCIM instance used in the equipment model. Cimetrix cannot use the Equipment Model without this.
3. Any error messages.
4. If it is a custom DCIM package, please provide the DCIM package directory for this DCIM.

2.17.4.4 Connection Issues

For issues with deploying or connecting equipment in CIMPortal Plus, please turn on CIMPortal Logging in the configuration manager. The logging to turn on includes, under CIMPortal:<EquipmentName>, all items except threading and performance, under CIMPortal:<EquipmentName>:<DCIMName>, all items except DCIM. Reproduce the issue to capture the issue in the logs and include the information for all issues and the following additional information:

1. Screenshot of the CIMPortal Plus Control Panel CIMPortal Configuration tab.
2. Screenshot of the CIMPortal Plus Control Panel Equipment Status and Settings screen for each equipment.
3. The Package file (.pkg) for the equipment including password.
4. The CIMPortal Plus logs as previously described.

2.17.4.5 CIMStore Issues

For issues with storing data in CIMStore, please turn on CIMStore Logging in the configuration manager. The logging to turn on includes, under CIMStore:<EquipmentName>, all items except threading. Reproduce the issue to capture the issue in the logs and include the information for all issues and the following additional information:

1. The Package file (.pkg) for the equipment including password.
2. The CIMStore logs (located in the CIMPortal Plus log) as previously described .
3. If issue deals with data corruption or loss, MySQL data records showing the issue.

2.17.4.6 ECCE Issues

For issues with the ECCE, please turn on CIMPortal Logging in the configuration manager. The logging to turn on includes, under CIMPortal:<EquipmentName>, all items except threading and performance, under CIMPortal:<EquipmentName>:<DCIMName>, all items except DCIM Threading. Reproduce the issue to capture the issue in the logs and include the information for all issues and the following additional information:

1. The Package file (.pkg) for the equipment including password.
2. The Command History tab, Command Data and Command Response for the data items calls relating to the issue.
3. The CIMPortal Plus logs as previously described.

2.17.4.7 DCIM Issues

For issues with DCIM Applications developed by the customer, please turn on CIMPortal Logging in the configuration manager. The logging to turn on includes, under CIMPortal:<EquipmentName>:<DCIMName>, all items except DCIM. Reproduce the issue to capture the issue in the logs and include the information for all issues and the following additional information:

1. The Package file (.pkg) for the equipment including password.
2. Source code for the application. It does not have to be the entire source code rather it could be an excerpt that demonstrates the issue.
3. The CIMPortal Plus logs as previously described.

3. DCIMs

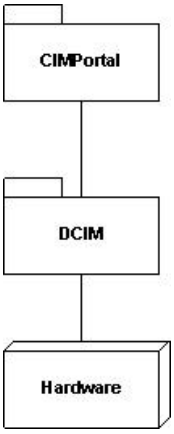
DCIMs are CIMPortal Plus Data Collection Interface Modules. CIMPortal Plus uses DCIMs primarily to gather parameter values, receive event notification and receive exception state changes for the parameters, events and exceptions declared in the equipment models. Before a DCIM Package can be used in an equipment model, one or more DCIM Instances must be defined. CIMPortal Plus creates a unique DCIM object for each DCIM Instance referenced in an equipment model. The same DCIM Package can be referenced by multiple DCIM Instances since the data collection for each DCIM Instance is independent. The DCIM Instance defines the list of handled events, exceptions and parameters. The DCIM Instance also configures communication to the data source, such as the XML file name (XML DCIM) or TCP/IP port configuration (TCP/IP App DCIM and WCF App DCIM). Use the DCIM Administrator window in Equipment Model Developer to define DCIM Instances. Each DCIM Package may be used one or many times, even in the same equipment model.

3.1 DCIM Developer

The CIMPortal Plus DCIM Developer Kit (DDK) aids the equipment supplier in connecting data sources to CIMPortal Plus for data collection through Interface A.

DCIM stands for Data Collection Interface Module. A DCIM is a plug-in for CIMPortal Plus that provides a data source such as DeviceNet, Modbus, or an Analog Input card to CIMPortal Plus.

A DCIM must manage reports defined by CIMPortal Plus and provide the collected data to CIMPortal Plus as well as provide model information to the model developer. This tutorial describes the behavior expected of a CIMPortal Plus DCIM.

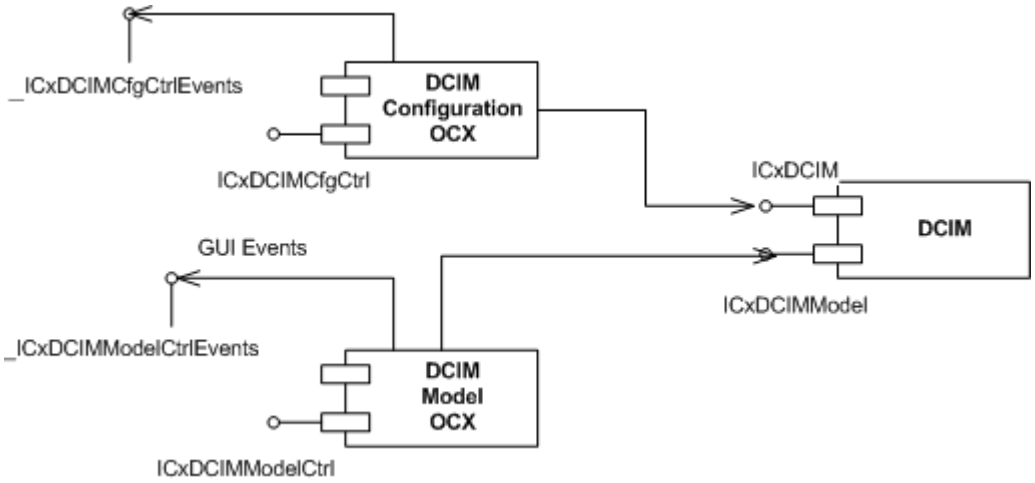


3.1.1 Terminology

- Data Source - A source of data for CIMPortal Plus. This may be software (database, application, etc.) or hardware (DeviceNet, Profibus, PLC etc.).
- DCIM - Data Collection Interface Module. A Data Source plug-in for CIMPortal Plus.
- DCIM Instance - a named association of a DCIM package binary with its required configuration information for a specific data source
- DCPObject - Data Collection Plan Object. An event, exception, or parameter the DCIM can produce and collect.
- Equipment Model Developer (EMDeveloper) - An application provided by Cimetrix to assist the user in creating a model of the equipment.

3.1.2 Architecture

A DCIM is a COM object for performing data collection for CIMPortal Plus. Ideally it is an in-process COM server (DLL) for performance reasons. The different parts of a DCIM package are shown below.



3.1.2.1 Package Items

- The DCIM Configuration OCX is not optional. It is used in several UIs for creating the model and runtime configurations necessary for the DCIM to operate on a given data source. This is a highly customized ActiveX control specific to the data source of the DCIM.
- The DCIM itself is a COM object that performs report management and data collection from a data source.

3.1.2.2 DCIM Interface

The interface between a DCIM and CIMPortal Plus consists of COM interfaces.

3.1.2.2.1 ICxDCIM Interface

ICxDCIM is the main interface implemented by DCIMs that is used by CIMPortal Plus to manage reports.

3.1.2.2.2 ICxDCIMModel Interface

ICxDCIMModel is used by the Equipment Model Developer to get DCIM items to add into the equipment model.

3.1.2.2.3 ICxDCIMCB Interface

ICxDCIMCB is implemented by CIMPortal Plus to receive report data from the DCIM. The DCIM will call methods on ICxDCIMCB to report data.

3.1.3 Design

The main areas of DCIM functionality are listed below with pages describing how the DCIM will be used and its expected behavior. The COM interface calls for each area of functionality are listed and shown in a sequence diagram to better illustrate behavior and usage.

A DCIM package should include Install_XXX.bat and Uninstall_XXX.bat batch file for performing any registration/unregistration necessary for the DCIM package. The batch files, along with the DCIM binary(ies), should be placed in a folder (whose name is the DCIM package name) in the Storage\dcim packages directory. The batch files will be used by EMDeveloper when retrieving a DCIM Instance from the repository for use in creating a model. They will also be used by CIMPortal Plus when deploying a package on a runtime system to register the DCIM.

The Background Information link describes objects and enumerations used by DCIMs

The Cimetrix AppDCIM may be used as a starting point for any DCIM. It will provide plan management and report formatting so the DCIM developer can concentrate on the connection to the data source. At a

minimum, this could be used in a phased development approach and if using the AppDCIM does not meet the performance of the data source, then its functionality could be replaced in a later phase. This would allow the connection to the data source to be developed and mapped into CIMPortal Plus quickly. Also, if the data source provided plan management capabilities (like a GEM connection) then the AppDCIM would not be a good choice.

3.1.3.1 Background

Besides the COM interfaces, DCIMs have dependencies on several objects and enumerations. The enumerations are all documented in the References section.

3.1.3.1.1 CxDCPObject

CxDCPObject is a COM object used to store definitions of the events, exceptions, and parameters the DCIM can produce. These objects are generated by the DCIM and returned through the ICxDCIMModel interface. Later, those objects are passed to the DCIM through the ICxDCIM interface in report definitions.

3.1.3.1.2 CxRequestDefinition

This class contains information for Event, Exception, or Trace requests and is used in the CreateRequest() API.

3.1.3.1.3 CxCPReturnCode

This is a structure used to return error information from many ICxDCIM APIs. The DCIM should return this structure filled in on any error in the API. The structure contains an error code which should be >= 0 for success and < 0 on failure, just like a COM HRESULT. If the error code is negative, then the error string element of the structure is expected to contain a useful description of the error.

3.1.3.1.4 Cimetrix Value Object

The Value object is able to hold any SECS-II value or message structure. It is widely used in Cimetrix communications products such as SECSConnect (E4,E5,E37), CIMConnect(E30), and CIM300(E39,E40,E58,E87,E90,E94,E116). In CIMPortal Plus, the Value object is used to pass parameter values and reports between the DCIM and CIMPortal Plus. Value objects are implemented in the VALUELib COM library. .Net assemblies should reference the Interop.VALUELib.dll to access these types. A complete tutorial and reference for the Value object is available in the Value.chm online help file.

If all the method calls the user wants to make on the Cimetrix libraries only have byte buffer arguments containing the value object contents, and the user is writing a .Net application, then as an alternative the Cimetrix.Value assembly can be used. This is an alternate implementation of the Value object in .Net that has string representations that are compatible with the byte buffer family of API calls. Reference documentation for this can be found in the Common.chm help file.

3.1.3.1.5 Timestamps

All times passed through the DCIM and DCIM callback interfaces will be in UTC format and will be a 64 bit integer as returned in the Windows API GetSystemTimeAsFileTime(). The resolution on most computers is 15.6 milliseconds.

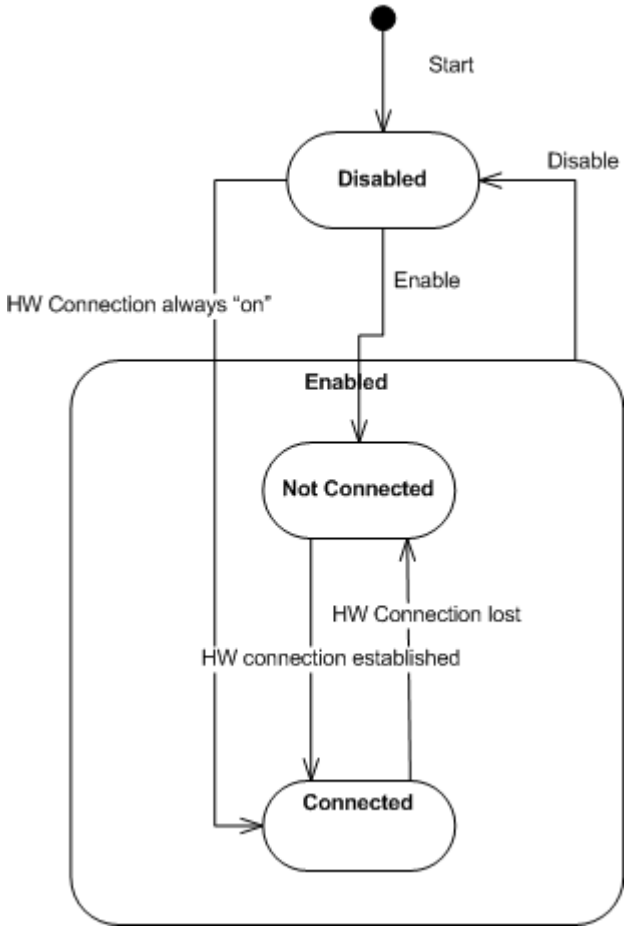
3.1.3.2 DCIM State Machine

The initialization sequence of a DCIM is described here.

DCIMs may be Enabled and Disabled by CIMPortal Plus. This should enable or disable data collection. ICxModel API should be allowed when Disabled. When Disable is called, all traces should be stopped. Cyclic trace definitions should be kept, non-cyclic should be deleted (reported through TraceChange callback). Event and Exception requests should be kept, but disabled. When Enable is called, any defined

requests should be re-enabled. Cyclic traces should be enabled and waiting for Start trigger. This would be the result of a prior Disable call with requests active.

All DCIMs must implement the DCIM State Machine. State changes are reported to CIMPortal Plus through the ICxDCIMCB::StateChange callback.



Old State	Trigger	New State	Description
None	DCIM is started	Disabled	DCIM has been instantiated
Disabled	Enable API is called	Not Connected	DCIM data collection has been enabled and the DCIM is negotiating the HW connection (e.g. GEM S1,F13)
Disabled	Enabled API is called	Connected	DCIM has no hardware connection scheme, it is always able to collect data.
Not Connected	HW connection is established	Connected	The DCIM is now able to collect data.
Connected	HW connection is lost	Not Connected	The HW connection has dropped either due to HW failure or the other party has stopped.
Enabled	Disable API has been called	Disabled	DCIM has been deactivated, no further data reporting will occur. It is not necessary to disconnect from the hardware, but every effort should be made to minimize CPU usage in this state.

3.1.3.2.1 Do this:

1. Initialize the state to Disabled
2. When ICxDCIM::Enable is called begin the connection to the data source and follow the transition table triggers and descriptions above.

3.1.3.3 Initialization

The initialization sequence of a DCIM is described here.

GetCfgCtrlID() should return the ProgID for the DCIM configuration ActiveX control and should work at any time after the DCIM is created. It should not be necessary to call Init() or Enable() in order to call GetCfgCtrlID().

3.1.3.4 Time Synchronization

When a DCIM receives a SynchronizeTime call, it should compute the offset between the masterTime passed in and the current time it has. This offset should be retained and applied to any timestamps reported by the DCIM. Since most DCIMs run on the same PC as CIMPortal Plus, this offset is usually 0. If the DCIM is smart enough to know that it is on the same PC as CIMPortal Plus (it is a DLL), then the offset can always be 0. In the event that the data source supports time synchronization, the DCIM should synchronize the data source clock with the master clock only if it will not affect other collectors. In this case, an offset should be maintained and applied.

3.1.3.4.1 Do this following the sequence diagram below:

1. Create is normal COM object creation, not an API call.
2. Init will give the DCIM the model and runtime configuration string for it to initialize itself and possibly the data source it is connected to. Any configuration files should be parsed and any DCPObject initialization should be performed.
3. When Initialize is called, any active requests should be deactivated and destroyed.
4. The name for this DCIM Instance is also provided in the call and should be saved. The DCIM Instance name should be put into all DCPObjects passed back from through the ICxDCIMModel interface.
5. The DCIM should initialize to the Disabled state.
6. It is possible to distinguish whether the Init() call is made during modeling time or during runtime. If the model configuration string is given but the runtime configuration string is empty or null, it is modeling time. On the other hand, if the runtime configuration string is given but the model configuration string is empty or null, it is runtime.
7. During modeling time, the DCIM is only required to provide a list of available events, exceptions and parameters. It is not required to connect to any underlying hardware or data source for actual data collection. On the other hand, during runtime, not only should the DCIM provides the available events, exceptions and parameters, but also it must be connected to the underlying hardware or data source to be ready for data collection.
8. GetInformation should return the build, vendor, and version information for the DCIM including the build date. This information is logged by CIMPortal Plus on startup.
9. RegisterCB allows DCIM clients to register to receive ICxDCIMCB callbacks. The DCIM should maintain a collection of registered callbacks and call all the DCIM clients for every callback.
10. Enable is used to inform the DCIM to begin data collection and plan management.
11. SynchronizeTime is used to match the DCIM time with CIMPortal Plus time. Do NOT change the system clock of the PC that CIMPortal Plus is running on. If it is possible to synchronize time in the data source the DCIM is connected to, then the DCIM should do so.
12. Once the DCIM has been fully initialized (either for modeling or for runtime), report the StateChange.

3.1.3.5 Shutdown

The initialization sequence of a DCIM is described here.

3.1.3.5.1 Do this:

1. On Disable, perform the correct actions per the DCIM State Machine explanation.
2. For UnregisterCB, remove the callback from your collection and release the COM interface.
3. On Shutdown, release any allocated memory or resources and disconnect from the data source.

3.1.3.5.2 Verify

Verify your DCIM releases all its resources and memory as well as disconnects from its data source correctly.

3.1.3.6 Properties

A DCIM must support the following properties:

- LoggingLevel

  LoggingLevel is used to adjust the amount of information reported by the DCIM.
- DCIMInstanceName

  DCIMInstanceName uniquely identifies the DCIM object instance and is reported in every DCPObject the DCIM returns through the ICxDCIMModel interface.

3.1.3.7 Logging

Logging is used to test and troubleshoot the DCIM. Anything the DCIM developer deems important or useful in troubleshooting should be included in logging.

3.1.3.7.1 COM Interface Methods

- ICxDCIM::put_LoggingLevel, get_LoggingLevel
- ICxDCIMCB::Log

LoggingLevel is used to adjust the amount of information reported by the DCIM. The DCIM should filter the information reported through the ICxDCIMCB::Log callback based on the value of LoggingLevel. For best performance, it is a good idea to check the logging level prior to generating logging information and calling the Log callback even though this may lead to duplicated code.

The values for the logging level are in the CxDCIMLogLevel enumeration:

Logging levels may be combined.

3.1.3.8 Modeling

Modeling is representing the data source the DCIM is connected to using Events, Exceptions, and Parameters. This Modeling will vary depending on the data source technology. The DCIM may need logic to convert a data value into an event or exception. For example, maybe the DCIM is monitoring a temperature and produces that value as a Parameter. In addition, the DCIM may produce an Exception when the temperature value exceeds a certain value. Events may require similar logic.

The DCIM exposes these Events, Exceptions, and Parameters using CxDCPObjects. The DCPObjects are identified by name and the DCIMInstance that produces them. The DCIM should use the attrs property of the DCPObject to store any information it may need to collect the item such as an ID, an address, channel, etc.

The Equipment Model Developer application (EMDeveloper.exe) uses the ICxDCIMModel interface to get DCPObject definitions from the DCIM. The model developer will then include the DCIM

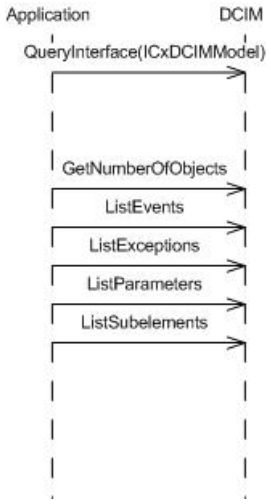
DCPObjects into the equipment model to allow CIMPortal Plus clients to use them for data collection. CIMPortal Plus will return the DCPObject name, DCIMInstanceName, and attrs when it uses the DCPObject in a report definition.

The ICxModel interface API should be allowed when the DCIM is in the Disabled state.

3.1.3.8.1 Do This:

1. Decide what parameters, events, and exceptiond the DCIM will produce.
2. Implement the ICxDCIMModel interface to return the DCPObjects the DCIM will produce.

3.1.3.9 SEMI Objects



SEMI Objects are SEMI E39 Object Services derived object such as the E87 Carrier Management Carrier object and E94 Control Job Manamgement ControlJob object. These objects are normally accessed through Stream 14 messages on the GEM connection to the tool. Because of the dynamic nature of these objects, it is not possible to identify most object instances in the equipment model.

The E125 standard defines a web service method GetObjInstanceIds to retrieve the list of object ids (ObjID) given an object type (ObjType). When CIMPortal Plus receives this web service request, it calls GetObjectInstanceIds on the DCIMs to get their object instances. CIMPortal Plus's client may then request parameters which are attributes of those objects by creating specially formatted parameter names. CIMPortal Plus will then request those parameters from the DCIM using CxDCPObjects of type cxdtE39Attr.

When defining reports using E39 Object instance attributes, CIMPortal Plus will create CxDCPObjects with the DCPType of cxdtE39Attr and the name will be formatted as: "urn:semi-org:objInst:<typeName>:<objId>:objAttr:<parameterName>". <typeName> comes from the model SEMIObjType elements. <objId> is returned by the DCIM in the GetObjInstanceIds() API. <parameterName> is the name of a Parameter defined in the model SEMIObjType Attributes element. For instance, to get the PRJobState attribute of the E40 ProcessJob object named ProcessJob1, the cxdtE39Attr CxDCPObject name would be "urn:semi-org:objInst:ProcessJob:ProcessJob1:objAttr:PRJobState".

If an object does not support SEMI objects, return an empty list and S_OK.

In the GetObjectInstanceIds API, objType should be a SEMI standard object type. objType may be an empty string to signify get all objects of all types. If the sourceID parameter is an empty string, then instances for all sources of objects should be reported. Otherwise, only the instances at the given sourceID should be reported. objIdList is formatted as:

```
L,n n= #object types
  1. L,2
    1. A objType
    2. L m #objects of the type
      1. A objID
```

For the above, you would have one list of two items for every unique objType.

3.1.3.10 Job List

E134 defines a web service for reporting performance issues. This service requires the current running jobs. To get this data, CIMPortal Plus calls the GetJobList API on each DCIM. Most DCIMs will not implement this method and should return empty arrays and S_OK for the method.

Return VT_EMPTY if no job list can be produced. VARIANTs are of type SAFEARRAY of BSTR. jobTypes should be one of:

- urn:semi-org:E94:ControlJob
- urn:semi-org:E40:ProcessJob
- urn:semi-org:E30:ppSelect
- urn:semi-org:E30:rcpSelect

3.1.3.11 Request Management

Event, Exception, and Trace request definitions must be managed by the DCIM.

3.1.3.11.1 Event Requests

An Event Request is a set of parameters with a set of events. The DCIM should monitor the events and if any of them happen, it should collect the values of the parameters and return them in the EventReport callback with the original request ID and the event DCPObject that was triggered. The Event Request should be maintained until DeleteRequest is called. The EventReportByteStream callback should be called if the DCIM is not in the CIMPortal Plus process. The DCIM should validate that the condition parameters have the appropriate values before sending the event report.

3.1.3.11.2 Exception Requests

An Exception Request is a set of parameters with a set of exceptions. The DCIM should monitor the exceptions and if any of them change to the state specified in the request, it should collect the values of the parameters and return them in the ExceptionReport callback with the original request ID and the exception DCPObject that was triggered and its state. The Exception Request should be maintained until DeleteRequest is called. Each exception in the Exception Request may specify a state in CxExceptionState. If the state specified is cxesClear, then a report should be generated when the exception goes to the clear state. Likewise for cxesSet. If the state is cxesEither, then a report should be sent any time the exception changes state. The ExceptionReportByteStream callback should be called if the DCIM is not in the CIMPortal Plus process.

When a Data Collection Plan with Exception request is created by an Interface A client, the CIMWeb actually creates two exception requests for each Exception in the DCP - one with Set Data parameters and one with Clear Data. It is done this way because the Exceptions may specify different parameters to be collected when Exception is Set vs. when its Cleared.

3.1.3.11.3 Trace Requests

A trace is periodic data collection. A trace request contains a list of parameters to be collected, the interval at which to collect them, and the number of samples to collect.

If a trace is defined without start triggers, it should wait until StartTrace is called before collecting data. The first sample should be collected upon the StartTrace call or a start trigger.

A trace without parameters is allowed if it has one or more of either start or stop triggers. The DCIM should monitor the triggers and report the TraceChange with started or stopped, but should not send any reports.

If the number of samples to collect is 0, then the trace runs indefinitely, until DeleteRequest is called, until StopTrace is called, or until an optional stop trigger happens.

In addition, optional sets of start and stop triggers may be specified. A trace with start triggers waits until the trigger occurs, and then begins sampling data. A trace with stop triggers stops collecting data when the trigger occurs or when the number of samples is fulfilled. If exceptions are used as triggers, the state of the exception may be specified. If the state is specified as set or clear, then the trigger is only acted upon when the exception state matches the state specified in the trigger. If cxesEither (no state specified) is used in the trigger, then any time the exception happens the trigger is acted upon.

A trace may be marked as Cyclic. When the number of samples or a stop trigger occurs, a cyclic trace ceases data collection, but remains defined waiting for a start trigger or StartTrace. Cyclic traces with no start or stop triggers are allowed. Their activation is governed by StartTrace and StopTrace.

If an event is used as a trigger, then the occurrence of that event is used as the trigger. An event is stateless and the trigger is only when the event occurs. If the event is synthesized from an equation, the event would only be the first time the expression evaluates to a "set" state. The trigger would not happen again until the expression evaluates to "clear" and back to "set". Any conditions parameters must have the appropriate values before triggering a state change.

If an exception is used as a trigger, then the change in state of that exception may be used as a trigger. An exception has set and clear states. If the exception is synthesized from an equation, the trigger would happen when the evaluation of that equation changes the exception state.

As soon as a trace starts, either via StartTrace or a start trigger, the DCIM should call a TraceChange callback with cxtsCollecting.

Trace reports should be sent using the TraceReport callback when in-process and the TraceReportByteStream callback when not in the CIMPortal Plus process.

3.1.3.11.3.1 Trace without Triggers

A trace without any triggers waits for a StartTrace call to begin data collection. As soon as StartTrace is called, the DCIM should collect the parameters and return the values in a TraceReport callback with the collection number. The DCIM should also start a timer from which to perform subsequent periodic collection. The trace will continue to run until it has collected the required number of samples. At that time if the trace is not Cyclic, it will terminate collection and be deleted by the DCIM.

3.1.3.11.3.2 Trace with Start triggers

A trace with a start trigger will wait until the trigger occurs or until StartTrace is called, then collect a data sample and start a timer.

3.1.3.11.3.3 Trace with Stop triggers

A trace with a stop trigger will stop data collection as soon as the sample count is fulfilled, the StopTrace method is called, the DeleteRequest method is called, or one of the stop triggers occurs.

3.1.3.11.4 Immediate data requests

RequestData or RequestDataByteStream may be called at any time to synchronously request values for a set of parameters.

3.1.3.12 Data Reporting

Expected behavior for data reporting is outlined below. Use the Data Values table to determine the correct Cimetrix Value object data type for a given Interface A base type.

3.1.3.12.1 Do this:

1. Do not report any data when the DCIM is Disabled.
2. When an Event occurs:
   a. Check for reports defined for the event.
   b. If the DCP event in the event report or trace report has conditions, get the values of the condition parameters and validate that the current parameter values are the same as the condition values. If they are not the same, disregard the event for that DCP event only.
   c. Collect the data for the reports.
   d. Generate a timestamp for the report.

e. Post the reports using the EventReport(in-process) or EventReportByteStream(out-of-process) callback for each report.

f. See if the event is a start or stop trigger for any traces and start or stop traces as appropriate.

3. When an Exception occurs:
   a. Check for reports defined for the exception and its current state (set or clear).
   b. Collect the data for the reports.
   c. Generate a timestamp for the report.
   d. Post the reports using the ExceptionReport(in-process) or ExceptionReportByteStream(out-of-process) callback for each report.
   e. See if the exception is a start or stop trigger for any traces and start or stop traces as appropriate.
4. When a Trace timer expires:
   a. Generate a timestamp for the data collection.
   b. Collect the data for the trace.
   c. Report the trace data using TraceReport(in-process) or TraceReportByteStream(out-of-process).
   d. Increment the number of samples reported.
   e. If the number of samples collected is equal to the number of samples to collect:
      i. Stop the trace timer.
      ii. Call the TraceChange(in-process) or TraceChangeString(out-of-process) callback with state cxtsStopped.
      iii. If the trace is not cyclic:
         (1) Delete the trace.
         (2) Call the TraceChange(in-process) or TraceChangeString(out-of-process) callback with state cxtsDeleted.

3.1.4 Coding

The Cimetrix AppDCIM may be used as a starting point for any DCIM. It will provide plan management and report formatting so the DCIM developer can concentrate on the connection to the data source. At a minimum, this could be used in a phased development approach and if using the AppDCIM does not meet the performance of the data source, then its functionality could be replaced in a later phase. This would allow the connection to the data source to be developed and mapped into CIMPortal Plus quickly.

Thorough performance testing should be done on this implementation approach to be sure it meets the performance of the data source. As always, a general solution (using the AppDCIM) will be slower than a custom solution.

3.1.4.1 Do This (for in-process DLL DCIMs):

1. Begin a new COM DLL project in your development environment.
2. Add a new COM object to your project as your DCIM object.

3.1.4.1.1 Using the AppDCIM

1. Aggregate the AppDCIM following the instructions in the AppDCIM documentation.
2. Use the AppDCIM ICxAppDCIM interface when data changes, events happen, or exceptions change state.

3.1.4.1.2 Custom Solution

1. Implement the ICxDCIM interface.
2. Implement the ICxDCIMModel interface.

3. Implement the functionality defined in the design section.

3.1.4.2 Do This (for out-of-process EXE DCIMs):

1. Add a new COM object to your project as your DCIM object.

3.1.4.2.1 Using the AppDCIM

1. Implement the ICxAppDCIMCB interface on your COM object
2. Get the AppDCIM instance from CIMPortal Plus using ICxCPEstablishedSession::GetDCIM()
3. Register your object with the AppDCIM for callbacks using ICxAppDCIM::RegisterCB()
4. Use the ICxAppDCIM AppDCIM interface when data changes, events happen, or exceptions change state.

3.1.4.2.2 Custom Solution

1. Implement the ICxDCIM interface.
2. Implement the ICxDCIMModel interface.
3. Implement the functionality defined in the design section.

3.1.5 Testing

DCIMs may be tested within CIMPortal Plus in a normal deployed environment.

1. Develop a model that uses the DCIM for parameters, events, or exceptions.
2. Use CIMStore or an Interface A Client application such as ECCE or EDAConnect to define plans and collect data

3.1.6 Debugging in CIMPortal Plus

3.1.6.1 EXE DCIMs

1. Stop CxCIMPortalSvc using either the Control Panel or the Windows Services applet
2. On a command line in the \Program Files\Cimetrix\Comm Products\bin directory, execute CxCIMPortalSvc /regserver. This will remove CxCIMPortalSvc as a service and run it as a COM Local Server. This will allow you to debug with it as it will now run under your user account instead of the Windows System account.
3. Go to the Windows Control Panel->Administrative Tools->Component Services applet. Under Computers\My Computer\DCOM Config check the Properties\Security for Cimetrix CxCIMPortalSvc and be sure ASP.NET Machine Account has both Launch and Access permissions.
4. Start your application in Debug mode in Visual Studio
5. Start CxCIMPortalSvc
6. When you are done and want to run CxCIMPortalSvc as a Windows Service again, execute CxCIMPortalSvc /service on a command line.

3.1.6.2 DLL DCIMs

1. Stop CxCIMPortalSvc using either the Control Panel or the Windows Services applet
2. On a command line in the \Program Files\Cimetrix\Comm Products\bin directory, execute CxCIMPortalSvc /regserver. This will remove CxCIMPortalSvc as a service and run it as a COM Local Server. This will allow you to debug with it as it will now run under your user account instead of the Windows System account.

3. Go to the Windows Control Panel->Administrative Tools->Component Services applet. Under Computers\My Computer\DCOM Config check the Properties\Security for Cimetrix CxCIMPortalSvc and be sure ASP.NET Machine Account has both Launch and Access permissions. Also make sure the Identity tab has The Interactive User radio button selected, not The Launching User.
4. Specify CxCIMPortal PlusSvc.exe as the executable for your debugging session in Visual Studio
5. Register the dll in your Debug directory so CxCIMPortalSvc doesn't try to use the dll in the deployment package. Use regsvr32 <yourdcim>.dll.
6. Begin debugging in Visual Studio
7. When you are done and want to run CxCIMPortalSvc as a Windows Service again, execute CxCIMPortalSvc /service on a command line.

3.1.6.3 Verify

Place a breakpoint in your DCIM object ICxDCIM::Init method and verify it is called.

3.2 Standard DCIM Packages

Following is a summary of the DCIM Packages distributed with CIMPortal Plus.

- CIMConnect DCIM -- for collecting data directly from CIMConnect by setting up the DCIM as an EDA host. This takes advantage of CIMConnect's multiple-host functionality.
- CIMConnect SR13+ DCIM -- for collecting data directly from CIMConnect (version SR13 and above). Unlike older CIMConnect DCIM it does not require an extra connection setup in the .EPJ file for use and can collect data from any connection. New DCIM also does not use CxEDATask.dll.
- App DCIM -- for collecting data from a custom Windows DLL, from a Windows executable on the same computer or from a Windows executable on the same network. The App DCIM also serves as a base DCIM for creating custom DCIMs.
- PerfMon DCIM -- for collecting data from Windows performance counters on the same computer or a Windows computer on the same network
- TCP/IP Extender, AppDCIM interface -- extends the Application DCIM Interface for collecting data from non-Windows platforms using TCP/IP
- XMLFile -- for collecting data from an XML file using XPath expressions. The XML file can be on the same computer or network.
- WCF Extender, AppDCIM interface -- extends the Application DCIM Interface for collecting data using Windows Communication Foundation (WCF)

3.3 Custom DCIM Packages

A custom DCIM package must implement the ICxDCIM and ICxDCIMModel COM interfaces. The easiest way to create a custom DCIM package is by using the App DCIM. This can be done from an executable or from a DLL. A DLL implementation generally performs better. For each method in the DCIM interface, either use the App DCIM implementation or overwrite the implementation with custom software. Check the Cimetrix support web site for source code to sample DCIM implementations.

3.4 DCIM Responsibilities

Each DCIM instance must fulfill the required responsibilities. A DCIM optionally may also implement other features

3.4.1 Required

- List of handled Parameters, Events and Exceptions
- Handle at least one parameter, event or exception.

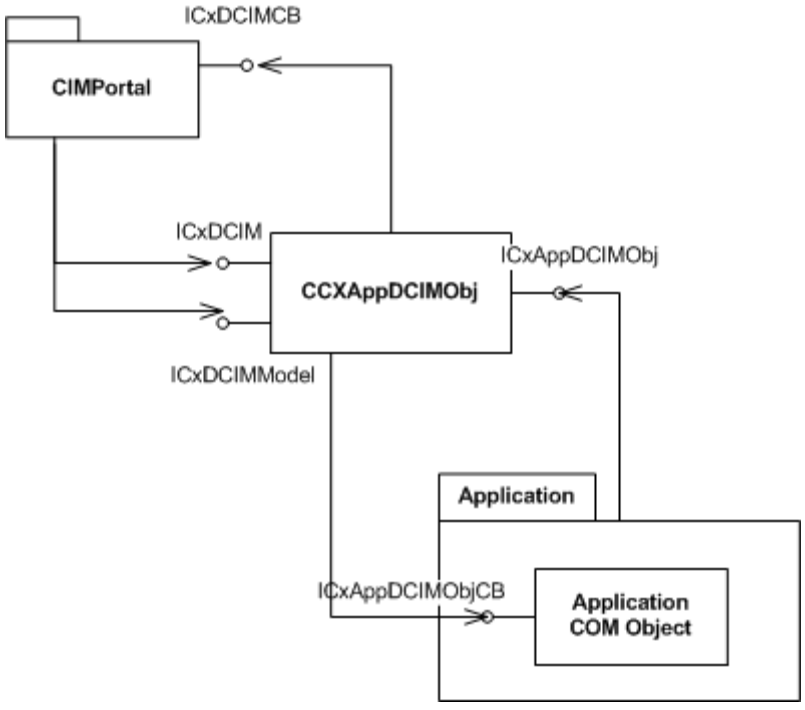
3.4.2 Optional

- Provide values for Parameters
- Trigger events, with context data
- Trigger exceptions, with context data
- Performance Warnings
- E39 Data, object IDs and attribute values (SEMIObjTypes)
- List of active jobs
- Quality of Service (time stamping)
- Logging
- Initialization & Shutdown method

3.5 App DCIM Overview

The Cimetrix App DCIM is a plug-in to CIMPortal Plus used to collect data from any Application or from any data source. The App DCIM provides plan management and report formatting so the DCIM developer can concentrate on the connection to the data source. There are two fundamentally different ways to use the App DCIM. An executable can use its API functions directly if queried from the CIMPortal Plus Service. It can also be used as a starting point in creating a custom DCIM. Both techniques are described below.

The App DCIM uses an ADC text file to define the set of parameters, events and exceptions that will be available in the DCIM instance. Although there is a simple ADC editor available through EM Developer when configuring the DCIM, another ADC Text Editor is also available with advanced features. See ADC Text Editor for details. After a DCIM instance has been created, be sure to edit the copy of the ADC file that is located in the DCIM instance folder.



3.5.1 Using an Application DCIM Instance in the CIMPortal Plus service

This is the most common use of the Application DCIM. In this technique, the App DCIM object is created by the CIMPortal Plus Service during startup or after the PKG file is deployed. The data collection application that wishes to use the App DCIM, must use the ICxCPEstablishedSession CIMPortal Plus COM interface to get the CxAppDCIM COM interface pointer to the object that was created by CIMPortal Plus.

The data collection application is responsible for creating a COM object that implements the ICxAppDCIMObjCB interface to receive callbacks from the CxAppDCIM object.

There are several samples included with CIMPortal Plus that demonstrate how to use the Application DCIM using this strategy, including Application DCIM Demo VC6 , ApplicationDCIMDemoCSharp and ApplicationDCIMDataTypeTest. Here is a summary of the basic steps.

1. Create DCIM Instance using the App DCIM Package in EMDeveloper
2. Implement the ICxAppDCIMObjCB COM interface in a COM object in your application
3. In your application, connect to the CIMPortal Plus service using the ICxCPService COM interface and get an ICxCPEstablishedSession interface.
4. Use the GetDCIM or ListDCIMs API to get the interface pointer to the CxAppDCIMObj COM object using the predefined DCIM Instance name
5. Use the AppDCIM as described in Application DCIM Reference documentation

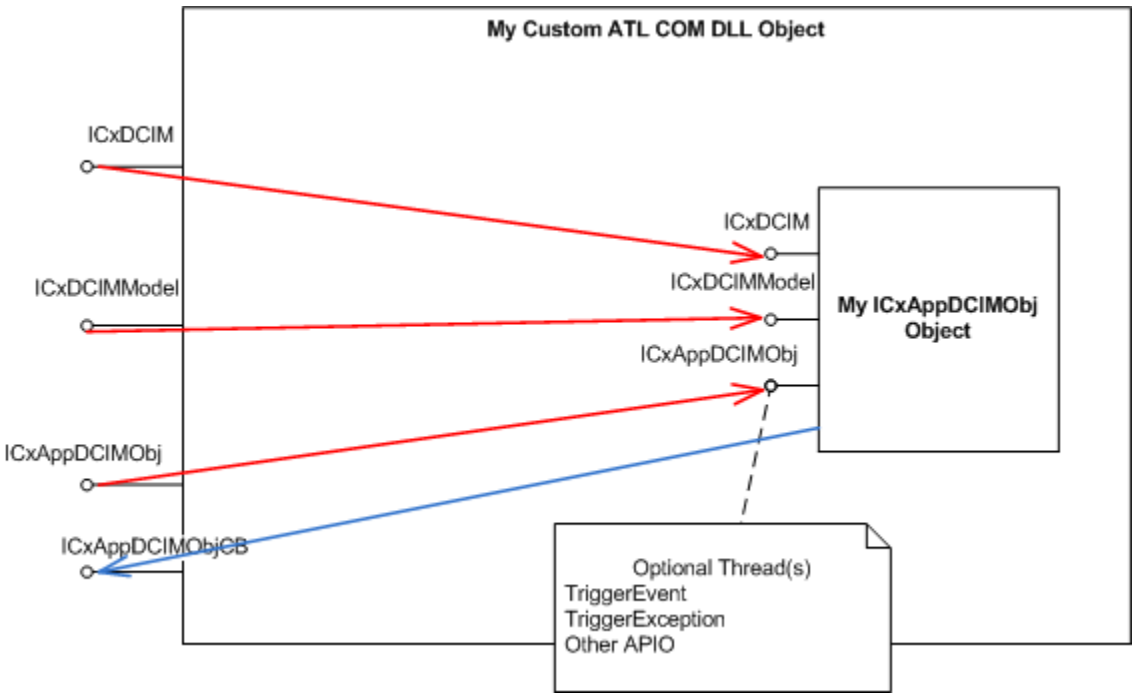
3.5.2 Custom DCIM using the Application DCIM

The Application DCIM provides the quickest way to create a custom DCIM. This can be done using COM aggregation or containment. The COM object must be creatable from another process using COM/DCOM technology.

1. Add new COM object to your project
2. Implement ICxAppDCIMObjCB interface on that COM object

- Containment

Implement the ICxDCIM, ICxDCIMModel and ICxAppDCIMObj interfaces. Create a local ICxAppDCIMObj COM object and use it to implement the ICxDCIM, ICxDCIMModel, and ICxAppDCIMObj interfaces.



- Aggregation

Aggregate the CCxAppDCIMObj COM object and expose its ICxDCIM, ICxDCIMModel and ICxAppDCIMObj interfaces. The COM object must be created in MTA (Multi-Threaded Apartment) because the CCxAppDCIMObj object is MTA and has to be in the same COM apartment as the aggregating object. The ProgID of the custom Application COM object must be used in Equipment Model Developer to create a DCIM Package.

3. Use ICxAppDCIMObj interface to update Data, trigger Events and Exceptions in response to changes on Equipment hardware or software
4. See either sample (for containment) ApplicationDCIMDemoDLLVC6 or InprocDCIMAggregation .

3.5.3 AppDCIM performance tuning

Create DCIM as a DLL whenever possible. When a DCIM is loaded by CIMPortal Plus service into its own process space, the overhead of passing data and making calls across process boundaries is greatly reduced, freeing up the CPU and lowering latency. If it is not possible to create DCIM as DLL, consider using the second method of using AppDCIM - 'Using an Application DCIM Instance in the CIMPortal Plus service', especially when majority of the parameters don't change values frequently and the UpdateData method can be used. In this case, the COM calls across the process boundaries have to happen only when the data changes and CIMPortal Plus service will be able to get to the data faster.

The AppDCIM interface have two versions of functions for passing ICxValueDisp objects - passing the COM interface pointers or serialized byte stream. When the DCIM is a DLL loaded by CIMPortal Plus service into its own process space, the regular versions should be used, otherwise - the ByteStream versions.

The usage of UpdateData/UpdateDataByteStream should be favored for parameters which values do not frequently change. The GetData callback mechanism should be used for frequently changing parameters.

3.5.4 COM ProgID

The COM ProgID for the App DCIM is `CxAppDCIM.CxAppDCIMObj`.

3.5.5 Deliverables

- CxAppDCIM.dll - Application DCIM

3.5.6 AppDCIM Programming Overview

This section contains documents describing fundamental steps to utilize and implement AppDCIM in various IDE.

There are various design and methods to implement AppDCIM. The procedure described in this section is "Using Application DCIM instance in the CIMPortal Plus service" mentioned in AppDCIM Overview.

3.5.6.1 Implement AppDCIM with Visual Studio .NET 2005 in C#

This section describes fundamental steps to implement AppDCIM in Visual Studio .NET 2005 (C#). Also, "Using an Application DCIM Instance in the CIMPortal service" in AppDCIM Overview contains the short implementation description.

A completed project (ExeAppDCIMCSharp) may be found in the Examples folder.

1. Using EMDeveloper, create an equipment model with an Application DCIM instance. Then, create a deployment package and deploy it. The sample PKG file (ExampleAppDCIM.pkg, password: 1234) is available with ExeAppDCIMCSharp project in the Example folder.
2. Start Visual Studio .NET 2005. From the desktop taskbar Start menu, select Microsoft Visual Studio 2005. (The default language setting is Visual C#).
3. Create a new project. Click the File menu, point to New, and then click Project. Select Visual C# for Project types and Windows Application in New Project dialog box. Also, specify the project name (ExeAppDCIMCSharp) in Name box and an appropriate path in Location box. Click OK button.

4. Add references to COM type libraries. Click on References in the Solution Explorer, and select Add Reference… When the Add Reference window comes up select the COM tab. This will list all Component Names which are available on your machine. Select "Cimetrix CxAppDCIM 1.0 Type Library", "Cimetrix CxCIMPortalSvc 1.0 Type Library", and "Cimetrix Value 1.0 Type Library". Then, click OK button.
5. Click on ExeAppDCIMCSharp in the Solution Explorer, point to Add, and click Class. The Add New Item window comes up. Select Class from Templates and type a class name (MyCIMPortal) in Name box. Then, click Add button.
6. Implement ICxAppDCIMObjCB COM interface in the newly created class. After "class MyCIMPortal", type " : ICxAppDCIMObjCB".

```
using System;
using System.Collections.Generic;
using System.Text;
namespace ExeAppDCIMCSharp
{
  class MyCIMPortal: ICxAppDCIMObjCB
  {
  }
}
```

Bring the cursor over "ICxAppDCIMObjCB" and click it. The red smart tag comes up below "B" of "ICxAppDCIMObjCB" Click the smart tag for "Options to help bind the selected item). It will list 2 choices: using CxAppDCIMLib; and CxAppDCIMLib.ICxAppDCIMObjCB. Select "using CxAppDCIMLib;"

```
using System;
using System.Collections.Generic;
using System.Text;
using CxAppDCIMLib;
namespace ExeAppDCIMCSharp
{
  class MyCIMPortal: ICxAppDCIMObjCB
  {
  }
}
```

Bring the cursor over "ICxAppDCIMObjCB" and click it. The blue smart tag is displayed below the first "I" of "ICxAppDCIMObjCB" Hover over this blue smart tag for "Options to implement ICxAppDCIMObjCB" and click it. It will list 2 choices, "Implement interface 'CxAppDCIMLib'" and "Explicitly implement 'ICxAppDCIMObjCB'" Select "Implement interface 'ICxAppDCIMObjCB'". It will generate method stubs for ICxAppDCIMObjCB interface's methods.

Add the following variables to MyCIMPortal class, below "#region Member Variables".

```
private CxCIMPortalSvcLib.CxCPService m_CxCPService = null;
```

```
private CxCIMPortal PlusSvcLib.CxCPEstablishedSession
m_CxCPEstablishedSession = null;

private DCIMInterfaceLib.ICxDCIM m_ICxDCIM = null;

private CxAppDCIMLib.ICxAppDCIMObj m_ICxAppDCIMObj = null;

private DCIMInterfaceLib.CxCPReturnCode m_CxCPReturnCode = new
DCIMInterfaceLib.CxCPReturnCode();
```

7. Connect to the CIMPortal Plus service and get an ICxCPEstablishedSession interface

   At first, hard code the name of equipment, client, and password. These settings should be predefined. Without this information, you cannot establish a connection to the deployed equipment.

   Add a public method (Initialize) to MyCIMPortalclass.

```
public void Initialize()

{

}
```

   The equipment model is the name of the deployed equipment (ExampleAppDCIM). A client (MyClient) is the one defined by Configuration Manager. It is the one with an access right to the deployed equipment (CIMPortalExampleAppDCIM).

   Add following logics to the newly created function.

```
string equipmentName = "ExampleAppDCIM";

string clientName = "MyClient";

string clientPassword = "1234";

string applicationDCIMInstance = "AppDCIMInstance"
```

   Create the CxCPService COM object and get the ICxCPService interface and call the EstablishSession function to establish a session to the deployed equipment.

```
m_CxCPService = new CxCIMPortalSvcLib.CxCPService();

m_CxCPEstablishedSession = m_CxCPService.EstablishSession (equipmentName,
clientName, clientPassword, ref m_CxCPReturnCode);
```

   Then use the ICxCPEstablishedSession pointer and get an ICxDCIM pointer by calling GetDCIM function. Use the predefined Application DCIM instance name "AppDCIMInstance" for the first argument for the GetDCIM function.

```
m_ICxDCIM =
m_CxCPEstablishedSession.GetDCIM(applicationDCIMInstance,
ref m_CxCPReturnCode);
```

   In order to use callback functions, a callback object has to be registered. First, create a callback COM object. Then call, RegisterCB function to register it.

```
m_ICxAppDCIMObj = (CxAppDCIMLib.ICxAppDCIMObj)m_ICxDCIM;

m_ICxAppDCIMObj.RegisterCB(this, true);
```

8. Add your appropriate logic to the ICxAppDCIMObjCB::GetDataByteStream function in MyCIMPortal class. Return 'true' from the function if you are returning data. From the Program class, call the Initialize function.

```
static void Main()
{
  MyCIMPortalmyCP = new MyCIMPortal();
  myCP.Initialize();

  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new Form1());
}
```

9. Add logic to update values into the cache or trigger events and exceptions, by calling other API functions (UpdateDataByteStream, TriggerEventByteStream, TriggerExceptionByteStream, etc).

3.5.6.2 Implement AppDCIM with Visual Studio 6.0 in C++

This section contains documents describing fundamental steps to implement AppDCIM in Visual Studio 6.0 (C++). Also, "Using an Application DCIM Instance in the CIMPortal service" in AppDCIM Overview contains the short implementation description.

There is a completed project (ExeAppDCIM) in the Examples folder.

1. Using EMDeveloper, create equipment model with Application DCIM instance. Then, create a deployment and deploy it. The sample PKG file (ExampleAppDCIM.pkg, password: 1234) is available with ExeAppDCIM project in the Example folder.
2. Start Visual C++. From the desktop taskbar Start menu, select Microsoft Visual C++ 6.0.
3. Create a new project. Select New from the File menu to create a new project.
4. Select the type of project you want to create. Click the Projects tab in the New dialog box. Select MFC AppWizard (exe) from the list of project types.
5. Specify the project name in the Project Name box in the New dialog box. Type "ExeAppDCIM" in the Project Name box. If you want to place your project files in a specific directory, entry an appropriate path in the Location box.
6. Select Dialog Based radio button. (The type of application does not affect Application DCIM implementation.) Click the Finish button.
7. Click OK on the New Project Information dialog box.
8. In order to use CIMPortal Plus COM objects, include the following import directive in stdafx.h file.

   // For using CIMPortal Plus COM objects

```
#import "CxValue.dll" raw interfaces only, raw native types, no namespace,
named_guids

#import "CxDCIMInterface.dll" raw_interfaces_only, raw_native_types,
no_namespace, named_guids

#import "CxAppDCIM.dll" raw_interfaces_only, raw_native_types,
no_namespace, named_guids
```

```
#import "CxCIMPortalSvc.exe" raw_interfaces_only, raw_native_types,
no_namespace, named_guids
```

If you want to check the available interfaces in each DLLs, use OLE/COM Object Viewer under Tools.

9. Select Class View tab of the project workspace pane. Select the topmost ExeAppDCIM classes and display the context menu by right-clicking. From the context menu, select New ATL Object… By doing this, it will add New ATL Object Wizard functionality.

   ATL Object Wizard message asking you to add ATL support to your MFC project. Select Yes button.

   ATL Object Wizard message warns errors. Click OK and ignore the error.

10. Again, select the topmost ExeAppDCIM classes and display the context menu by right-clicking. From the context menu, select New ATL Object...

    This time, you will see ATL Object Wizard dialog box. Select Objects from Category and Simple Object from Objects. Then, click Next button.

    Specify a name of your callback in the ATL Object Wizard Properties dialog box. Type "MyAppDCIMCB" in Short Name box of Names tab. Then click OK button. It generates CMayAppDCIMCB class with IMyAppDCIMCB interface.

11. Implement ICxAppDCIMObjCB COM interface in a COM object in your project. Select newly created CMyAppDCIMCB and display the context menu by right-clicking. From the context menu, select Implement Interface...

    Click Add Typelib… button in the Implement Interface dialog box to select appropriate type library. Scroll down until you see "Cimetrix CxAppDCIM 1.0 Type Library(1.0)" and click on it. Then, click OK button.

    You are back to the Implement Interface dialog box. A new CxAppDCIMLib tab with available interfaces. Click "ICxApDCIMObjCB" to include callback functions, and click OK button.

12. Declare necessary COM smart pointers and Interface pointers in CExeAppDCIMDlg class as private member variables. Open ExeAppDCIMDlg.h file and add the following member variables.

    ```
    ICxCPServicePtr m_pICxCPService;
    ```

    ```
    ICxCPEstablishedSessionPtr m_ICxCPEstablishedSessionPtr;
    ICxDCIMPtr m_ICxDCIMPtr;
    IMyAppDCIMCallbacks* m_pIMyAppDCIMCallbacks;
    ICxAppDCIMObjCBPtr m_ICxAppDCIMObjCBPtr;
    ICxAppDCIMObjPtr m_ICxAppDCIMObjPtr;
    ICxCPAdminPtr m_ICxCPAdminPtr;
    ```

13. Connect to the CIMPortal Plus service and get an ICxCPEstablishedSession interface.

    At first, hard code the name of equipment, client, and password. These settings should be predefined. Without this information, you cannot establish a connection to the deployed equipment.

The equipment model is the name of the deployed equipment (ExampleAppDCIM). A client (MyClient) is the one defined by Configuration Manager. It is the one with an access right to the deployed equipment (CIMPortalExampleAppDCIM).

Type the following codes in CExeAppDCIMDlg::OnInitDalog function below: "// TODO: Add extra initialization here" line.

```
// TODO: Add extra initialization here
_bstr_t bstrEQName = _bstr_t("ExampleAppDCIM");

// Assumes the hard-coded client name and password are configured
correctly in
// the Cimetrix Configuration Manager
_bstr_t bstrClientName = _bstr_t("MyClient");
_bstr_t bstrPasswords = _bstr_t("1234");
```

Create CxCPService COM object and get ICxCPService interface and call EstablishSession function to establish a session to the deployed equipment.

```
// At first get CIMPortal Plus Server
HRESULT hr = CoCreateInstance(CLSID_CxCPService, NULL, CLSCTX_ALL,
IID_ICxCPService, (void**)&m_pICxCPService);

CxCPReturnCode myCxCPReturnCode = {0};

// Access the equipment's session in CIMPortal Plus
hr = m_pICxCPService->EstablishSession( bstrEQName, bstrClientName,
bstrPasswords, &myCxCPReturnCode, &m_ICxCPEstablishedSessionPtr );
```

Then you use ICxCPEstablishedSession pointer and get available ICxDCIM pointer by calling GetDCIM function. Use the predefined Application DCIM instance name (AppDCIMInstance) for the first argument for GetDCIM function.

```
// Get access to the Application DCIM
hr = m_ICxCPEstablishedSessionPtr->GetDCIM( _bstr_t("AppDCIMInstance"),
&myCxCPReturnCode, &m_ICxDCIMPtr );
m_ICxAppDCIMObjPtr = m_ICxDCIMPtr;
```

In order to use a callback functions, it has to be registered. First create a callback COM Object. Then call the RegisterCB function to register it.

```
// Create an Application DCIM callback interface object:
hr = CoCreateInstance( CLSID_MyAppDCIMCB, NULL, CLSCTX_SERVER,
IID_IMyAppDCIMCB, (void **)&m_pIMyAppDCIMCB );

// Register the Application DCIM callback interface
m_ICxAppDCIMObjCBPtr = m_pIMyAppDCIMCallbacks;

if ( NULL != m_ICxAppDCIMObjCBPtr )
{
  // Register to use bytestream callbacks since we are out-of-process.
```

```
  hr = m_ICxAppDCIMObjPtr->RegisterCB( m_ICxAppDCIMObjCBPtr, VARIANT_TRUE
);
}

//If there is any error, free sErrorString of CxCPReturnCode to avoid
memory leak.
if(NULL != myCxCPReturnCode.sErrorString)
  SysFreeString(myCxCPReturnCode.sErrorString);

return TRUE;
```

Set bUseByteStream in RegisterCB to TRUE if the application DCIM is out-of-process from the CxAppDCIM, when it is not a DLL. This setting will cause AppDCIM to call byte stream versions of callbacks for increase in performance.

14. Add your appropriate logic to GetDataByteStream functions of COM interfaces (CMyAppDCIMCB class) inheriting ICxAppDCIMObjCB interface.

    Please set the value of bHandled argument in GetDataByteStream API if all the requested parameters and paramValues argument is populated at the time of the callback are available. Otherwise application has to return from callback and use ICxAppDCIMObj::GetDataReply API asynchronously to return requested data values.

    (If callback is registered with FALSE bUseByteStream argument in RegisterCB API. Add these logic to GetData API function.)

15. Add logic to update values into the cache or trigger events/exceptions, by calling other API functions(UpdateDataByteStream, TriggerEventByteStream, TriggerExceptionByteStream, etcs).

3.5.7 AppDCIM Configuration

The AppDCIM uses .ADC file to store its configuration. The same .ADC file must be used for runtime and model configuration of the DCIM Instance (in Equipment Model Developer). The ADC file is modified using the ADC Editor. See the section for this application for more details.

3.5.8 App DCIM Data Collection

The App DCIM support several methods for collecting parameter values and implementing other data collection functionality. This section summarizes the various options for handling Parameters, Events, and Exceptions.

3.5.8.1 Parameters

The data collection application can cache any Parameter values in the App DCIM using any of the UpdateData, TriggerEvent and TriggerException functions. For any TraceReport, ExceptionReport, EventReport or GetParameterValues request, the Application DCIM will use the current cached value. When a parameter's value is not cached, then the App DCIM will expect the data collection application to implement a GetValue callback function to return any requested value.

All configuration data (equivalent to GEM Equipment Constants), EventData (equivalent to GEM Data Variables), Exception SetData and Exception ClearData should be cached in the App DCIM by passing in the value with the corresponding TriggerEvent or TriggerException function. This ensures that the parameter value is correct in the associated TraceReport or ExceptionReport.

For status data, the App DCIM supports three basic techniques for updating the parameter values. A combination of techniques can, of course, be implemented.

3.5.8.1.1 Cache All the Data All the Time

The simplest strategy is to update all the parameter values all the time to provide a constant CPU load on the system using one of the UpdateData functions. All of the data does not need to be updated at the same rate or in the same UpdateData call. If fact, this is often not the best approach. Some critical data might be updated every 100 ms while less critical data might be updated every 10 s. There might be multiple threads and/or applications where each updates a different group of related data. The update rate for each parameter should be appropriately designed to provide both sufficient data quality and reasonable CPU usage.

3.5.8.1.2 On Demand Callback

The App DCIM can query Parameter values on demand using the GetValue methods in the App DCIM callback. For each TraceReport, ExceptionReport, EventReport or GetParameterValues request, the Application DCIM will request a new current value using a GetValue callback. This method uses no CPU when there is not data collection. However, if there is a high data collection load, then the CPU usage will be high, even higher than caching all the data all of the time.

It is particularly appropriate to use the callback technique for non-transient status data where the values are changing frequently like an analog signal.

The GetValue callback must be implemented very efficiently so that it returns the values to the App DCIM very quickly. Failure to do so will directly result in poor data collection performance.

3.5.8.1.3 Cache only Requested Values

The App DCIM can notify the data collection application whenever a data collection plan is activated and deactivated using one of the ReportChange callbacks. The data collection application can use this information to determine which is actually being gathered. The data collection application should not use this information to alter the data update rates to match the data collection plan. There might be multiple data collection plans requesting the same data. The ReportChange callback should only be used to turn on which data is updated the Application DCIM's cache. In a worst case situation where all data has been requested in one or more data collection plans, then this implementation should be the same as the first technique mentioned above where all data is cached all the time.

3.5.8.2 Events

There are two basic techniques for handling events. In all cases when triggering an event, include the values of any EventData in the TriggerEvent API function. This ensures that the correct values are included in the EventReport.

3.5.8.2.1 Trigger All Events All the Time

This is the simplest technique for handling events. Call the TriggerEvent function each time an event occurs.

3.5.8.2.2 Trigger Only Requested Events

Use the App DCIM's ReportChange callback to notify the data collection application which events should be triggered.

3.5.8.3 Exceptions

There are two basic techniques for handling exceptions. In all cases when triggering an exception, include the values of any SetData and ClearData in the TriggerException API function. This ensures that the correct values are included in the ExceptionReport.

3.5.8.3.1 Trigger All Exceptions All the Time

This is the simplest technique for handling exceptions. Call the TriggerException function each time an exception changes state.

3.5.8.3.2 Trigger Only Requested Exceptions

Use the App DCIM's ReportChange callback to notify the data collection application which exception state changes should be reported.

3.5.8.4 Additional Tips

When handling large lists of Parameters in one call, it is more efficient to use the ID version of the UpdateData functions, particularly when updating data from another computer and when using long Parameter names.

During startup, give the App DCIM an initial value for each cached parameter. This ensure that the GetValue callback won't be called unexpectedly.

Use the ByteBuffer functions when the data collection application is an executable. The non-ByteBuffer functions should be used when the data collection application is using the App DCIM in a DLL.

For status data, document the implemented update rates. This will discourage clients from unnecessarily collecting data at rates that are too fast.

3.6 CIMConnect DCIM Overview

The CIMConnect DCIM is a plug-in to CIMPortal Plus used to collect data from CIMConnect. This is the best way for customers already using CIMConnect to make derived data, including all CIM300 data, available to the EDA port. Making the CIMConnect data available to CIMPortal Plus is a simple matter of configuring the EDA connection in CIMConnect and configuring the DCIM for CIMPortal Plus, **NO** code changes are required in the application for this derived data.

It has four parts:

- `CxCIMConnectDCIM.dll` is the DCIM living in CIMPortal Plus
- CxCCEDA.dll contains a Communications object for CIMConnect which
- `CxEDATask.dll` which contains the CIMConnect message processing tasks for the EDA connection
- CIM300Mapping.txt is an optional mapping file for when CIM300 is used

3.6.1 COM ProgID

The COM ProgID for the CIMConnect DCIM is `CxCIMConnectDCIM.CxCIMConnectDCIMObj`.

3.6.2 Deliverables

In the Comm Products\bin directory:

- CxCCEDA.dll
- CxEDATask.dll
- CIM300Mapping.txt (needed if CIM300 is used)

In the CIMConnect DCIM Package:

- CxCIMConnectDCIM.dll
- Install_CIMConnectDCIM.bat
- Uninstall_CIMConnectDCIM.bat

3.6.3 CIMConnect and CIMPortal Plus on Different PCs

3.6.3.1 Equipment PC Configuration

This section describes necessary procedure to configure the Equipment PC to let CIMConnectDCIM collect data and send it to CIMPortal Plus PC.

1. Copy **vcredist_x86.exe** file (Microsoft Visual C++ 2005 Redistributable Package (x86)) to Equipment PC and execute it. The file is available from the CIMPortal Plus installation CD or Microsoft website. It installs runtime components of Visual C++ Libraries required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2005 installed.

   Note: The version number of vcredist_x86.exe has to be **2.0.50727.762**.

2. Copy and register the following files from CIMPortal Plus PC to Equipment PC so that CIMConnectDCIM can be used on Equipment PC.

   Copy these files to C:\Program Files\Cimetrix\comm products\bin in Equipment PC.

   - CxCCEDA.dll
   - CxEDATask.dll
   - CxDCIMInterfance.dll
   - CxCIMPortalSvc.tlb

   Register the copied file. (CxEDATask.dll does not have to be registered because it is not COM dll.)
   regsvr32 CxCCEDA.dll

   regsvr32 CxDCIMInterface.dll

   regtlb.exe CxCIMPortalSvc.tlb

   regtlb.exe EMService.tlb

   If you are interested in how CIMPortal Plus works together with CIMConnect through CIMConnectDCIM, please see CIMPortal Plus help file, CIMConnectDCIM Overview.

3. Modify EPJ files loaded in CIMConnect on Equipment PC.

   Please see CIMConnect Configuration for details.

4. Load the updated EPJ file into CIMConnect.

   Copy the modified EPJ file under C:\CIMConnectProjects\Equipment1\Projects directory.
   Start CIMConnect Control Panel. Change the mode to Build and load the updated EPJ file. (Or, set the updated EPJ file as a default EPJ file.)
   For further detailed instruction, please see CIMConnect Help file.

3.6.3.2 CIMPortal Plus PC Configuration

1. Copy and register the following files from Equipment PC (the separate PC with installed CIMConnect) to CIMPortal Plus PC so that CIMPortal Plus can use CIMConnect's interface to communicate.

   Copy these files to C:\Program Files\Cimetrix\comm products\bin in CIMPortal Plus PC.

   - EMService.tlb
   - EMServiceps.dll

   Register the copied files.
   regtlb.exe EMService.tlb

   regsvr32 EMSErviceps.dll

If you are interested in how CIMPortal Plus works together with CIMConnect through CIMConnectDCIM, please read CIMPortal Plus help file, CIMConnectDCIM Overview.

2. Configure CIMConnectDCIM instance for an equipment model used for CIMPortal Plus on CIMPortal Plus PC.

   Please read CIMConnect Configuration in CIMPortal Plus help file.

3. Create a model with the CIMConnect instance created in previous step.

   For further detailed instruction, please read Equipment Model Developer Overview in CIMPortal Plus help file.

4. Create a PKG file for equipment, from which CIMConnectDCIM will collect data from.

   For further detailed instruction, please read Model Package Creation.

5. Deploy the PKG file.

   For further detailed instruction, please read Deploying Packages.

3.6.4 CIMConnect Configuration

To add an EDA connection to CIMConnect, add a new connection to the epj file calling out the CxEDA communications object and the CxEDATask task dll as shown below. This will be shipped as EDA.epj. Simply copy and paste this file into the existing tool's epj file. If there is already a CONNECTION2 in the epj file that is used, simply change the 2 to the next available number.

```
[CONNECTION2]

name=EDA Port

encoder=

decoder=

commif=CxCIMConnectDCIM.CxEDA

commifcfg=

taskdll=CxEDATask.dll
```

In addition, the following Connection specific variables are required. Again, change the 2 if necessary.

```
[CONNECTION2 VARIABLES]

#id=name, description, units, varType, valType, eventID, private, persistant, min,
max, default, wellknownname

2004=Clock,Machine's internal clock in the format YYYYMMDDhhmmsscc where Y = year M
= month D = day h = hour m = minute s = second c =
centisecond.,csec,SV,A,,0,0,,,,Clock

2029=EVENTSENABLED,List of all enabled CEID.,,SV,L,,0,1,,,L,EventsEnabled

2035=CommEnableSwitch,The GEM communications enable/disable operator switch where 0
= disabled and 256 = enabled,,SV,U4,,0,0,,,U4 256,CommEnableSwitch

2036=CommState,The current state of the GEM communications state machine where 0 =
disabled 260 = communicating 273 = WaitCRA | WaitCRFromHost and 273 = WaitDelay |
WaitCRFromHost,,SV,U4,,0,0,,,U4 260,CommState

4020=TimeFormat,Y2K time format selection. Possible values include 1 (compliant 16
byte) and 0 (not compliant 12 byte).,,EC,U4,,0,1,,,U4 1,TimeFormat

4022=EventReportMsg,Message sent on event trigger,,EC,U4,,0,1,,,U4
67083,EventReportMsg

4023=DefaultCommState,The default state of the GEM communications state machine
where 0 = disabled and 256 = enabled.,,EC,U4,,0,1,,,U4 256,DefaultCommState
```

```
4026=ClockOffset,Time offset between host's and local time,sec,SV,I4,,0,0,,,I4
0,ClockOffset
```

3.6.4.1 Do this:

1. Edit your epj file to include the information above for the new EDA connection.

3.6.4.2 Verify

Start the CIMConnect Control Panel and verify the new EDA connection is listed. If you click on the connection, an empty error dialog box with an OK button will pop-up; this is normal for this connection.

3.6.5 Alternate GEM startup

Following the above instructions will cause the CIM300 service to start whenever CIMConnect is started which in turn is whenever the CIMConnectDCIM is loaded at startup by CIMPortal Plus. If more control is desired over the startup of the CIM300 the following instructions should be followed.

Change the DefaultCommState EC mentioned above to be private, assign an unused ID number and a different name. We recommend something like this:

```
300000=DefaultCommStateInternal,The default state of the GEM communications state
machine where 0 = disabled and 256 = enabled.,,EC,U4,,1,0,,,U4 0,DefaultCommState
```

Define a new, public “DefaultCommState” EC, which is not mapped to well-known DefaultCommState, like this:

```
4023=DefaultCommState,The default state of the GEM Host communications state
machine where 0 = disabled and 256 = enabled.,,EC,U4,,0,1,,,U4 256,
```

Then in software, at the end of initialization when GEM host communication should be allowed, check the value of the new DefaultCommState EC, to see if EnableComm should be called or not. Something similar to this:

```
ulong defaultCommState = 0;
 for (int i = 1; i <= m_NumberOfConnections; i++)
 {
  GetVariableValue(i, "DefaultCommState", ref defaultCommState);
  if (256 == defaultCommState)
   m_CxClientClerk.EnableComm(i);
 }
```

3.6.6 DCIM Configuration

3.6.6.1 Prerequisites

- CIMConnect installed
- CIMPortal Plus installed
- CIMConnect configured for an EDA port
- If CIMConnect is running on a different computer, then DCOM permissions and security must be correctly configured.

3.6.6.2 Description

The Model and Runtime configuration strings for the CIMConnect DCIM have the same format:

```
<machine>,<equipment>,<eda port>,<gem port>,<message timeout>,<mapping file>,<user name>,<password>
```

Where:

<machine> is the machine name CIMConnect is running on. Use "localhost" if CIMConnect is on the same machine as CIMPortal Plus. If running on Windows 7 or later, use “0.0.0.0” instead of “localhost.”

<equipment> is the equipment number in CIMConnect to connect to. Use 0 if you only have one equipment.

<eda port> is the port number in CIMConnect that has been configured to use the CCxEDA communications object.

<gem port> and is the port that handles S14 messages for CIM300. This is not needed if E39 Object Attribute data is not to be collected and could be left empty. This is typically 1 and should be the same value passed into the CIM300 CIMConnect Abstraction layer ICCAlObject::Init method as ConnectionID.

<message timeout> is an optional timeout that the CIMConnect DCIM will wait for CIMConnect to reply to a message before it declares an error. This is time in milliseconds. The default timeout is 20000 milliseconds. Exceptions to this are the ICxDCIMModel interface ListEvents, ListExceptions, and ListParameters methods. These methods wait 4 times the timeout, or 80000 msec by default. This has a 1000 msec minimum value.

<mapping file> is the name of a mapping file like CIM300Mapping.txt that is used to determine the source of events from CIM300. If CIM300 is not used, the file is not needed, and this field should be empty.

<user name> is an optional user name for an account on the target machine when establishing a DCOM connection and the target machine requires identity.

<password> is the password for the user specified in <user name>.

Examples:

```
localhost,0,2,1,10000,CIM300Mapping.txt
```

would initialize for a connection on the same machine CIMPortal Plus is running on, connecting to equipment 0 in CIMConnect. The EDA port in CIMConnect is port 2, CIM300 is using port 1 and the timeout is 10 seconds. (40 seconds for the ICxModel methods).

```
localhost,0,2,,1000
```

would initialize the same connection except without the ability to collect E39 Object Attribute data.

```
localhost,0,2,,1000,,fred,fredpwd
```

would initialize the same connection without E39 data and over DCOM using fred’s credentials.

3.6.6.3 To configure a CIMConnect DCIM

1. In Equipment Model Developer’s DCIM administration tool, select the CIMConnect DCIM from the DCIM selector and press the Configure button. This will display the CIMConnectDCIM configuration dialog.
2. If CIMConnect is running on a different computer, enter the computer's name in the Machine Name box and, if necessary, the user name and password of an account on that machine.
3. Press the Query button. This will connect to CIMConnect and retrieve the list of running Equipment interfaces.

4. Select an Equipment interface from the Equipment list box
5. Press the Select button. This will query a list of host connections for that equipment.
6. Select the EDA connection from the Connections list box
7. Select the 300mm connection (if needed) from the 300mm Connection list box. This is the connection number passed into the CIM300 CIMConnect abstraction layer (CCAlObject) Init method.
8. If using CIM300, specify the mapping filename
9. Press the OK button.
10. If using CIM300, add the same mapping file name specified in step 8 to the DCIM Instance using the Add File(s) button.
11. Save the DCIM Instance configuration.

3.6.6.4 Mapping File

A default CIM300Mapping.txt mapping file is provided. It looks similar to the text below.

```
#CIMConnect DCIM mapping file for events/dvvals to model locators
#* is the only non-value object string allowed for the dvval field.
#it is used to mean "match everything"

#modify this section to match your model
 [DVTOLOCATORS]
#dvname=dvval,locator,dvval,locator etc.
CarrierID=*,Equipment/MaterialManager
PortID=U1 1,Equipment/LP1,U1 2,Equipment/LP2
PRJobID=*, Equipment/JobManager
CtrlJobID=*, Equipment/JobManager
SubstLocID=A Aligner, Equipment/Aligner,A EfemRobot, Equipment/EfemRobot
SubstID=*,Equipment/MaterialManager

#modify this section to match your model
[OBJECTS2LOCATORS]
#objtype=objid,locator,objid,locator
Carrier=*,Equipment/MaterialManager
Substrate=*,Equipment/MaterialManager
ProcessJob=*, Equipment/JobManager
ControlJob=*, Equipment/JobManager
SubstrateLocation=Aligner,Equipment/Aligner,EfemRobot,Equipment/EfemRobot
EPTTracker=Equipment,Equipment,Aligner,Equipment/Aligner,EfemRobot,Equipment/Ef
emRobot,LoadLock1, Equipment/LoadLock1
TS-Clock=TS-Clock,Equipment/TS-Clock

[EVENTTODV]
#eventname=dvname
#dvname must exist under DVTOLOCATORS
CarrierClosed=CarrierID
```

```
CarrierClamped=PortID
```

3.6.6.4.1 Mapping File Contents

Blank lines are allowed anywhere. Full, single- line comments start with '#'. Partial line comments are not allowed. The file is only read by the DCIM so the formatting will not be changed by the DCIM.

There are sections to the file: DVTOLOCATORS, OBJECTS2LOCATORS and EVENTTODV.

DVTOLOCATORS describes DVs to be used and the value/locator mapping pairs. For example, PortID=U1 1, Equipment/LP1, U1 2, Equipment/LP2 defines a mapping where Equipment/LP2 will be used as the locator (or source) of an event if the value of the PortID DV is 2. CarrierID=*,Equipment/MaterialManager defines a mapping where all carriers will be located in the MaterialManager node of the model. This section of the mapping file must be edited to match the model you create for your equipment. The '*' wildcard is only used for dynamic E39 Objects like Carriers, Substrate, ControlJobs and ProcessJobs.

EVENTTODV describes which DV will be used to determine the locator for a given event. CarirerClosed=CarrierID defines a mapping where the CarrierID DV will be used to determine the locator for the CarrierClosed event according to the mapping for CarrierID in the DVTOLOCATORS section. This section typically doesn't need to be edited. The only events that need to be in this section are events that are used by multiple E39 object instances or multiple state machine instances. Unique events that are only in the model at a single location do not need to be in this file.

OBJECTS2LOCATORS maps object ids to locators by object type. This section must be edited to match the locators in your model. The '*' wildcard is only used for dynamic E39 Objects like Carriers, Substrate, ControlJobs and ProcessJobs.

Your own, non-CIM300, events and DVs may be added to this file as needed. This would only apply to items published to CIMPortal Plus through the CIMConnect DCIM where a single GEM event is used for multiple E39 Object instances or state machine instances.

3.6.6.4.2 Mapping File Location

When configuring the CIM300 mapping file in the CIMConnectDCIM instance, the path to the file can be specified in these ways:

1. File with no path

   Example: `CIM300Mapping.txt`

   Meaning: Upon deployment, the mapping file will be searched for in the configuration directory of the package deployment location, typically %ProgramFiles%\Cimetrix\Comm Products\bin\Storage\Deployed Equipment\<EquipmentName>\DCIM Instances\<DCIMInstanceName>

   This option is recommended. When the mapping file is included in the DCIM instance configuration as an "Added File", then the file will be included in the package file, so that upon deployment, it will be found automatically.

2. File with absolute path

   Example: `C:\MyProject\CIM300Mapping.txt`

   Meaning: Upon deployment, the mapping file will be searched for in the specific folder.

   If using this option, the path chosen must exist on the runtime system. When using this option, the installation of the mapping file must be managed by the customer.

3.6.7 Redeploying Packages for CIMConnectDCIM Equipment

Section 2.2.2.6 Redeploying Packages describes package redeployment. This section contains additional instructions required for CIMConnectDCIM equipment.

3.6.7.1 CIMConnectDCIM Redeployment Addendum

The purpose of these instructions is to ensure that any changes to the configuration in the package are installed correctly when the changed package is redeployed.

1. Redeploy with the changed package according to the instructions in section 2.2.2.6.3 Steps to complete:.
2. Stop CIMPortal service. This can be done in the CIMPortal Plus Control Panel by using the menu command File > Shutdown CIMPortal Service.
3. Stop EMService service. This can be done in the CIMConnect Control Panel by using the menu command CIMConnect > Shutdown EMService.
4. Start CIMPortal service. This can be done by logging into the CIMPortal Plus Control Panel. When CIMPortal service starts, the equipment will be started and the CIMConnectDCIM will become enabled. This will cause the startup of CIMConnect service.

3.6.8 Troubleshooting

3.6.8.1 Items do not appear in DCIM properties

If the DCIM is connected but expected items do not appear in the DCIM properties box, the problem most likely exists in the CIMConnect's EPJ file.

Common issues to check are:

Duplicate ID numbers. The first field of the EPJ file contains an identifier number. This number must be unique to keep from having problems. If there are conflicts, then one or more of the duplicated items will not be found.

The Private property is set. Parameters have a "private" property. If this property is set to greater than zero (the default value), then the parameter is considered private and will not be shown in Equipment Model Developer although it does show in the CIMConnect Control Panel.

The Property is in the wrong section. If a property is in a connection specific section of the EPJ file, other than the EDA section, it will not show in the Equipment Model Developer. Typically properties should be in the section for common variables.

3.6.8.2 Active Traces are terminated when Equipment Model Developer is closed

If a model is opened in Equipment Model Developer using a DCIM configured to communicate with the same CIMConnect service as a deployed equipment, active traces on the equipment may be interrupted when Equipment Model Developer is closed.

CIMConnect assumes each port will only have one active connection at any point in time. The CIMConnectDCIM follows this assumption. When an equipment is deployed, it uses the port that was configured for the DCIM in Equipment Model Developer. When Equipment Model Developer opens the same model, it establishes a second connection to the port, violating the assumption and causing cross-talk between the connections.

Possible resolutions:

1) Don't open models using DCIMs that are also used by deployed equipment. Before opening a model that's the same as a deployed equipment, disable or delete the equipment or shutdown the CIMPortal service. This option is recommended for customers using CIMConnect SR12 or earlier.

2) If using a CIMConnect that is SR13 or later, instead of using CIMConnectDCIM, use CIMConnectSR13PlusDCIM. This new DCIM uses a completely different means of communicating

with CIMConnect that does not have the problem. "This option is recommended for customers using CIMConnect SR13 or later.

3) Multiple EDA ports can be configured in CIMConnect. The DCIM configuration can be manually edited so Equipment Model Developer and the deployed equipment use different ports. This option is fragile and requires advanced understanding of CIMConnect and DCIM configurations. The details are beyond the scope of this document. This option is not recommended.

3.6.9 Interpreting Logging in CIMConnect

This section will enable you to interpret the logs from the CIMConnect DCIM that appear in CIMConnect's EMService NSL logs.

3.6.9.1 Package Name

The Package name will always be CX_PKGID_EMSERVICE for messages from CIMPortal Plus and CX_PKGID_EMTASK for the message handling functions.

3.6.9.2 Sender Object

Sender Object will be of the form 0:CxEMComm:2, if the equipment is 0, and the CCxEDA connection is 2. Message handlers will be named like 0:CxEMTasks:2.

3.6.9.3 Decoded Message (Data)

An excerpt from an NSL log is shown in the table below showing the logging sequence for the ListParameters call from CIMPortal Plus to CIMConnect. As you can see, the sequence is very similar to a normal SECS message sequence. CX_EMNOT_MSGREADY followed by CX_EMNOT_TASKSTARTED and CX_EMNOT_TASKCOMPLETE. In the case below, the message body is actually empty.

Package Name	Sender Object	lExtra	Decoded Message (Data)
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_MSGREADY	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 4 | Value = 14 | String =| void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = GetIncomingMsg called | void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = GetIncomingMsg returning msg 4:14 | void* = 0
CX_PKGID_EMTASK	0:CxEMTasks:2	CX_EMNOT_TASKSTARTED	CTaskListParameters: 0:2
CX_PKGID_EMTASK	0:CxEMTasks:2	CX_EMNOT_LOGINFO	Message:
CX_PKGID_EMTASK	0:CxEMTasks:2	CX_EMNOT_LOGINFO	CTaskListParameters: Outgoing Message Type = 0XE
CX_PKGID_EMTASK	0:CxEMTasks:2	CX_EMNOT_TASKCOMPLETE	CTaskListParameters: 0:2
CX_PKGID_EMSERVICE	0:CxEMMsgClerk:2	CX_EMNOT_LOGINFO	ProcessMsg returned eq:0 c:2 id:4 type:0xe reply:1 msg:L {A CC|Clock|3|Machine's internal clock in the format YYYYMMDDhhmmsscc where Y

			= year M = month D = day h = hour m = minute s = second c = centisecond.|2004||2||StringType|0|||0|||0|}
CX_PKGID_EMSERVICE	0:CxEMMsgClerk:2	CX_EMNOT_LOGINFO	Sending Message, eq:0 c:2 id:4 type:0xe reply:1 msg:L {ACC|Clock|3|Machine's internal clock in the format YYYYMMDDhhmmsscc where Y = year M = month D = day h = hour m = minute s = second c = centisecond.|2004||2||StringType|0|||0|||0|}
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = SendMsg called | void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = SendMsg sending msg t:4 m:14 | void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = SendMsg synch msg sent t:4 m:14 | void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = SendMsg sending msg t:4 m:14 queue size=1 | void* = 0
CX_PKGID_EMSERVICE	0:CxEMComm:2	CX_EMNOT_LOGINFO	SEMNotify -- EquipmentID = 0 | ConnectionID = 2 | ItemID = 0 | Value = 256 | String = SendMsg exiting | void* = 0

3.6.9.4 EDA Message Types

Message Type	Number	Description
tEnable	0X1	enable the DCIM for data collection
tDisable	0X2	disable DCIM data collection
tTimeSync	0X3	synchronize the DCIM clock
tPing	0X4	connectivity check
tListJobs	0X5	get a list of the active jobs (CJ, PJ, recipes)
tListObjIDs	0X6	get a list of E39 object Ids given a type
tRequestData	0X7	get the values for a list of parameters
tCreateRequest	0X8	create an event, exception, or trace request
tDeleteRequest	0X9	delete an existing request
tStopTrace	0xA	stop an active trace
tStartTrace	0XB	start an inactive trace
tListEvents	0XC	list the available events

tListExceptions	0XD	list the available alarms
tListParams	0XE	list the available parameters
tNumObjects	0XF	return the number of available parameters, events and exceptions
tTraceChange	0X10	outgoing message to notify CIMPortal Plus that a trace has changed state
tInit	0X11	reinitialize the DCIM. All requests will be deleted.
tDeleteAllRequests	0X12	delete all requests
tGetTraceState	0X13	get the current state of a trace

3.7 CIMConnect SR13+ DCIM Overview

The CIMConnect SR13+ DCIM is a plug-in to CIMPortal Plus used to collect data from CIMConnect. This is the best way for customers already using CIMConnect to make derived data, including all CIM300 data. Making the CIMConnect data available to CIMPortal Plus is a simple matter of configuring the DCIM for CIMPortal Plus, NO configuration changes in CIMConnect or code changes in the application are required for this derived data.

It has four parts:

- `CxCIMConnectSR13PlusDCIM.dll` is the DCIM living in CIMPortal Plus
- CIM300Mapping.txt is an optional mapping file for when CIM300 is used (it is identical to the one used by the older CIMConnectDCIM).

3.7.1 COM ProgID

The COM ProgID for the CIMConnect DCIM is CxCIMConnectSR13PlusDCIM.CxCIMConnectDCIMObj.

3.7.2 Deliverables

In the Comm Products\bin directory:

- CIM300Mapping.txt (needed if CIM300 is used)

In the CIMConnect SR13+ DCIM Package:

- CxCIMConnectSR13PlusDCIM.dll
- Install_CIMConnectSR13PlusDCIM.bat
- Uninstall_CIMConnectSR13PlusDCIM.bat
- CxCIMConnectSR13PlusDCIM.tlb

3.7.3 CIMConnect and CIMPortal Plus on Different PCs

3.7.3.1 Equipment PC Configuration

This section describes necessary procedure to configure the Equipment PC to let CIMConnectDCIM collect data and send it to CIMPortal Plus PC.

1. Copy **vcredist_x86.exe** file (Visual C++ Redistributable for Visual Studio 2012 Update 3) to Equipment PC and execute it. The file is available from the CIMPortal Plus installation CD or Microsoft website. It installs runtime components of Visual C++ Libraries required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2012 installed.
2. Copy and register the following files from CIMPortal Plus PC to Equipment PC so that CIMConnectSR13PlusDCIM can be used on Equipment PC.

   Copy these files to C:\Program Files\Cimetrix\comm products\bin in Equipment PC.

   - CxDCIMInterfance.dll

- CxCIMPortalSvc.tlb

regsvr32 CxDCIMInterface.dll

regtlb.exe CxCIMPortalSvc.tlb

regtlb.exe EMService.tlb

3.7.3.2 CIMPortal Plus PC Configuration

1. Copy and register the following files from Equipment PC (the separate PC with installed CIMConnect) to CIMPortal Plus PC so that CIMPortal Plus can use CIMConnect's interface to communicate.

   Copy these files to C:\Program Files\Cimetrix\comm products\bin in CIMPortal Plus PC.

   - EMService.tlb
   - EMServiceps.dll

   Register the copied files.
   regtlb.exe EMService.tlb

   regsvr32 EMSErviceps.dll

2. Configure CIMConnectSR13PlusDCIM instance for an equipment model used for CIMPortal Plus on CIMPortal Plus PC.

   Please read CIMConnect Configuration in CIMPortal Plus help file.

3. Create a model with the CIMConnect instance created in previous step.

   For further detailed instruction, please read Equipment Model Developer Overview in CIMPortal Plus help file.

4. Create a PKG file for equipment, from which CIMConnectSR13PlusDCIM will collect data from.

   For further detailed instruction, please read Model Package Creation.

5. Deploy the PKG file.

   For further detailed instruction, please read Deploying Packages.

3.7.4 DCIM Configuration

3.7.4.1 Prerequisites

- CIMConnect SR13 or newer installed
- CIMPortal Plus installed
- If CIMConnect is running on a different computer, then DCOM permissions and security must be correctly configured.

3.7.4.2 Description

The Model and Runtime configuration strings for the CIMConnect SR13+ DCIM have the same format:

```
<machine>,<user>,<password>,<equipment>,<connection>,<using cim300>,<mapping file>
```

Where:

`<machine>` is the machine name CIMConnect is running on. Use "localhost" if CIMConnect is on the same machine as CIMPortal Plus. If running on Windows 7 or later, use “0.0.0.0” instead of “localhost.”

<user name> is an optional user name for an account on the target machine when establishing a DCOM connection and the target machine requires identity.

<password> is the password for the user specified in <user name>.

<equipment> is the equipment number in CIMConnect to connect to. Use 0 if you only have one equipment.

<connection> is the number of the connection that will be used for collecting connection-specific items (parameters and events). When CIM300 is used (<using cim300> is "1") it is the connection that handles S14 messages for CIM300 and should be the same connection number as passed into the CIM300 CIMConnect Abstraction layer ICCAlObject::Init method as ConnectionID.

<using cim300> is a flag (1=yes, 0=no) specifying that there is a CIM300 application and CIMConnect SR13+ DCIM should collect E39 Object Attribute data.

<mapping file> is the name of a mapping file, like CIM300Mapping.txt, that is used to determine the source of events from CIM300. If CIM300 is not used, the file is not needed, and this field should be empty.

Examples:

```
localhost,,,0,1,1,CIM300Mapping.txt
```

would initialize for a connection on the same machine CIMPortal Plus is running on, connecting to equipment 0, connection 1 in CIMConnect. CIM300 is being used and has CIM300 mapping data in "CIM300Mapping.txt" file.

```
localhost,,,0,1,0,
```

would initialize the same connection except without the ability to collect E39 Object Attribute data.

```
fredpc,fred,fredpwd,0,1,0,
```

would initialize the same connection without E39 data and over DCOM using fred's credentials.

3.7.4.3 To configure a CIMConnect SR13+ DCIM

1. In Equipment Model Developer's DCIM administration tool, specify the name for DCIM Instance, select the CIMConnectSR13PlusDCIM from the DCIM selector and press the Configure button. This will display the CIMConnect SR13+ DCIM's configuration dialog.
2. If CIMConnect is running on a different computer, enter the computer's name in the Machine Name box and, if necessary, the user name and password of an account on that machine.
3. Press the Query button. This will connect to CIMConnect and retrieve the list of running Equipment interfaces.
4. Select an Equipment interface from the Equipment list box
5. Press the Select button. This will query a list of host connections for that equipment.
6. Select the connection from the Connections list box
7. Check the "Used for CIM300" checkbox if the selected connection is configured for use with CIM300 and it is desired that the CIMConnect SR13+ DCIM provide E39 data to CIMPortal Plus.
8. If using CIM300, specify the mapping filename.
9. Press the OK button.
10. If using CIM300, add the same mapping file name specified in step 8 to the DCIM Instance using the Add File(s) button.
11. Save the DCIM Instance configuration.

3.7.4.4 Mapping File

A default CIM300Mapping.txt mapping file is provided. It looks similar to the text below.

```
#CIMConnect DCIM mapping file for events/dvvals to model locators
#* is the only non-value object string allowed for the dvval field.
#it is used to mean "match everything"

#modify this section to match your model
 [DVTOLOCATORS]
#dvname=dvval,locator,dvval,locator etc.
CarrierID=*,Equipment/MaterialManager
PortID=U1 1,Equipment/LP1,U1 2,Equipment/LP2
PRJobID=*, Equipment/JobManager
CtrlJobID=*, Equipment/JobManager
SubstLocID=A Aligner, Equipment/Aligner,A EfemRobot, Equipment/EfemRobot
SubstID=*,Equipment/MaterialManager

#modify this section to match your model
[OBJECTS2LOCATORS]
#objtype=objid,locator,objid,locator
Carrier=*,Equipment/MaterialManager
Substrate=*,Equipment/MaterialManager
ProcessJob=*, Equipment/JobManager
ControlJob=*, Equipment/JobManager
SubstrateLocation=Aligner,Equipment/Aligner,EfemRobot,Equipment/EfemRobot
EPTTracker=Equipment,Equipment,Aligner,Equipment/Aligner,EfemRobot,Equipment/Ef
emRobot,LoadLock1, Equipment/LoadLock1
TS-Clock=TS-Clock,Equipment/TS-Clock

[EVENTTODV]
#eventname=dvname
#dvname must exist under DVTOLOCATORS
CarrierClosed=CarrierID
CarrierClamped=PortID
```

3.7.4.4.1 Mapping File Contents

Blank lines are allowed anywhere. Full, single- line comments start with '#'. Partial line comments are not allowed. The file is only read by the DCIM so the formatting will not be changed by the DCIM.

There are sections to the file: DVTOLOCATORS, OBJECTS2LOCATORS and EVENTTODV.

DVTOLOCATORS describes DVs to be used and the value/locator mapping pairs. For example, PortID=U1 1, Equipment/LP1, U1 2, Equipment/LP2 defines a mapping where Equipment/LP2 will be used as the locator (or source) of an event if the value of the PortID DV is 2. CarrierID=*,Equipment/MaterialManager defines a mapping where all carriers will

be located in the MaterialManager node of the model. This section of the mapping file must be edited to match the model you create for your equipment. The '*' wildcard is only used for dynamic E39 Objects like Carriers, Substrate, ControlJobs and ProcessJobs.

EVENTTODV describes which DV will be used to determine the locator for a given event. CarirerClosed=CarrierID defines a mapping where the CarrierID DV will be used to determine the locator for the CarrierClosed event according to the mapping for CarrierID in the DVTOLOCATORS section. This section typically doesn't need to be edited. The only events that need to be in this section are events that are used by multiple E39 object instances or multiple state machine instances. Unique events that are only in the model at a single location do not need to be in this file.

OBJECTS2LOCATORS maps object ids to locators by object type. This section must be edited to match the locators in your model. The '*' wildcard is only used for dynamic E39 Objects like Carriers, Substrate, ControlJobs and ProcessJobs.

Your own, non-CIM300, events and DVs may be added to this file as needed. This would only apply to items published to CIMPortal Plus through the CIMConnect DCIM where a single GEM event is used for multiple E39 Object instances or state machine instances.

3.7.4.4.2 Mapping File Location

When configuring the CIM300 mapping file in the CIMConnectSR13PlusDCIM instance, the path to the file can be specified in these ways:

1. File with no path

   Example: `CIM300Mapping.txt`

   Meaning: Upon deployment, the mapping file will be searched for in the configuration directory of the package deployment location, typically %ProgramFiles%\Cimetrix\Comm Products\bin\Storage\Deployed Equipment\<EquipmentName>\DCIM Instances\<DCIMInstanceName>

   This option is recommended. When the mapping file is included in the DCIM instance configuration as an "Added File", then the file will be included in the package file, so that upon deployment, it will be found automatically.

2. File with absolute path

   Example: `C:\MyProject\CIM300Mapping.txt`

   Meaning: Upon deployment, the mapping file will be searched for in the specific folder.

   If using this option, the path chosen must exist on the runtime system. When using this option, the installation of the mapping file must be managed by the customer.

3.7.5 Troubleshooting

3.7.5.1 Items do not appear in DCIM properties

If the DCIM is connected but expected items do not appear in the DCIM properties box, the problem most likely exists in the CIMConnect's EPJ file.

Common issues to check are:

Duplicate ID numbers. The first field of the EPJ file contains an identifier number. This number must be unique to keep from having problems. If there are conflicts, then one or more of the duplicated items will not be found.

The item is in the wrong section. If an item is in a connection specific section of the EPJ file, other than the one specified in DCIM configuration, it will not show in the Equipment Model Developer.

3.8 PerfMon DCIM Overview

3.8.1 Description

The Cimetrix PerfMon DCIM is a plug-in to CIMPortal Plus used to collect data from Windows system Performance Counters.

The PerfMon DCIM may be used to collect system performance data from any available performance provider, same as the Performance Monitor application included with Windows NT and above (perfmon.msc Microsoft Management Console plug-in module in Windows 2000, XP and Server 2003).

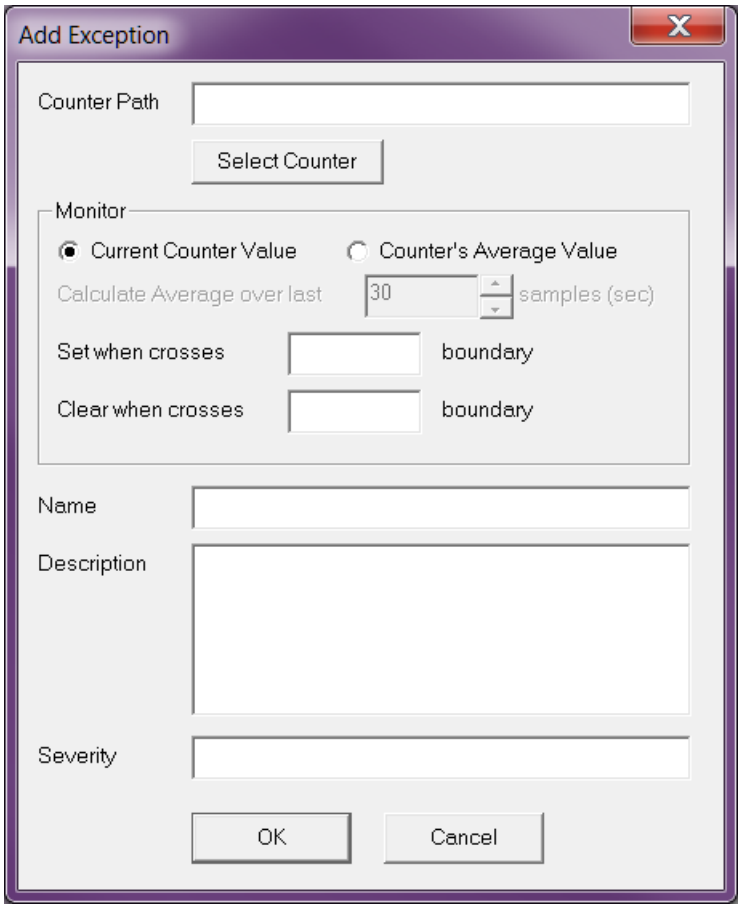
The PerfMon DCIM collects performance counter data at the set rate of 1 Hz. It allows performance data to be viewed as CIMPortal Plus DCP (Data Collection Plan) Parameter or Exception objects.

For the DCP Parameters, the data can be formatted in a structure with 4 elements, or individual `F8` values for Current, Minimum, Maximum or Average value of the Performance counter. For structure, format of the Parameter is:

```
L, 4
     {F8 <current counter value>}
     {F8 <minimum value>}
     {F8 <maximum value>}
     {F8 <average value>}
```

The minimum and maximum counter values are updated from the time PerfMon DCIM was enabled and reset when DCIM is disabled then re-enabled, or re-initialized. The average value is calculated from last N-number of samples, where N is configurable per each Parameter and may be in range of 1 to 86400 (from 1 second to a full day) or 0 to calculate this value from the time DCIM was enabled, re-enabled or initialized.

In case of DCP Exceptions, the PerfMon DCIM is monitoring a performance counter and changes Exception's state depending on the current or average (over N-samples) value of that counter. The Exception's state is changed to SET when the value crosses 'set limit' boundary from 'cleared' range and CLEAR when it crosses the 'clear limit':



3.8.2 COM ProgID

The COM ProgID for the PerfMon DCIM is `CxPerfMonDCIM.CxPerfMonDCIMObj`.

3.8.3 Deliverables

In the PerfMon DCIM Package:

- CxPerfMonDCIM.dll
- Install_PerfMonDCIM.bat
- Uninstall_PerfMonDCIM.bat

3.8.4 PerfMon DCIM Configuration

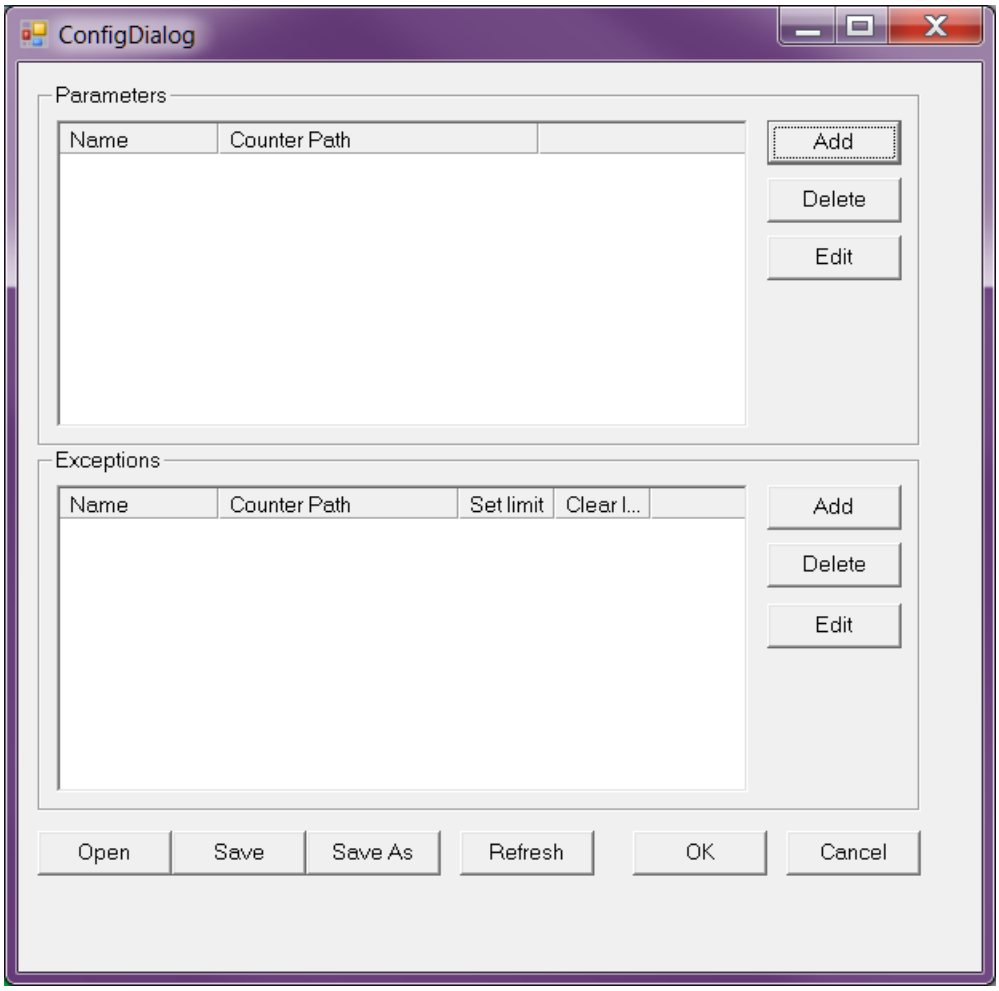
3.8.4.1 Prerequisites

- Equipment model information.

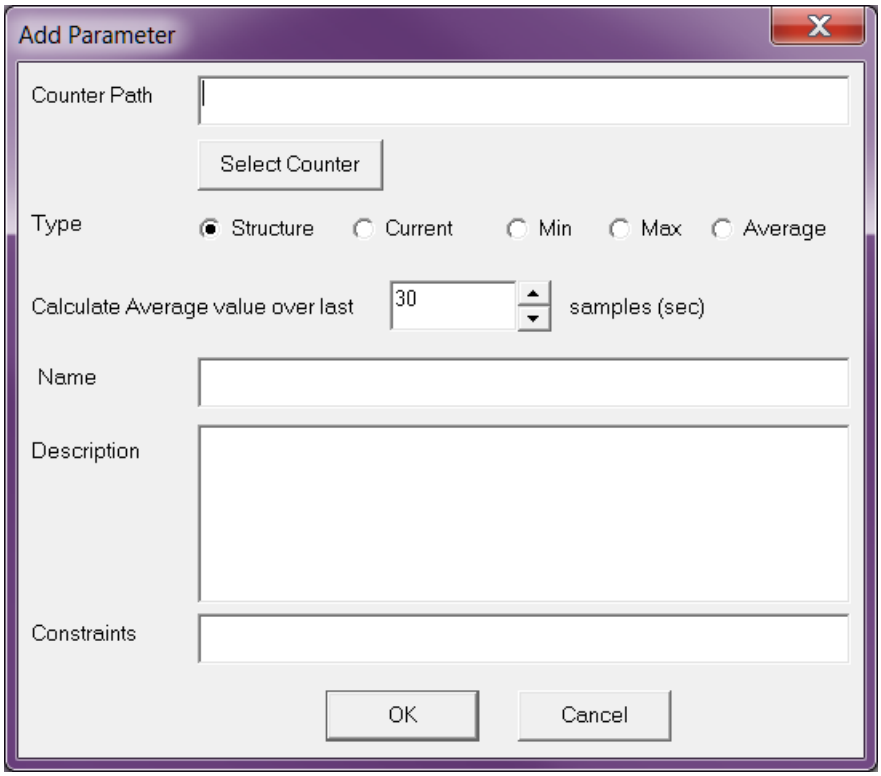
3.8.4.2 Description

This ActiveX control provides a manual, GUI-driven way of creating a model for PerfMon DCIM.

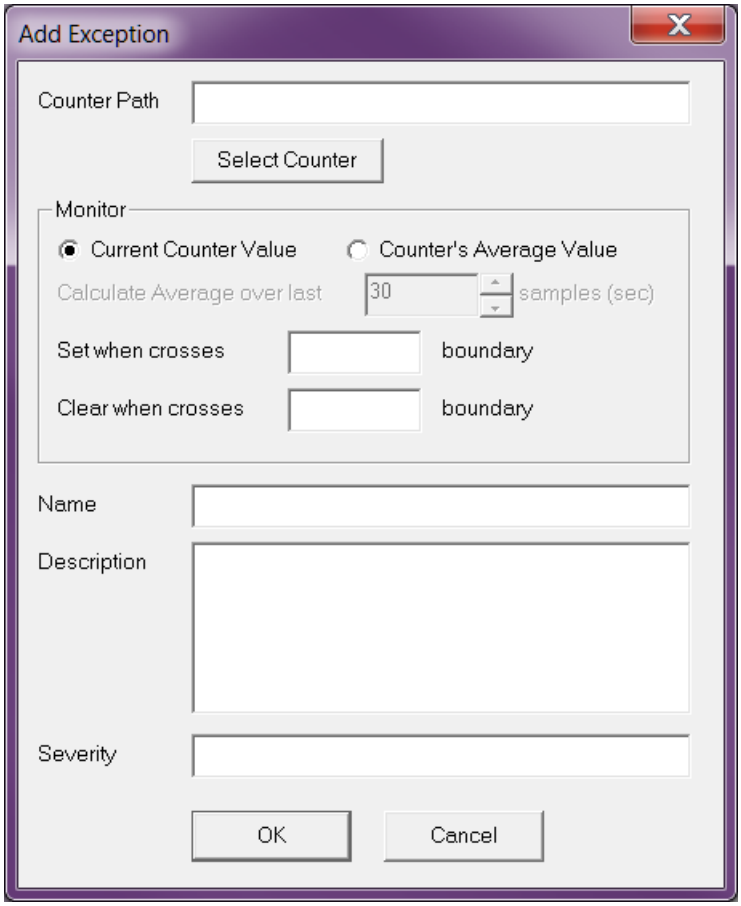
Screenshot of the ActiveX configuration control provided by the PerfMon DCIM



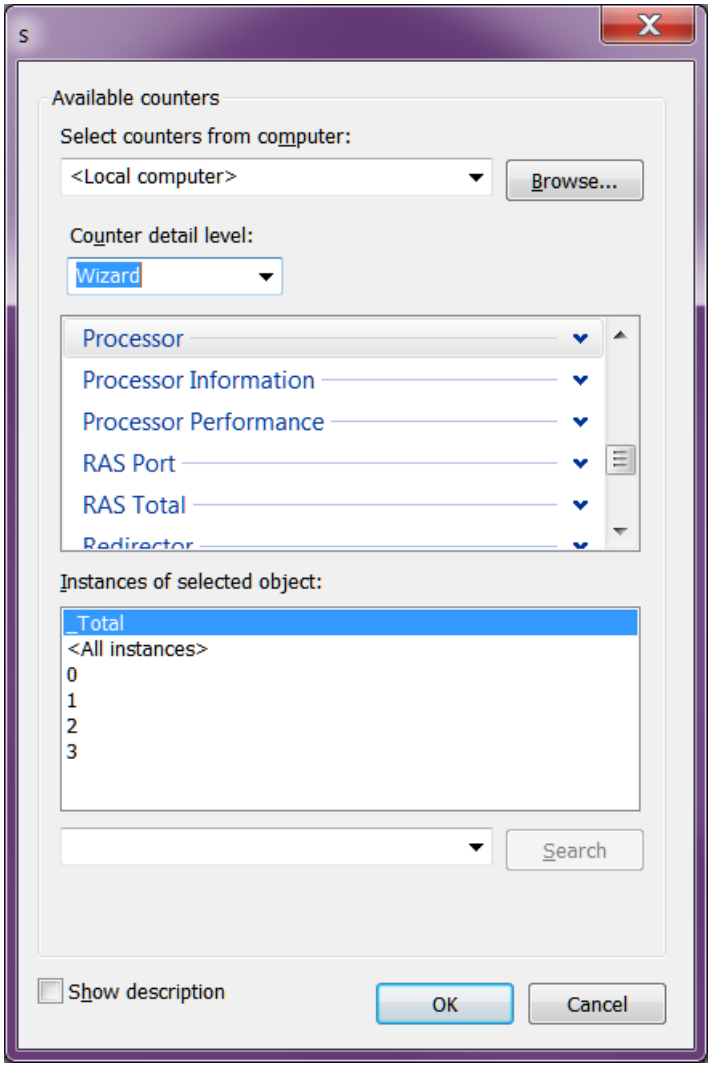
The ActiveX control has two lists - Parameters and Exceptions with ability to add new, edit or delete existing objects. When adding or editing Parameters an 'Add/Edit Parameter' form is displayed allowing selection of the Performance Counter, Parameter Type - wherever the Parameter's value is Current, Minimum, Maximum, Average or all four values in a structure, Parameter's Name, Description, Constraints and the Number of samples to keep for calculation of the Average Value.


Add Parameter
Counter Path
Select Counter
Type
Structure
Current
Min
Max
Average
Calculate Average value over last
30
samples (sec)
Name
Description
Constraints
OK
Cancel

When adding or editing Exceptions an 'Add/Edit Exception' form is displayed allowing selection of the Performance Counter, Exception's Name, Description, Severity and the properties for Limit monitoring.



Both 'Add/Edit Parameter' and 'Add/Edit Exception' form's 'Select Counter' button brings up a standard Windows 'Browse Performance Counters' dialog, allowing selection of one counter on local or remote system.



The ActiveX control also has the following buttons:

- Refresh - refreshes the tree view of DCP objects
- Open - brings up the standard Windows File Open dialog allowing loading of saved configuration files.
- Save - save the current model configuration into the file
- Save As - brings up the standard Windows File Save dialog allowing saving of current configuration into a file

3.9 TCPApp DCIM Overview

3.9.1 Description

The Cimetrix TCPApp DCIM is a plug-in to CIMPortal Plus intended to collect data from remote programs over a TCP/IP connection. TCPApp DCIM provides plan management and report formatting so the DCIM developer can concentrate on the connection to the data source.

It consists of two parts. The first part is a DCIM which runs on the CIMPortal Plus computer and communicates with CIMPortal Plus through a COM interface. This part is provided by Cimetrix. The second part is a data collection application (DCA) which normally runs on a different computer. The DCA communicates with the DCIM over a TCP/IP connection. The DCA is not provided by Cimetrix, but source code for a sample DCA has been provided. A DCA developer may use any of this source code in his own DCA.

The TCPApp DCIM tries to maintain a constant connection to the DCA. If an error occurs and the connection is broken, the DCIM will automatically try to reconnect until it has re-established the connection.

The TCPApp DCIM always runs on the same computer as CIMPortal Plus. The DCA does not normally, but can, run on the same computer as CIMPortal Plus. It is not required to run DCA during modeling - all Model information is retrieved from the DCIM.

3.9.2 COM ProgID

The COM ProgID for the TCPApp DCIM is `CxTCPAppDCIM.CxTCPAppDCIMObj`.

3.9.3 Deliverables

In the Comm Products\Examples\CIMPortal Plus\CxTCPAppDCA directory:

- Example DCA source code

In the Comm Products\bin\Storage\DCIM Packages\TCPAppDCIM directory:

- CxTCPAppDCIM.dll
- Install_TCPAppDCIM.bat
- Uninstall_TCPAppDCIM.bat

3.9.4 TCPApp DCIM Configuration

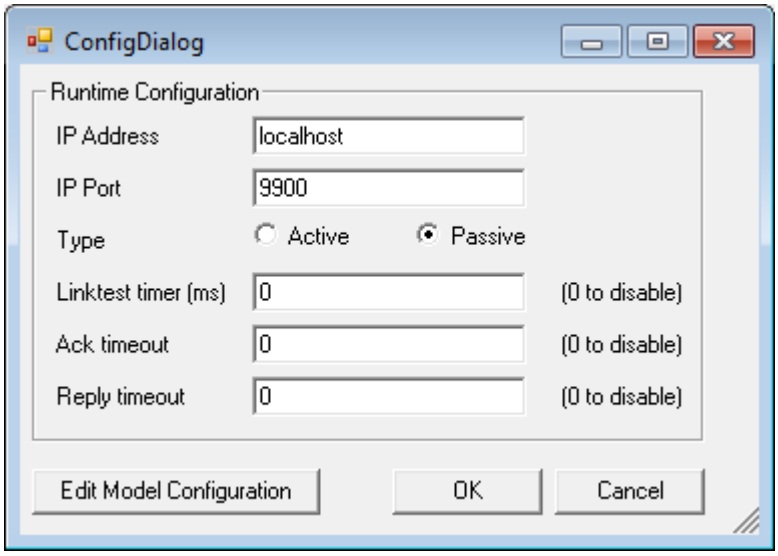
3.9.4.1 Prerequisites

- The type of TCP/IP connection the DCIM should use and the TCP/IP socket settings. Select an available TCP/IP socket.

3.9.4.2 Description

The configuration information is collected by an ActiveX configuration control provided by the TCPApp DCIM. This application control can be overridden if the DCIM and the DCA are connected.

The default ActiveX configuration control provided by the TCPApp DCIM looks similar to the following image:



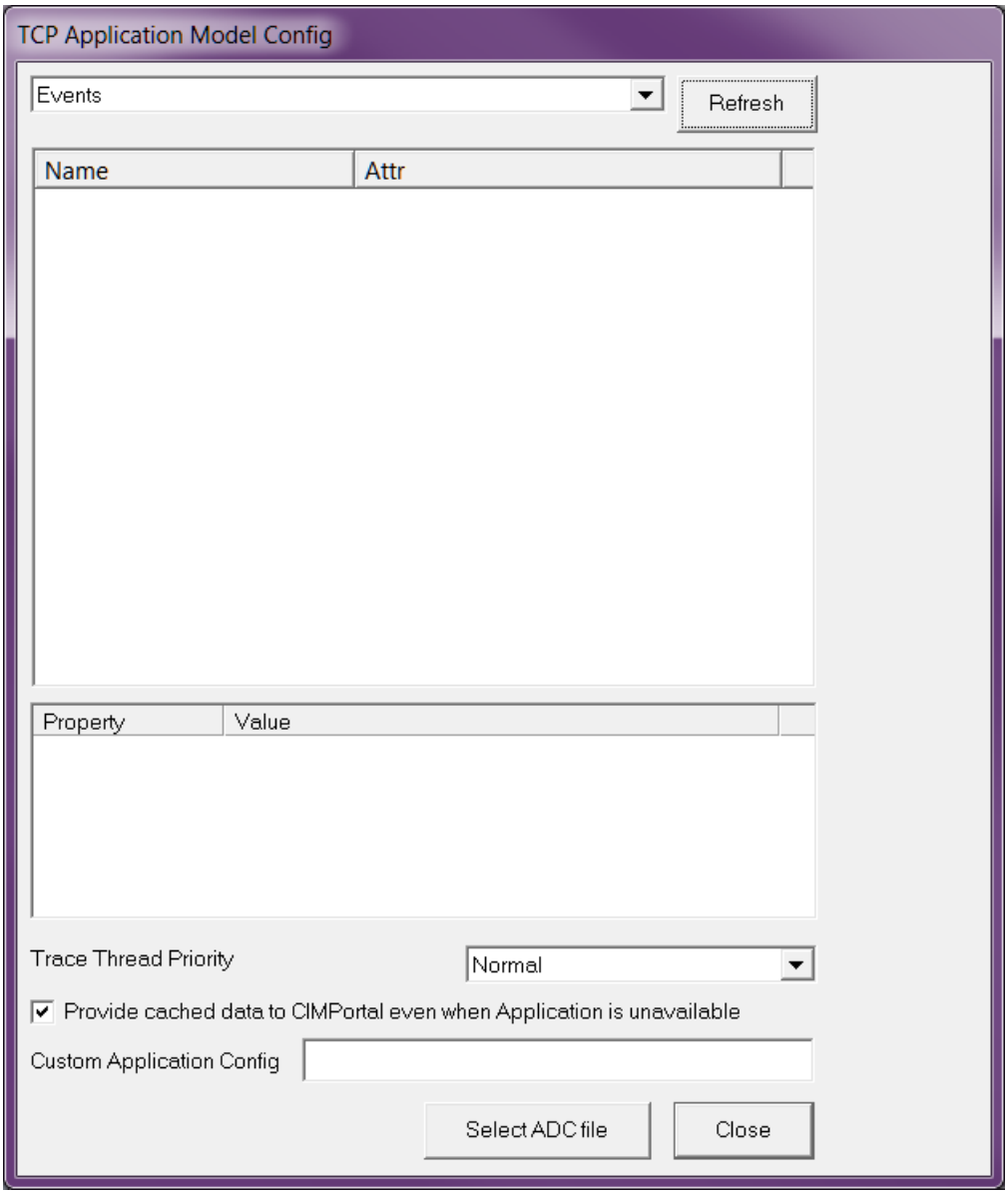
3.9.4.3 Runtime Configuration

1. The **IP Address** field should contain either the TCP/IP hostname or the IP address in the form of X.X.X.X. For a passive connection (recommended), this is the hostname or IP address that will be used to listen for an incoming connection from the DCA. When the network settings allow, the IP Address can be set to "0.0.0.0". For an active connection, this is the address of the computer with the DCA. Note that using an IP Address of "localhost" or "127.0.0.1" will limit the communication to local DCA applications running on the same computer whether the DCIM is passive or active.
2. The **IP Port** field will be the TCP/IP port number used to establish the connection. An available TCP/IP port number must be selected and known to both the DCIM and the DCA.
3. **Active/Passive** is the type of connection the DCIM should make. Passive is the recommended selection. The DCA should be the opposite of the DCIM where when one is passive the other must be active. The passive connection is sometimes known as a server connection and the active connection is sometimes known as a client connection. The recommended configuration is to setup the DCIM as the passive connection (set in this configuration window) and therefore the client as the active connection.
4. The **Linktest** timer field is the time period in milliseconds between sending linktest message. Linktest messages are only sent when no other messages are being sent. A zero (0) disables linktest messages.
5. The **Ack timeout** field is the time in milliseconds the DCIM will wait for an acknowledgment message before timing out and returning an error. A zero (0) value disables the timeout. If the timeout is disabled and the DCA dies, the DCIM (and CIMPortal Plus) will wait forever.
6. The **Reply timeout** field is the time in milliseconds the DCIM will wait after the acknowledgment message for the actual reply message before timing out and returning an error. If the timeout is disabled and the DCA dies, the DCIM (and CIMPortal Plus) will wait forever.

3.9.4.4 Model Configuration

The TCPAppDCIM uses an.ADC file to store its configuration.

Click on 'Edit Model Configuration' to select an ADC file. Use the ADC File Editor to create the ADC file.



The GUI control displays DCP objects in the list view filtered by object's type (Event, Exception or Parameter) and group. When a DCP object is selected in the list view, all its properties are displayed in the detail view bellow the list.

- **Refresh** - refreshes the tree view of DCP objects
- **Select ADC file** - brings up the standard Windows File Open dialog allowing loading of saved configuration files. Opening new files cause the TCPAppDCIM object to re-initialize and destroys previously created reports
- **Close** – close the dialog
- **Provide cached data to CIMPortal Plus even when Application is unavailable** - When checked, the TCPAppDCIM will always stay in Connected state and will provide the cached parameter data (see UpdateData message) to CIMPortal Plus. When unchecked, the TCPAppDCIM will only go into Connected state (allowing CIMPortal Plus to create requests and collect data) only after Application connects to the DCIM. When Application disconnects, the TCPAppDCIM will change its state to Not Connected and clear the cached data.

- **Custom Application Config** - TCPAppDCIM allows a custom configuration string stored with the DCIM Instance. Use GetCustomConfig message to get this string.

3.9.5 TCPApp DCA Development Components

3.9.5.1 Description

Any application which needs to connect to CIMPortal Plus using the TCPApp DCIM needs to address specific areas of functionality. The sample DCA provided by Cimetrix provides source code which implements each of these functionalities. The DCA developer can use the sample source code as is, modify it, or implement their own functionality. Regardless of the approach chosen, the DCA developer must address the following:

3.9.5.2 OS abstraction

OS abstraction restricts code which is OS specific to certain areas of the application. This allows the source code for the main logic of the application to be OS independent, normally resulting in "cleaner" source. This functionality is not required but is recommended. The sample DCA provides several OS abstraction classes to make it easier to port to different OSs.

3.9.5.3 Modeling

The DCA developer must determine how to map application data variables to CIMPortal Plus. This application-to-CIMPortal Plus model can be exposed to CIMPortal Plus by either building a model .ADC file using the ADC File Editor or by programmatically building the model using AddDCPObjects, DeleteDCPObjects and ListDCPObjects messages.

3.9.5.4 Data collection

DCA must provide parameter data, event and exception notifications to the CIMPortal Plus. It may also provide information about E39 objects and active jobs on the system. Send TriggerEvent, TriggerException messages to notify CIMPortal Plus about events and exceptions. To provide parameter data the DCA may use the UpdateData message and/or subscribe to GetData callback message.
The usage of UpdateData should be favored for parameters which values do not frequently change. The GetData callback mechanism should be used for frequently changing parameters.

3.9.5.5 E39 SEMI Objects

The DCA must publish changes to SEMI Objects.

- Use AddSemiObjectInstance to add objects that exist as they are created.
- Use RemoveSemiObjectInstance to remove objects when they are deleted.
- Use UpdateSemiObjectInstanceAttributeValue to update object attributes.
- Use TriggerSemiObjectInstanceEvent to trigger a semi object related event.
- Use ListAllSemiObjectInstances to retrieve a collection of all semi objects in the cache.
- Use RemoveAllSemiObjectInstances to clear the semi object cache.

3.9.5.6 TCP/IP connection management

The DCA will need some management of the TCP/IP connection.

In the sample DCA, the connection management is done by the CCxTcpipMessageHandler class. This same class is used by the TCPApp DCIM.

3.9.5.7 Message processing

Once the TCP/IP connection has been made, the DCA will need to listen for messages from CIMPortal Plus and respond to the messages.

In the sample DCA, message processing is done by the CCxTcpipDCIMExt class. A "Process Messages"

thread is created. This thread makes the connection to CIMPortal Plus (through the TCPApp DCIM) using the CCxTcpipMessageHandler class. Once the connection has been established, the thread waits for messages from CIMPortal Plus, unpacks the message, calls the correct DCA function, packs and sends the reply message.

3.9.5.8 Message packing

The messages between the TCPApp DCIM and the DCA must be known by each. The CCxMsgCodec is used by both the sample DCA and the TCPApp DCIM to pack and unpack messages. Messages are packed by adding parameters. An important feature is that a message can be embedded inside another message. This allows a message object to be used as a container. For example, a parameter value might be a composite value with a string and an integer. To pass the value of this parameter to CIMPortal Plus in a report message, the string and integer would be placed into a message object (valueMsg). This value message would be placed into another message with all the other parameter values that needed to be sent (dataMsg). This data message would be placed into yet another message along with other parameters of a report message (such as timestamp and reportID). See CCxTCPAppDCA::GetData for sample code of a complex parameter being packed (look at the 'Param1' case).

3.9.6 TCPApp DCA Development OS Porting

If the DCA developer chooses to use some of the provided source code on a different OS platform, there may be some porting issues. The sample DCA provided has been compiled and tested on Windows XP, Linux 2.4 (using gcc), and with some minor modifications, on a Big-endian Unix platform.

There are several classes to remove OS specific functionality from the main parts of the application. These classes deal with TCP/IP connections, threading, mutex locks, and time keeping.

3.9.6.1 Windows XP

To use the Windows version, make sure the project defines WIN32. This is normally already done by Microsoft Developer Studio. The project needs to support multithreading. There are no other issues if you are using a Windows XP platform.

3.9.6.2 Non-Windows

The messaging class CCxMsgCodec uses 32 and 64 bit integers. The typedefs for these variable types should be made at the top the CxMsgCodec.h file.

If the OS supports POSIX threading, make sure to define POSIX. This will use the POSIX threading and mutexes. If the OS does not support POSIX, the DCA developer will need to implement the CCxMutex and CCxThread classes.

If the OS supports the gettimeofday function, make sure to define USE_GETTIMEOFDAY. If the OS does not support WIn32 or gettimeofday, the DCA developer will need to implement the CCxTime class.

If the OS supports BSD sockets, there should not be a great deal of changes needed for the CCxTcpipMessageHandler class. Due to minor variations in header file location and function argument types, there may be some changes to get the class to compile. The DCA developer may wish to start with the changes made for the Linux 2.4. If the OS does not support BSD sockets, the DCA developer will need to implement the CCxTcpipMessageHandler class.

3.9.7 TCPApp DCA Development Overview

At its most basic level, a data collection application (DCA) using the Cimetrix TCPApp DCIM to connect to CIMPortal Plus must be able to send and receive messages in a pre-determined format. As long as the message format is correct, it doesn't matter if the DCA developer uses source code from the sample DCA or not. However, re-use of the sample code should speed development in most cases. Understanding the sample code should speed development in all cases.

This documentation goes over the implementation of the sample DCA with the understanding that the DCA developer may choose not to use the sample code. Where it makes sense, the documentation will combine both the general information with the DCA sample specific implementation.

The DCA sample files divided into two directories. The CxTcpipDCIMExt directory contains files believed to be common among many different DCAs. Some of these files are used by the TCPApp DCIM as well. These files are provided as source code instead of a compiled library so they can be used on whatever operating system is needed. The CxTCPAppDCA directory contains files specifically developed for the sample DCA. It is possible that some of this code may be reused as well.

3.9.8 TCPApp DCA Development Quick Start

For many DCA developers, the quickest way to get CIMPortal Plus to collect information from their application is to modify the sample application provided.

3.9.8.1 Step1

The first step in this process is to port (if necessary) the sample application to the OS which will be used to run the target application. Once the port is complete, the developer should run the sample application and verify the connection to CIMPortal Plus is working correctly.

3.9.8.2 Step 2

The second step is to create model. The model contains all the information necessary to map all the data exposed by the DCA into CIMPortal Plus objects. One of the most important issues the DCA developer must deal with is how this mapping occurs. Each piece of data collectible by CIMPortal Plus must have a unique name (within the DCIM which provides the data). To help the DCA developer, CIMPortal Plus associates an object with each data variable the DCA wishes to expose. This CIMPortal Plus object has two free-form strings the DCA developer can use to map the CIMPortal Plus object to the DCA data: the name and the attribute string. The name will be displayed to model developers and will be how the data variable is requested by E134 clients. The name should be meaningful and human readable. For some DCAs, a name is sufficient to uniquely identify a data variable. If the DCA needs additional information to identify the data variable, or wants to store additional information to help access it, the DCA should use the attribute string. For example, the attribute string could be used to hold a shared memory offset, the data type of the variable, the physical address of an I/O point, or a description on how to combine other data to form this data. This string is not generally displayed to humans, but should contain any information in a text format. The DCA may choose to either store all modeling information in an .ADC file, let DCIM load it during initialization or use AddDCPObjects/DeleteDCPObjects messages to build the model at runtime, once connected to DCIM. DCA can retrieve all available DCP Objects from DCIM using ListDCPObjects message.

3.9.8.3 Step 3

The third step is to implement the data collection functionality.

3.9.8.4 Step 4

The next step is to integrate the sample application into the target application. Normally the main.cpp file should be removed, and the remaining source files should be added to the target application. The target application will need to instantiate a CCxTcpipDCA object and call its CCxTcpipDCA::Run method. At this point, CIMPortal Plus should be able to gather data from the application.

3.9.8.5 Step 5

The last step is optimization. The DCA developer should examine the sample code and determine if it meets the application needs.

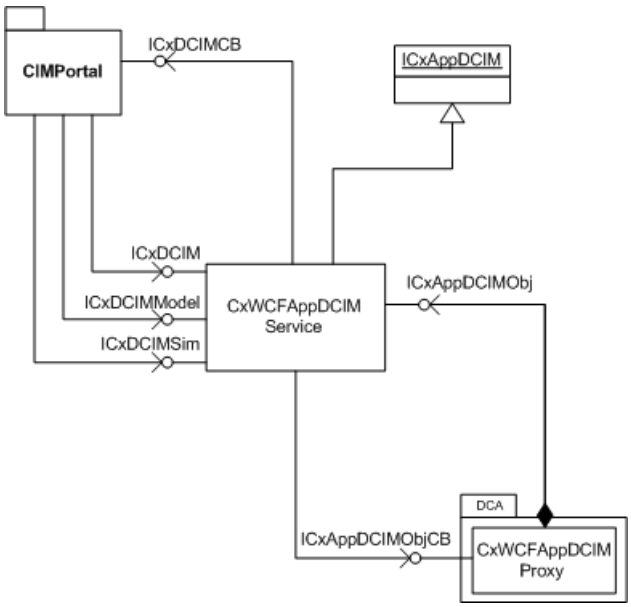
3.10 WCFAppDCIM Overview

The Cimetrix WCFAppDCIM is a plug-in to CIMPortal Plus used to collect data from applications using WCF (Windows Communication Foundation). The Cimetrix WCFAppDCIM provides plan management and report

formatting so the DCIM developer can concentrate on the connection to the data source. (NOTE: It is outside the scope of this document to describe WCF or why a developer may or may not choose to use this communication protocol.)

This consists of two parts. The first part is a DCIM which runs on the CIMPortal Plus computer and communicates with CIMPortal Plus through a COM interface. This part is provided by Cimetrix. The second part is a data collection application (DCA) which normally runs on a different computer. The DCA communicates with the DCIM using WCF. It is not necessary for the developer to have an in-depth understanding of WCF to use this feature, Cimetrix provides a Proxy object (CxWCFAppDCIMProxy) which provides the necessary WCF communication interfaces between the DCA and the DCIM. The developer need only use the functions contained in the Proxy, and create handlers for the events. The DCA is not provided by Cimetrix, but source code for a sample DCA has been provided. A DCA developer may use any of this source code in his own DCA.

The WCFAppDCIM always runs on the same computer as CIMPortal Plus. The DCA does not normally, but can, run on the same computer as CIMPortal Plus. It is not required to run DCA during modeling - all Model information is retrieved from the DCIM.



3.10.1 COM ProgID

There is no COM ProgID for this DCIM (communication is handled through the WCF).

3.10.2 Deliverables

In the Comm Products\Examples\CIMPortal Plus\WCFAppDCIM DCA directory:

- the sample DCA source code

In the Comm Products\bin\Storage\DCIM Packages\WCFAppDCIM directory:

- **CxWCFAppDCIM.dll** - This is the library containing the DCIM. This DCIM will be loaded by CIMPortal Plus based on the Model configuration.
- **CxWCFAppDCIMProxy.dll** - This library contains the Proxy object, and must be used by the DCA.
- **CxWCFAppDCIMContracts.dll** - This is the WCF Contract library, it is here that the WCF communication interfaces are defined. Both the Proxy and DCIM objects use this library and so must the DCA.

3.10.3 WCFAppDCIM DCA Async Error Handling

Many of the WCFAppDCIM methods discussed in subsequent sections are Async methods. If an Async method detects an error during its execution, it reports the error asynchronously. The error will be reported to the DCA by an event named OnAsyncServiceError. The DCA may register an event handler with the Proxy object. To ensure that all Async errors are handled, the event handler should be registered before invoking any Async methods.

To register for this event, use the standard .NET conventions in your DCA:

```
...
CxWCFAppDCIMProxy appDcim = new CxWCFAppDCIMProxy();
...
appDcim.OnAsyncServiceError += appDcim_OnAsyncServiceError;
```

You would then need to create the function for this handler in your DCA:

```
void appDcim_OnAsyncServiceError( object sender, DateTime timestamp,
     string serviceName, WCFAppDCIMFault error )
{
     int errorCode = error.errorCode;
     string errorDescription = error.errorDesc;
     ...
}
```

3.10.4 WCFAppDCIM DCA Event Management

When an event occurs, the DCA should notify the DCIM.

3.10.4.1 CxWCFAppDCIMProxy

The proxy object has three methods for notifying the DCIM of events (see the reference for more details):

```
public void TriggerEventAsync( string eventName, string source, DateTime
eventTimeStamp )
public void TriggerEventAsync( string eventName, string source, DateTime
eventTimeStamp, string[] paramNames, object[] paramValues )
public void TriggerEventAsync( string eventName, string source, DateTime
eventTimeStamp, int[] paramIds, object[] paramValues )
Proxy Example
if( appDcim != null && appDcim.Connected )
{
     try
     {
          int[] ids = { 1 }; // ID of the 'Parameter1' parameter
          CxValue[] values = { new I4Value(this.rand.Next()) };
          this.appDcim.TriggerEventAsync("Event1", "WCFAppSample", DateTime.Now,
ids, values);
     }
     catch( Exception exp )
     {
```

```
        Console.WriteLine("{0}\n{1}", exp.Message, exp.StackTrace);
    }
}
```

3.10.5 WCFAppDCIM DCA Exception Management

When an exception occurs, the DCA should notify the DCIM.

3.10.5.1 CxWCFAppDCIMProxy

The proxy object has three methods for notifying the DCIM of exceptions (see the reference for more details):

```
public void TriggerExceptionAsync( string exName, bool exSet, DateTime
exTimeStamp )

public void TriggerExceptionAsync( string exName, bool exSet, DateTime
exTimeStamp, string[] paramNames, object[] paramValues )

public void TriggerExceptionAsync( string exName, bool exSet, DateTime
exTimeStamp, int[] paramIds, object[] paramValues )
```

3.10.5.2 Proxy Example

```
if( appDcim != null && appDcim.Connected )
{
    try
    {
        appDcim.TriggerExceptionAsync( "Exception1", true, DateTime.Now );
    }
    catch( Exception exp )
    {
        Console.WriteLine( "{0}\n{1}", exp.Message, exp.StackTrace );
    }
}
```

3.10.6 WCFAppDCIM DCA Parameter Management

The DCA may update parameter data either by notifying (pushing) the data to CIMPortal Plus as it changes, or responding to requests (polling) as they are made, or a combination of both.

3.10.6.1 Push

The Proxy object provides two APIs to update data when parameter data changes (please see the reference for more details about these APIs):

```
public void UpdateDataAsync( string[] paramNames, object[] paramValues )

public void UpdateDataAsync( int[] paramIds, object[] paramValues )
```

3.10.6.2 Push Example

```
if( appDcim != null && appDcim.Connected )
```

```
{
     // Set Parameter3 to a List value
     var param3Value = new LValue();
     param3Value.Value.Add(new BoValue(true));
     param3Value.Value.Add(new AValue("test string"));
     param3Value.Value.Add(new F8Value(1.23456789));

     int[] ids = { 1, 2, 3 }; // IDs of the parameters
     var values = new CxValue[] { new I4Value(1), new F8Value(2.2), param3Value
};

     try
     {
          this.appDcim.UpdateDataAsync(ids, values);
     }
     catch( Exception exp )
     {
          Console.WriteLine("{0}\n{1}", exp.Message, exp.StackTrace);
     }
}
```

3.10.6.3 Poll

Occasionally, CIMPortal Plus requests data from the DCIM. If the data is not already cached, the DCIM will inquire the DCA for current data. To accomplish this, it is necessary for the DCA to create handlers for these requests, and register them with the Proxy object. The following is the declaration of the required event in the DCIM package. Please review the CxWCFAppDCIM reference for the full list of events:

```
CxWCFAppDCIMProxy
{
     ...
     public event GetDataEvent OnGetData
     ...
}
```

To register for this event, use the standard .NET conventions in your DCA:

```
...
CxWCFAppDCIMProxy appDcim = new CxWCFAppDCIMProxy();
...
appDcim.OnGetData += new GetDataEvent(appDcim_OnGetData);
```

You would then need to create the function for this handler in your DCA:

```
void appDcim_OnGetData( object sender, int requestID, int[] paramIds,
     out object[] paramValues, out bool handled )
```

```
{
      . . .
}
```

3.10.7 WCFAppDCIM Configuration

3.10.7.1 Prerequisites

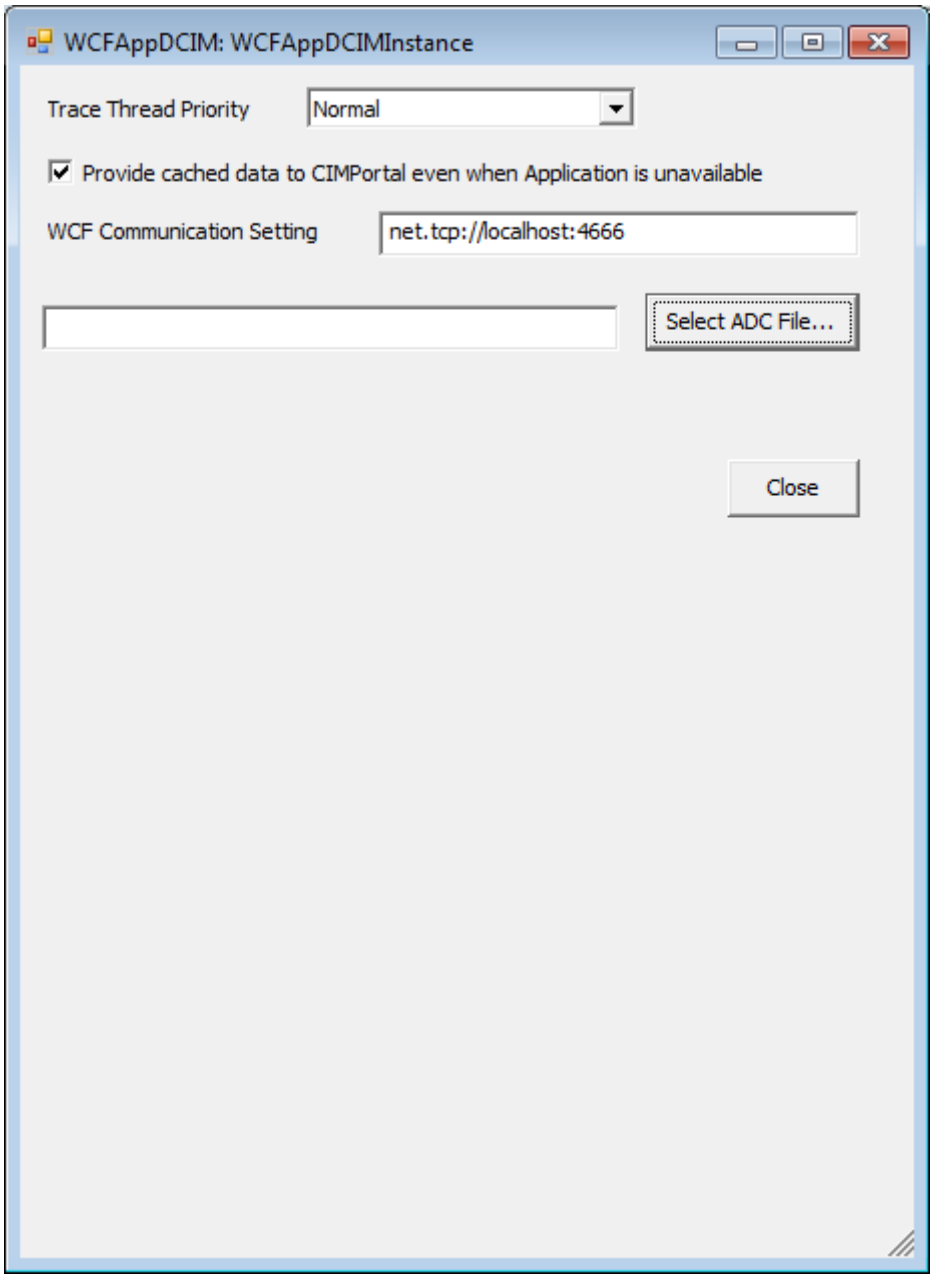
- The hostname (or TCP/IP address) and port number must be known for both the client and the service

3.10.7.2 Model Configuration

The WCFAppDCIM uses .ADC file to store its model configuration.

Use the Equipment Model Developer to configure the DCIM.

- Select the DCIM Administrator
- Type in a name of WCFAppDCIM instance in Instance Name textbox.
- Select WFCAppDCIM from the drop down list of DCIM Package.
- Press the "Configure DCIM" button.
- Set the TCP/IP port you want WCF to use in the WCF Communication Setting edit.
- Optionally select an ADC file by browsing to it after clicking on the ‘Select ADC File…’ button.



- **Select ADC File…** - brings up the standard Windows File Open dialog allowing loading of saved configuration files. Opening new files cause the DCIM object to re-initialize and destroys previously created reports
- **Close** – close the dialog
- **Provide cached data to CIMPortal Plus even when Application is unavailable** - When checked, the WCFAppDCIM will always stay in the Connected state and will provide the cached parameter data (see UpdateData message) to CIMPortal Plus. When unchecked, the WCFAppDCIM will only go into the Connected state (allowing CIMPortal Plus to create requests and collect data) only after an Application connects to the DCIM. When an Application disconnects, the WCFAppDCIM will change its state to Not Connected and clear the cached data.

- **WCF Communication Setting** - This field should be populated with a WCF service endpoint address such as 'net.tcp://localhost:4666'. You can change the TCP/IP port that the DCIM uses by changing 4666 to whatever port you like. In addition, you can change localhost to the specific IP address of a specific network adapter on the computer you would like to use.

3.10.8 WCFAppDCIM DCA Development Components

Any application which needs to connect to CIMPortal Plus using the WCFAppDCIM needs to address specific areas of functionality. The sample DCA implements each of these functions. The DCA developer can use the sample source code as is, modify it, or implement their own functionality. Regardless of the approach chosen, the DCA developer must address the following:

3.10.8.1 Modeling

The DCA developer must determine how to map application data variables to CIMPortal Plus. This application-to-CIMPortal Plus model can be exposed to CIMPortal Plus by either building a model .ADC file using ADC Text Editor. The .ADC file may also be created programmatically.

3.10.8.2 Data collection

DCA must provide parameter data, event and exception notifications to the CIMPortal Plus. Call one of the TriggerEvent and TriggerException methods to notify CIMPortal Plus about events and exceptions. Use UpdateDataAsync or OnGetData for publishing parameter values.

The usage of UpdateDataAsync should be favored for parameters which values do not frequently change. The OnGetData event/delegate mechanism should be used for frequently changing parameters.

3.10.8.3 E39 Semi Objects

E39 SemiObject information is published using the following methods:

- Call AddSemiObjectInstanceAsync when objects are created.
- Call RemoveSemiObjectInstanceAsync when objects are deleted.
- Call UpdateSemiObjectInstanceAttributeValuesAsync to update an object's attributes.
- Call TriggerSemiObjectInstanceEventAsync to publish a semi object related event.
- Call ListAllSemiObjectInstances to retrieve a collection of all semi objects in the cache.
- Call RemoveAllSemiObjectInstances to clear the semi object cache.

3.10.8.4 WCF connection management

The DCA does not need to manage the WCF Connection.

3.10.8.5 Interprocess Communication

Communication between the DCA and the DCIM is handled by the WCF. Any data traffic that originates on the DCA, would be pushed to the DCIM using the native proxy API functions. Any requests made by the DCIM would need to be handled by the DCA by creating event handlers/delegates for the specific events triggered by the DCIM. Eg. WCFAppDCIMProxy.OnGetData is an event that should be handled by the DCA in the event that the DCIM does not have the required data in its cache. The sample application shows several of these event/delegate handlers.

3.10.9 WCFAppDCIM DCA Development Quick Start

For many DCA developers, the quickest way to get CIMPortal Plus to collect information from their application is to modify the sample application provided.

3.10.9.1 Step 1

The first step is to create a model.

The DCA must store all modeling information in an .ADC file and let DCIM load it during initialization.

To build the ADC file, you can use the ADC File Editor.

Now use the EMDeveloper DCIM Administrator tool to create a DCIM Instance of the WCFAppDCIM using the ADC file created in the prior step. Remember to add the client endpoint address, such as net.tcp://localhost:4666, in the "Custom Application Config" field in the DCIM Configuration screen.

3.10.9.2 Step 2

The next step is to integrate CxWCFAppDCIMProxy object into the target application. See the sample program Form1.cs for details. Your application may not be a Windows Forms application, so the functionality in Form1.cs that integrates the CxWCFAppDCIMProxy must be integrated into appropriate areas of your application as needed.

Once the CxWCFAppDCIMProxy object is instantiated, it will automatically attempt to connect to the WCFAppDCIM. There should be little or no delay time from creation to connection, so data transfer can commence immediately. Please note in the DataCollection object, that before any command is invoked, it checks to see if the Proxy and DCIM are connected. This flag in the Proxy object is the only indication about whether or not the DCIM is ready to receive data.

3.10.10 WCFAppDCIM DCA Development Overview

3.10.10.1 Prerequisites

- The hostname (or TCP/IP address) and port number must be known for both the client and the service

3.10.10.2 Description

Use the WCFAppDCIM Proxy (CxWCFAppDCIMProxy) to send the parameter data to the CIMPortal Plus, trigger events and exceptions and to register for events from the CIMPortal Plus. As long as the developer is using this Proxy object, it doesn't matter if any of the sample code is used. However, re-use of the sample code should speedy development in most cases. Understanding the sample code should speedy development in all cases.

This documentation goes over the implementation of the sample DCA with the understanding that the DCA developer may choose not to use the sample code. Where it makes sense, the documentation will combine both the general information and the DCA sample specific implementation.

The DCA sample is a C# application, with a simple GUI interface that updates explicit parameters, events, and exceptions.

3.10.10.3 Application Configuration

The WCF connection information is stored in the DCA configuration file (app.config during development or <appname>.exe.config for runtime). This configuration file is in xml format. The following is an example of the two items that need to be added to this configuration file:

```
<services>
    <service
name="Cimetrix.CIMPortal.WcfAppDcim.CxWCFAppDCIMProxy+CxWCFAppDCIMProxyCallback
">
     <endpoint address="net.tcp://localhost:4667" binding="netTcpBinding"
     bindingConfiguration="DefaultBinding"
contract="CxWCFAppDCIMContracts.ICxAppDCIMCallback" />
    </service>
 </services>
 <client>
    <endpoint address="net.tcp://localhost:4666" binding="netTcpBinding"
```

```
    bindingConfiguration="DefaultBinding"
contract="CxWCFAppDCIMContracts.ICxAppDCIM"
    name="WcfAppDcimEndpoint" />
 </client>
```

3.10.11 WCFAppDCIM Proxy Configuration

3.10.11.1 Prerequisites

- The hostname (or TCP/IP address) and port number must be known for both the client and the service

3.10.11.2 Application Configuration

The configuration information is stored in the application configuration file (app.config during development or <appname>.exe.config for runtime). This configuration file is in xml format. The following is an example of the two items that need to be added to this configuration file:

```
<services>
    <service
name="Cimetrix.CIMPortal.WcfAppDcim.CxWCFAppDCIMProxy+CxWCFAppDCIMProxyCallback
">
     <endpoint address="net.tcp://localhost:4667" binding="netTcpBinding"
     bindingConfiguration="DefaultBinding"
contract="CxWCFAppDCIMContracts.ICxAppDCIMCallback" />
    </service>
 </services>
 <client>
    <endpoint address="net.tcp://localhost:4666" binding="netTcpBinding"
    bindingConfiguration="DefaultBinding"
contract="CxWCFAppDCIMContracts.ICxAppDCIM"
    name="WcfAppDcimEndpoint" />
 </client>
```

The endpoint of address in <services>

- It is a hostname and port number which a DCA makes available for CxWCFAppDCIM Service so that it can access ICxAppDCIMObjCB exposed by the DCA.

The endpoint of address in <client>

- It is a hostname and port number where a DCA will go to find ICxAppDCIMObj of CxWCFAppDCIM Service.
- This should be defined by "Custom Application Config" in ConfigDialog of EMDeveloper.

3.10.12 WCFAppDCIM application development

This sample demonstrates how to use the WCF Application DCIM (WCFAppDCIM). The sample project is a C# WinForms project.

Before starting using this sample, please extract "sample model.zip" file to your CIMPortal Storage directory, usually located in "C:\Program Files\Cimetrix\Comm Products\bin\Storage\". Load WCFAppSample.xml equipment model in EMDeveloper application and create a deployment package. Deploy the WCFAppSample equipment and configure it to be accessible from an EDA Client.

The WCFAppSample equipment is a very simple equipment model that uses the WCFAppDCIM. It contains only 3 parameters, 2 events and 2 exceptions. The .XML model file for this equipment is also included in WCFAppSample.xml file.

The application's configuration for WCF connections is configured through its App.config file.

The UI uses the CxWCFAppDCIMProxy helper class to update data, trigger events and change the state of the exception.

Once you open the Visual Studio project for the sample, update the references. CxWCFAppDCIMContracts and CxWCFAppDCIMProxy are in Cimetrix\Comm Products\bin\storage\dcim packages\WCFAppDCIM, the remained of the Cimetrix packages are in Cimetrix\Comm Products\bin. You should probably set Copy Local to true for the references.

3.10.12.1 DCIM Configuration

A WCFAppSample.adc application DCIM configuration file is provided in the directory and contains the definitions for the parameters, event, and exceptions used by the application.

3.11 XMLDCIM Overview

The Cimetrix XML DCIM is a plug-in to CIMPortal Plus used to collect data from an XML file. The Cimetrix XML DCIM allows data exposed in an XML file to be captured by CIMPortal Plus.

The XML DCIM collects the data from the file periodically. When CIMPortal Plus requests data from the XML DCIM, the DCIM checks to see if the file has been written since the time it was last loaded. If less than 10 seconds has elapsed since the last check, the XML DCIM does not do the file time check. Thus, this DCIM should not be used for data which changes frequently. The XML DCIM does not expose any events, exceptions, E39 objects, or jobs.

The XML DCIM logs to CIMPortal Plus for system debugging purposes. Log entries include, but are not limited to, when data collection plans are created, when plans start data collection, when plans stop data collection, and when data collection plans are deleted.

The XML DCIM always runs on the same computer as CIMPortal Plus.

3.11.1 COM ProgID

The COM ProgID for the XML DCIM is `CxXMLDCIM.CxXMLDCIMObj`.

3.11.2 Deliverables

In the XML DCIM Package:

- CxXMLDCIM.dll
- Install_XMLDCIM.bat
- Uninstall_XMLDCIM.bat

3.11.3 XML DCIM Configuration

3.11.3.1 Prerequisites

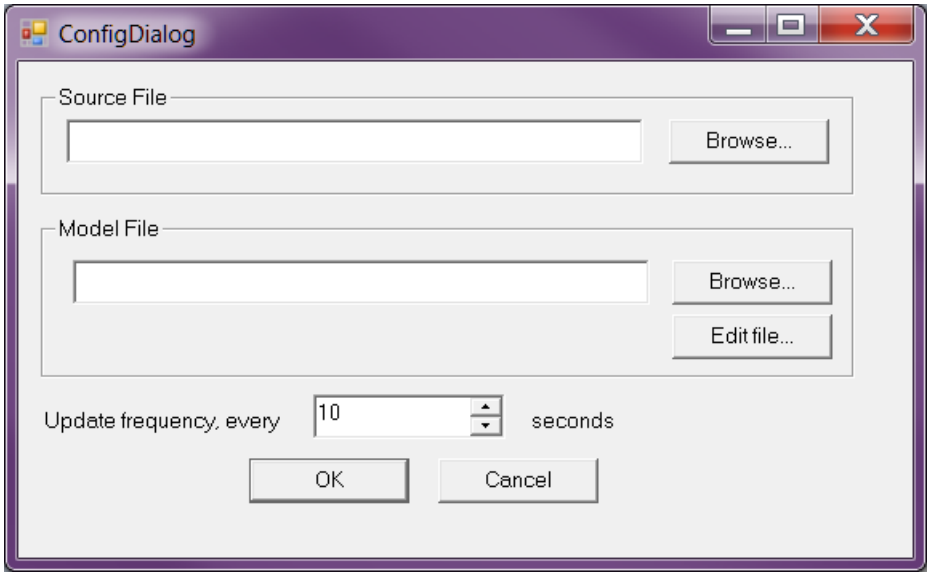
- The filename of the XML file must be known..
- If needed, the model filename must be known.

3.11.3.2 Description

The XML DCIM needs two types of information to be able to obtain useful data for the collection plans. The runtime data is the full path of the XML file from which the data will be extracted. The XML file must have the same name and location on computers used for modeling as the runtime data collection computer. The model data gives a mapping between the data in the XML file and the data the DCIM needs to report to CIMPortal Plus. This model data is stored in a file. If the model data is to be used, the

filename must be provided to the XML DCIM. The model data filename is only required in the modeling stage. The model data needed at runtime is stored in the equipment model file.

The ActiveX configuration control provided by the XML DCIM looks similar to the following image:



The runtime information is located at the top, in the Source File area. The fully qualified filename of the XML source data file should be provided here.

If needed, the model filename should be provided in the lower section, in the Model File area. If the file already exists, you may either type the filename in, or select it using the Browse button. If a new model file needs to be created or an old file needs to be changed, use the "Edit file..." button.

Once the runtime information and the model file are correct, the OK button will reinitialize the DCIM with the new configuration. The Cancel button will discard any changes.

3.11.4 XML DCIM Model Creator

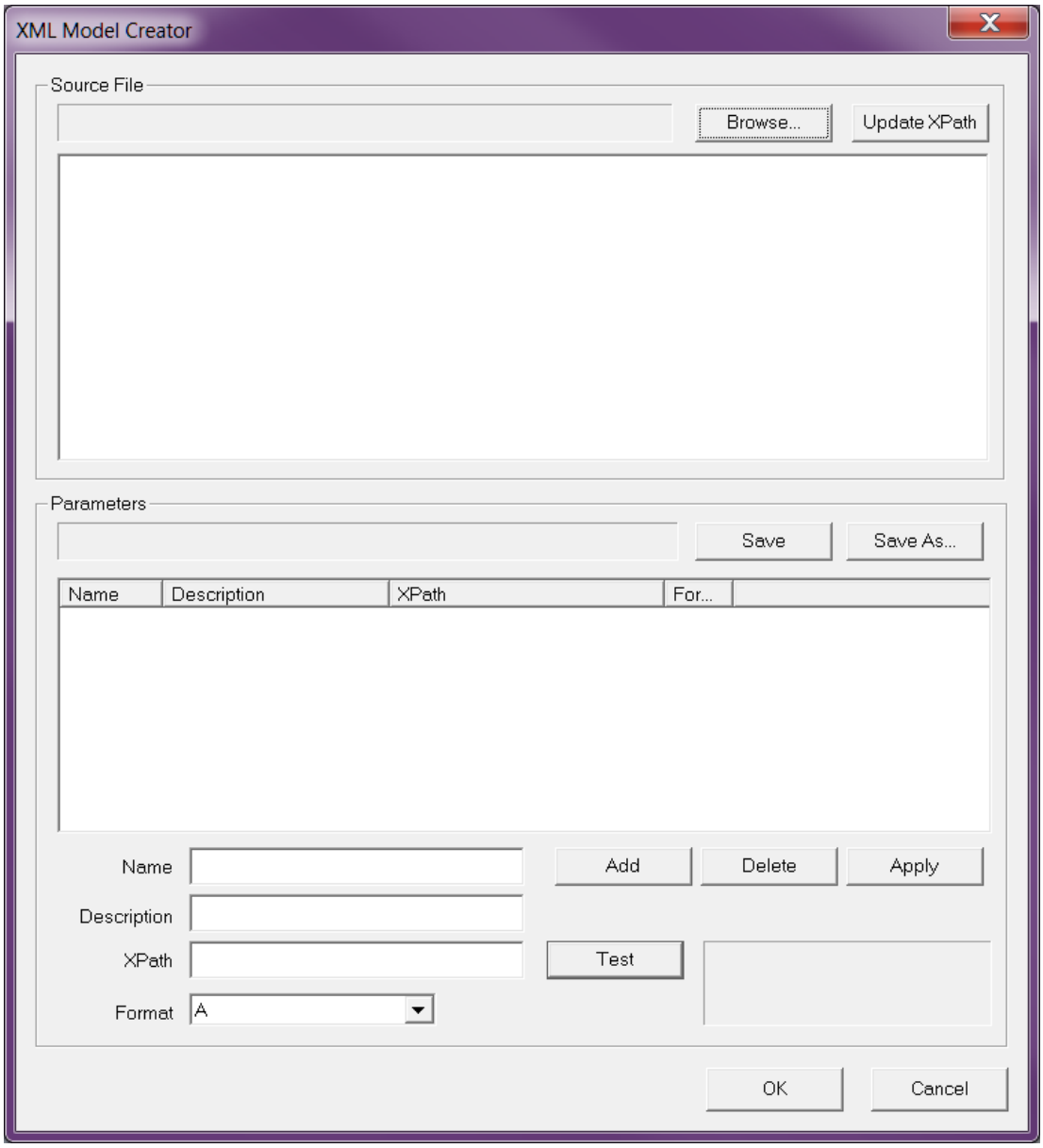
3.11.4.1 Prerequisites

- Some knowledge of the XML data for mapping between the XML data and CIMPortal Plus.

3.11.4.2 Description

The XML DCIM Model provides a mapping between the data in the XML file and the data the DCIM needs to report to CIMPortal Plus.

The XML DCIM Model Creator dialog looks similar to the following image:



The top portion of the dialog shows the currently selected XML file and a tree view of the XML it contains. The XML file does not have to be present in order to create the model information. It is required to test an XPath. Using the "Browse..." button allows a new XML file to be selected. The "Update XPath" button will put the XPath of the currently selected item into the XPath edit field at the bottom. Right-clicking the item in the XML tree view will also update the XPath. The XPath generator makes XPaths from the node names and may not generate correct paths for all possible XML items. The automatic XPath does not have to be used. The XPath is resolved using the Microsoft XML Parser. If multiple items are returned by the query, all the item values are concatenated together.

The lower portion of the dialog shows the current modeling data. At the top of the section, the last saved filename is displayed beside the Save and Save As... buttons. Clicking either of these buttons allows the current model information to be saved to a file. As with standard windows applications, the Save As... button allows a new filename to be selected and the Save button will use the last saved filename.

Just below the model filename is a list of all the currently defined parameters. Selecting a previously

defined parameter will cause the selected parameter's name, description, XPath, and format to show up in the edit field at the bottom. These values can then be changed. The Apply button causes the any changes to be put back to the parameter. The Delete button will delete all selected parameters. The Add button will create a new parameter from the values in the edit fields.

The test button will attempt to find the data from the Source XML file using the XPath defined by the edit field. If it is successful, the retrieved data will be shown in the text field to the right of the test button. If it is unsuccessful, an error message will be displayed in the same text field. The format is not applied to the retrieved data.

In the lower left corner all of the attributes needed to be defined a XML parameter are displayed. Each attribute will be described in detail below.

- **Name:** The name is text which usually describes the object. Using unique names is required. Names cannot contain line breaks, double quote ("), or pipe (|) characters. This name will be used by the EM Developer.
- **Description (optional):** The description is text which normally provides more details about the object. Descriptions cannot contain line breaks, double quote ("), or pipe (|) characters.
- **XPath:** The XPath describing the text to be extracted from the XML file.
- **Format:** The format of the data value that will be reported to CIMPortal Plus. The supported formats are: A (string value), I1 (signed 1 byte integer), I2 (signed 2 byte integer), I4 (signed 4 byte integer), I8 (signed 8 byte integer), U1 (unsigned 1 byte integer), U2 (unsigned 2 byte integer), U4 (unsigned 4 byte integer), U8 (unsigned 8 byte integer), F4 (4 byte IEEE floating point value), and F8 (8 byte IEEE floating point value).

After the model has been created, the dialog can be closed with either the OK or Cancel button. The OK button will put the source filename and model filename into the configuration dialog.

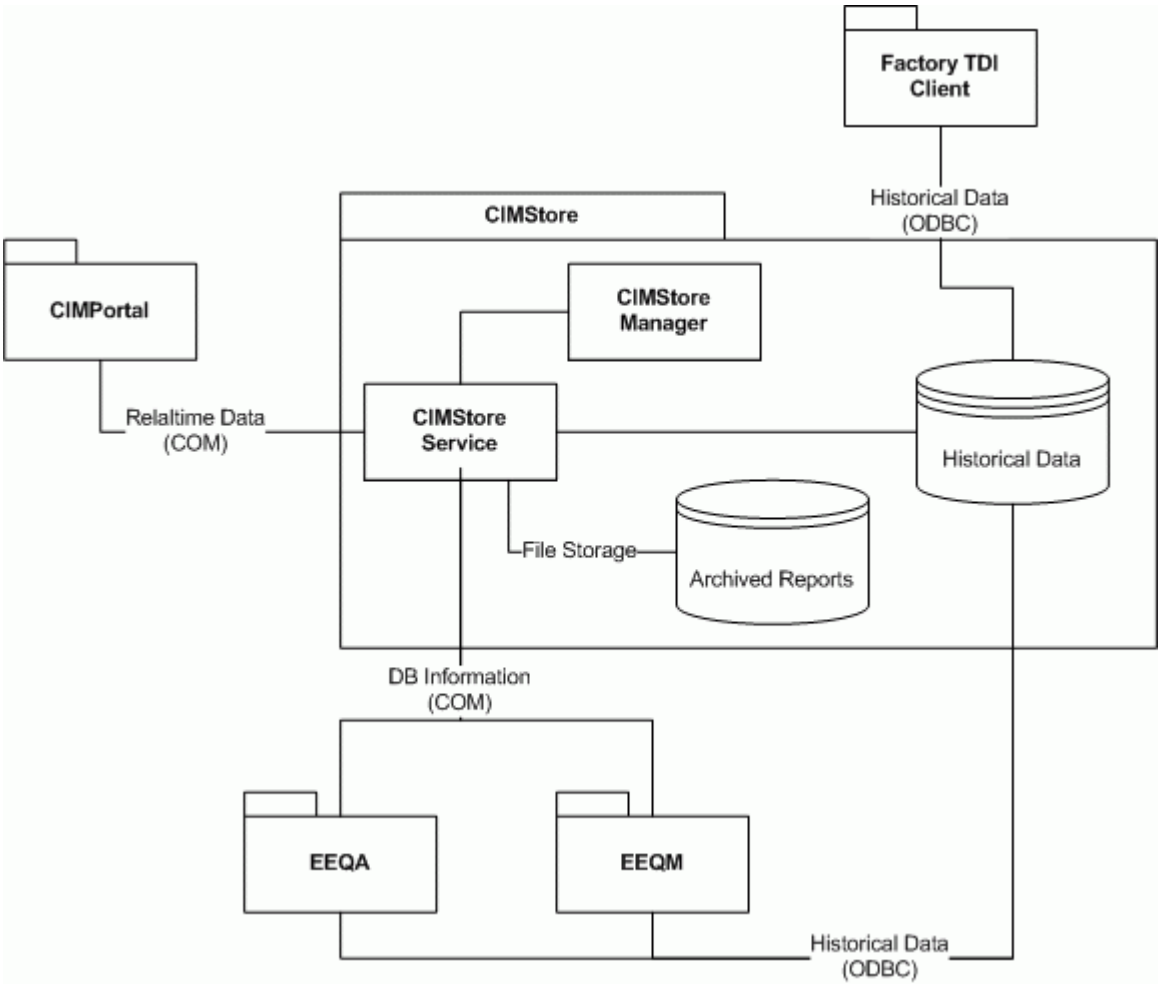
4. CIMStore

CIMStore provides database storage of data collected through CIMPortal Plus using the CIMPortal Plus High Speed Output interface. The data to be collected is defined using the CIMStore Manager in event requests, exception requests and trace requests. CIMStore Service will create the necessary database tables based on the requests defined using CIMStore Manager.

4.1 Architecture

CIMStore receives reports from CIMPortal Plus and stores them into tables in a Historical Data database. The data to be collected is defined using the CIMStore Manager in event requests, exception requests and trace requests. CIMStore Service will create the necessary database tables based on the requests defined using CIMStore Manager.

CIMStore is used anytime a historical data repository is required. The factory interface to retrieve data from the Historical Data database is ODBC.



The figure shows the following:

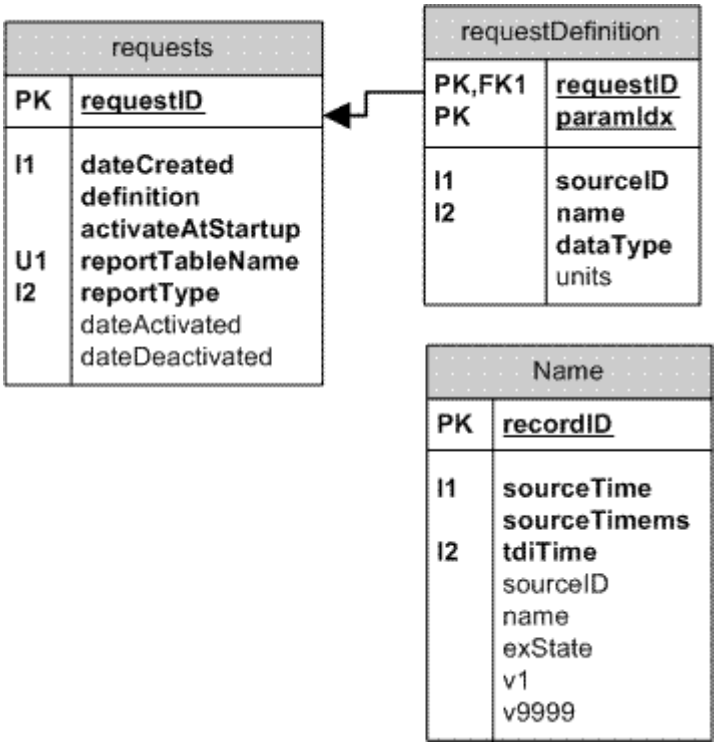
- **CIMStore Service**: The center-piece of the CIMStore product. It receives reports from CIMPortal Plus and stores individual values into fields in tables for the reports.
- **CIMStore Manager**: This UI allows a user to define the reports (event, exception and trace) necessary to collect the data to be stored into the Historical Data database.

- **Historical Data**: MS SQL, MySQL or Oracle database with multiple schemas (databases) - one for each CIMPortal Plus equipment controlled by CIMStore.
- **Archived Reports**: CIMStore deletes old report records from database to preserve database performance and data storage used by it. Deleted or Purged Reports may be archived into the text files. We suggest configuring the Archive Reports on a separate hard drive from Historical Data database. The archived files are not managed or deleted by CIMStore. It's your application's responsibility to maintain the Archive Reports.
- **CIMPortal Plus**: This is the Cimetrix EDA (i.e. Interface A) product that collects data from a piece of equipment.
- **EEQA**: This is an OEM Equipment Supplier's Equipment Engineering Quality Assurance application that is required by the TDI specification. The EEQA runs a specific scenario on the tool and compares the collected data to a previously accepted run.
- **EEQM**: This is an OEM Equipment Supplier's Equipment Engineering Quality Monitoring application that is required by the TDI specification. The EEQM monitors data over time looking for problems on the tool.

4.2 DB Schema

There will be one database in CIMStore for each deployed equipment in CIMPortal Plus that is added to CIMStore. These databases will have same schema and will be created by CIMStore after the CIMPortal Plus equipment is selected and as the requests are created by the user in CIMStore Manager.

Report timestamps in the database will be stored as 64 bit integer returned by the Windows API GetSystemTimeAsFileTime(). The timestamp is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). The resolution on most computers is 15.625 milliseconds (1/64 of a second).



4.2.1 requests Table

The requests Table is used to store information about requests.

Column	Data Type	Description
requestID	bigint(20) unsigned	Auto-incremented identifier of the request
dateCreated	bigint(20) unsigned	Timestamp of request creation
definition	text	ICxRequestDefinition CopyToString definition of the CIMPortal Plus service request
activateAtStartup	tinyint(1)	Boolean flag identifying wherever the request is automatically created and activated on CIMStore startup
reportTableName	varchar(25)	Name of the table containing the report data (see Report Data Table for more information)
reportType	tinyint(3)	Request type, 1-event, 2-exception, 3-trace
dateActivated	bigint(20) unsigned	Date and Time when the request was last activated in CIMPortal Plus service
dateDeactivated	bigint(20) unsigned	Date and Time when the request was last activated in CIMPortal Plus service

4.2.2 requestDefinition Table

The requestDefinition Table is used to store information about all parameters of a request – there will be one record per one parameter in the request.

Column	Data Type	Description
requestID	int unsigned	Identifier of the request, must match one record in requests Table
paramIdx	int unsigned	Index of the parameter inside of the request and the index of parameter value field in Report Data Table
sourceID	varchar(255)	E120 source ID of the parameter
name	varchar(64)	Name of the parameter object
dataType	varchar(32)	The SQL data type of the parameter's value field in the Report Data Table, i.e. varchar(64), bigint, double…
units	varchar(32)	Parameter Type Unit Name

4.2.3 Report Data Table

There will be multiple tables of this design, one for each defined request. The name of the table must match reportTableName in requests Table for corresponding request.

Column	Data Type	Description
recordID	bigint(20) unsigned	Auto-incremented identifier of the report
sourceTime	datetime	Timestamp of the event, exception or trace report
sourceTimems	smallint	Milliseconds component of Timestamp
sourceID	varchar(255)	E120 source ID of an Event or Exception, this column created only for requests of Event and Exception type (see reportType column in requests Table)
name	varchar(64)	Name of the Event or Exception that caused the report, this column created only for requests of Event and Exception type (see reportType column in requests Table)
exState	tinyint(3)	State of the Exception that caused the report, this column created only for requests of Exception type (see reportType column in

		requests Table)
vXXXX	See Report Parameter Data Type mapping section	Parameter value field, the XXXX in the column name is the paramIdx value in the requestDefinition Table for corresponding request (v1, v2… v9999 …). The SQL data type of the field is determined at the time of request definition and stored in dataType field of the requestDefinition Table

4.2.4 Report Parameter Data Type mapping

There are several ODBC functions such as SqlColAttribute() that may be used to determine the ODBC data type for each field directly from the database. As this mapping depends on the ODBC driver used, it is not included in this table and should be discovered.

CIMPortal Plus Model PTD base type (subtype)	SQL Database data type	Notes	Valid Value types
StringType, AnyURIType, EnumeratedType (EnumeratedString)	varchar or mediumtext	The type (varchar vs mediumtext) and the size of varchar is determined by user during creation of the Request in CIMStore Manager.	A(ASCII) or W(WCHAR)
DateTimeType	bigint unsigned		U8 or I8
BooleanType	tinyint(1)		Bo
ByteType, Base64BinaryType	tinyint	The parameter values of Base64BinaryType must be a single Bi value	I1,I2,I4,I8,U1,U2,U4,U8 for ByteType; Bi for Base64BinaryType Value must be between -128 and 127
ShortType	smallint		I1,I2,I4,I8,U1,U2,U4,U8 Value must be between -32768 and 32767
IntType, EnumeratedType (EnumeratedInt)	int		I1,I2,I4,I8,U1,U2,U4,U8 Value must be between -2147483648 and 2147483647
LongType	bigint		I1,I2,I4,I8,U1,U2,U4,U8 Value must be between -9223372036854775808 and 9223372036854775807
FloatType	float		F4 or F8 Value must be 0, 1.175494351e-38F .. 3.402823466e+38F, -3.402823466e+38F .. -1.175494351e-38F
DoubleType	double		F4 or F8

If any of the report parameters values do not comply with the Data Type mapping described in preceding table, the NULL value is stored into the database for that parameter.

4.3 Calculating Report Size

4.3.1 Description

This document describes how to calculate the size of a report based on the request. The formula is simple:

TotalSize = Report Data Size + Size of All Parameters + Database Overhead

4.3.2 Report Data Size

Each report has certain information associated with it regardless of whether the report is generated from an event request, exception request or trace request. This information has the following size:

Data	Size in Bytes	Description
recordID	8	Auto-incremented identifier of the report
sourceTime	8	Timestamp of the event, exception or trace report
sourceTimems	2	Milliseconds component of Timestamp
sourceID	255	E120 source ID of an Event or Exception, this column created only for requests of Event and Exception type
name	64	Name of the Event or Exception that caused the report, this column created only for requests of Event and Exception type
exState	1	State of the Exception that caused the report, this column created only for requests of Exception type

4.3.3 Parameter Size

Each parameter that will be collected in the report has a size based on the parameter type. The following table shows the size of each parameter type.

Parameter Type	Parameter Size in Bytes
StringType, AnyURIType, EnumeratedType (EnumeratedString)	Max character length for the string
DateTimeType	8
BooleanType, ByteType, Base64BinaryType	1
ShortType	2
IntType, EnumeratedType (EnumeratedInteger)	4
LongType, FloatType, DoubleType	8

Complex parameter types cannot be collected in reports for CIMStore. These types include:

- StructureType
- ArrayType
- VariableType

Add up the size of each parameter to be collected in the report.

4.3.4 Database Overhead

For each indexed column in the database, there is an additional 8 bytes of storage required. The indexed columns are:

- recordID
- sourceTime

For all text columns (if the parameter is of type StringType), add an additional 16 bytes.

Note: Due to fragmentation, the size required by the database may change when records are added and deleted.

4.4 CIMStore Service

4.4.1 Report Queue Handling

When a report for an active request is coming from CIMPortal Plus Service, CIMStore adds that report to a queue to be processed asynchronously and returns back to CIMPortal Plus. This is done so that CIMStore does not block CIMPortal Plus and so that other applications (like CIMWeb) may receive the data reports from CIMPortal Plus in a timely matter.

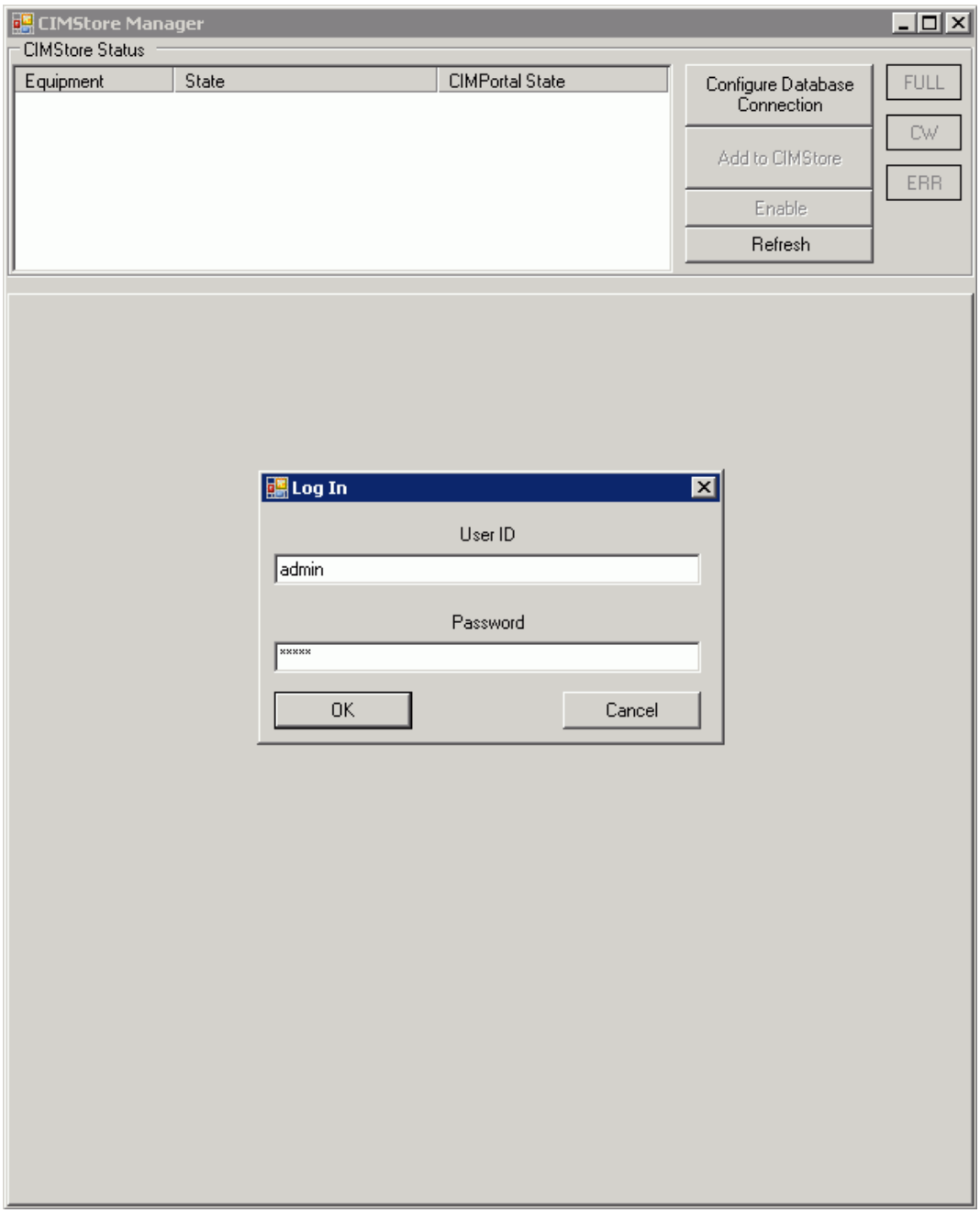
The data reports in this queue are processed in order they are received by a thread (there is a single thread per equipment), translated into SQL data and sent to the SQL database. The process of translating the data and storing it in the database takes time and it is possible that CIMStore could not handle the data reports coming from CIMPortal Plus as fast as CIMPortal Plus sends them. In this case the CIMStore's Report Queue would grow indefinitely until the Requests are deactivated or CIMStore process runs out of memory. In order to protect the CIMStore from this situation CIMStore has settings for dealing with Report Queue. These settings are exposed through API on ICxCSEqAdmin interface and can be changed in CIMStore Manager.

Name	Description
WarningReportQueueSize	Size of the CIMStore Report Queue at which CIMStore logs a warning message and calls CIMPortal Plus's ICxCPEstablishedSession::ClientError which in turn calls all registered application ICxCPEquipmentCB::EqClientError callbacks. This feature is turned off by setting it to 0 (zero).
MaxReportQueueSize	Size of the CIMStore Report Queue at which CIMStore logs an error message and, depending on MaxReportQueueSizeHandling setting, either deactivates all active Trace requests or discards all new reports from CIMStore until the Report Queue clears. This feature is turned off by setting it to 0 (zero).
MaxReportQueueSizeHandling	Specifies CIMStore's behavior when Report Queue size reaches the MaxReportQueueSize setting.

4.5 CIMStore Manager

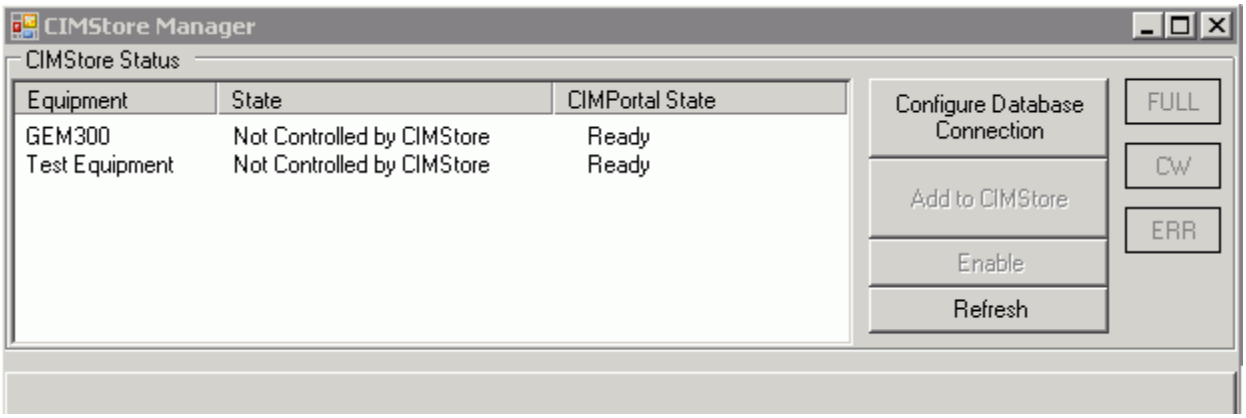
Cimetrix CIMStore Manager is Windows application that allows users to monitor the state and configure different aspects of CIMStore. CIMStore Manager was developed using Microsoft.NET technology and requires Microsoft .NET 2.0 Framework.

The CIMStore Manager is a secured program. Before accessing the main pages, the administrator user name and password must be entered in the login dialog. Users for CIMStore are setup using Cimetrix Configuration Manager tool (Please see the Configuration Manager Documentation for details).


CIMStore Manager
CIMStore Status
Equipment
State
CIMPortal State
Configure Database Connection
Add to CIMStore
Enable
Refresh
FULL
CW
ERR
Log In
User ID
admin
Password
*****
OK
Cancel

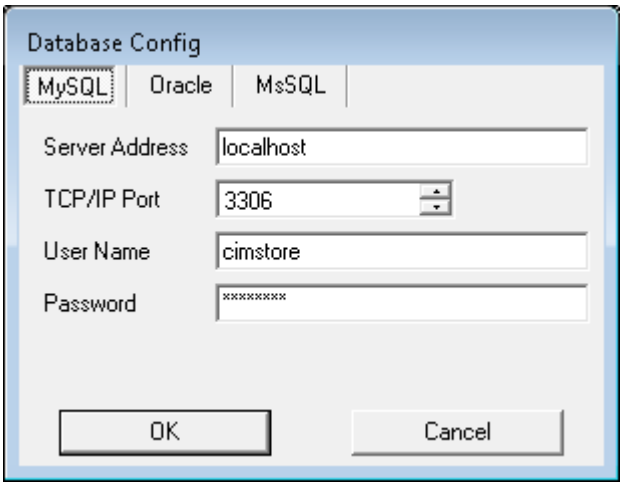
The CIMStore Manager's main view contains two sections: the CIMStore Status and Equipment Tab View.

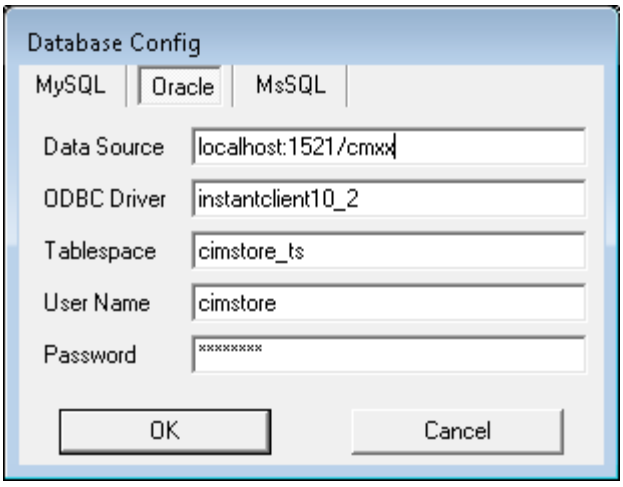
- CIMStore Status

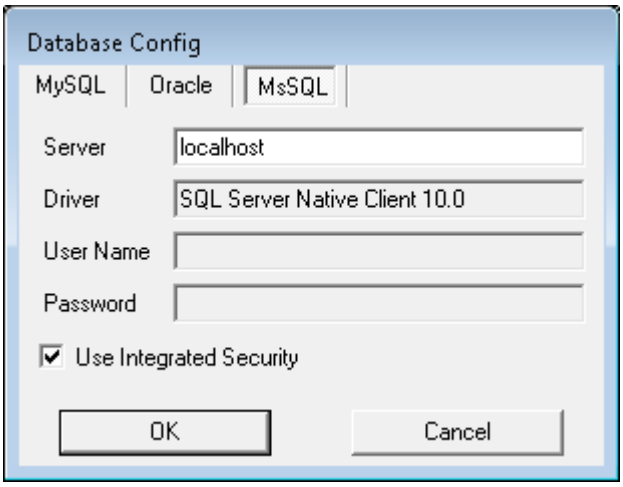


CIMStore Status displays the list of all available CIMPortal Plus equipment, regardless of the CIMPortal Plus state and wherever the equipment is controlled.

When running CIMStore Manager for the first time, the Database Config Dialog will be displayed automatically allowing Database Server connection for MS SQL, MySQL or Oracle to be configured. The Database Config Dialog can be also accessed using 'Configure Database Connection' button.







User specified for MySQL connection must have 'ALL PRIVILEGES' privilege, for Oracle - DBA role, for MS SQL – 'sa' privilege (see 'Installing CIMPortal Plus').

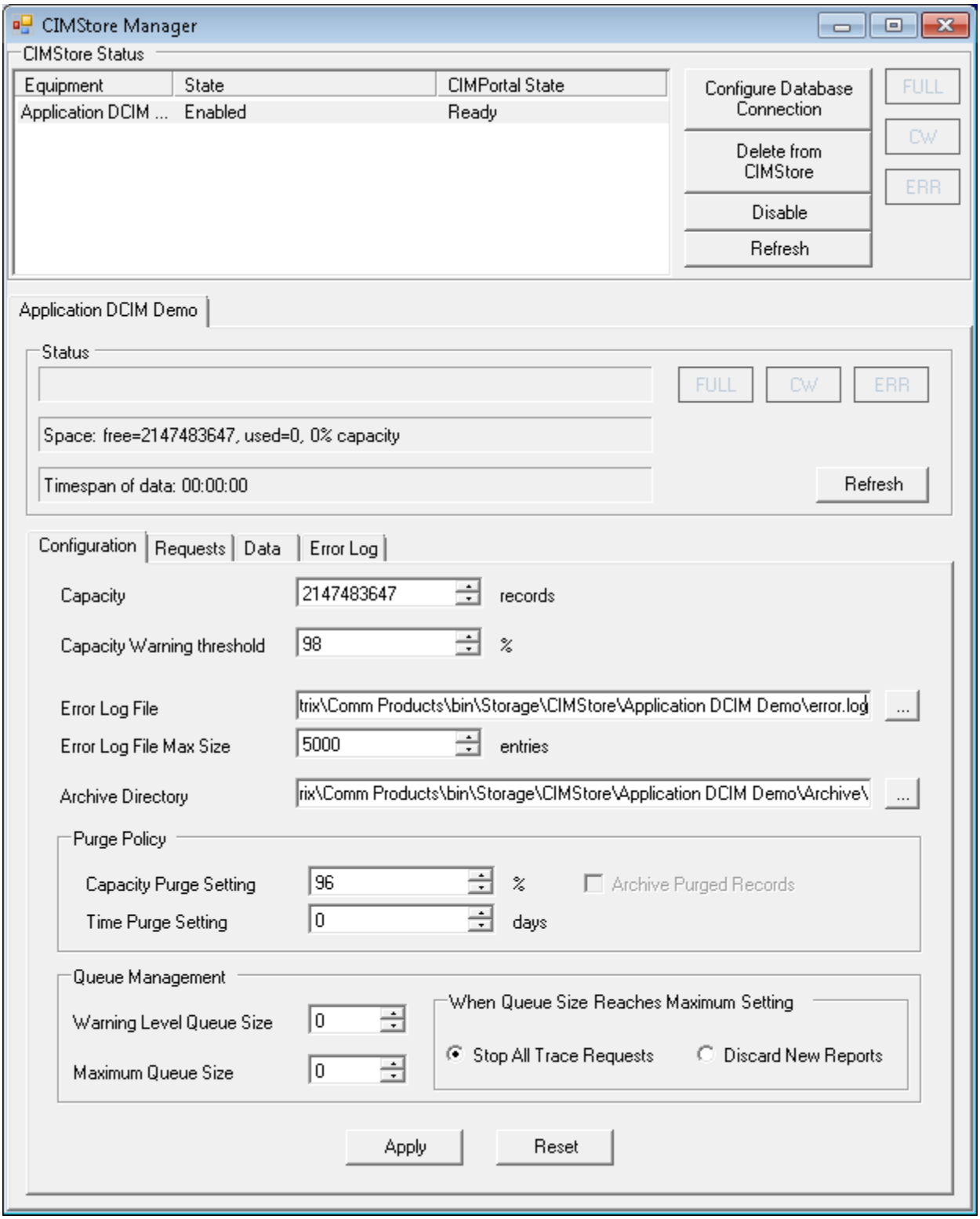
Note that whenever the Database connection is reconfigured, the equipment must be added to CIMStore (see next paragraph).

The equipment must be added to CIMStore using 'Add To CIMStore' button in order to be under the control CIMStore. When an equipment is added to CIMStore, CIMStore creates new schema (database in MySQLand MS SQL, user in Oracle) named 'CIMStore_<eq.name>' with 'requests' and 'requestDefinition' tables. If the database already exists in SQL server, that database is reused, and all request definition and data is preserved.

The equipment may also be deleted from CIMStore, which destroys the corresponding database tables, as well as any trace request, event request or exception request. CIMStore may also enable or disable its control over the equipment. Please note that the terms "enable" and "disable" mean whether the CIMStore acquires equipment data into its Historical Data database. Therefore enabling or disabling the CIMStore's control over the equipment does not affect the equipment's state in CIMPortal Plus. When CIMStore disables its control over a piece of equipment, any active trace request, event request or exception request are deactivated. These requests can no longer be deleted or activated until CIMStore enable the equipment again.

CIMStore Status view also contains three database state indicators: CW, FULL and ERR. These indicators change color as any equipment database reaches the respective state. CW means that the database capacity has reached its warning threshold. FULL means that the database has reached its maximum capacity. ERR means that there is a database related error. The capacity warning threshold and maximum capacity are configured on Equipment's Configuration tab.

- Equipment Tab View



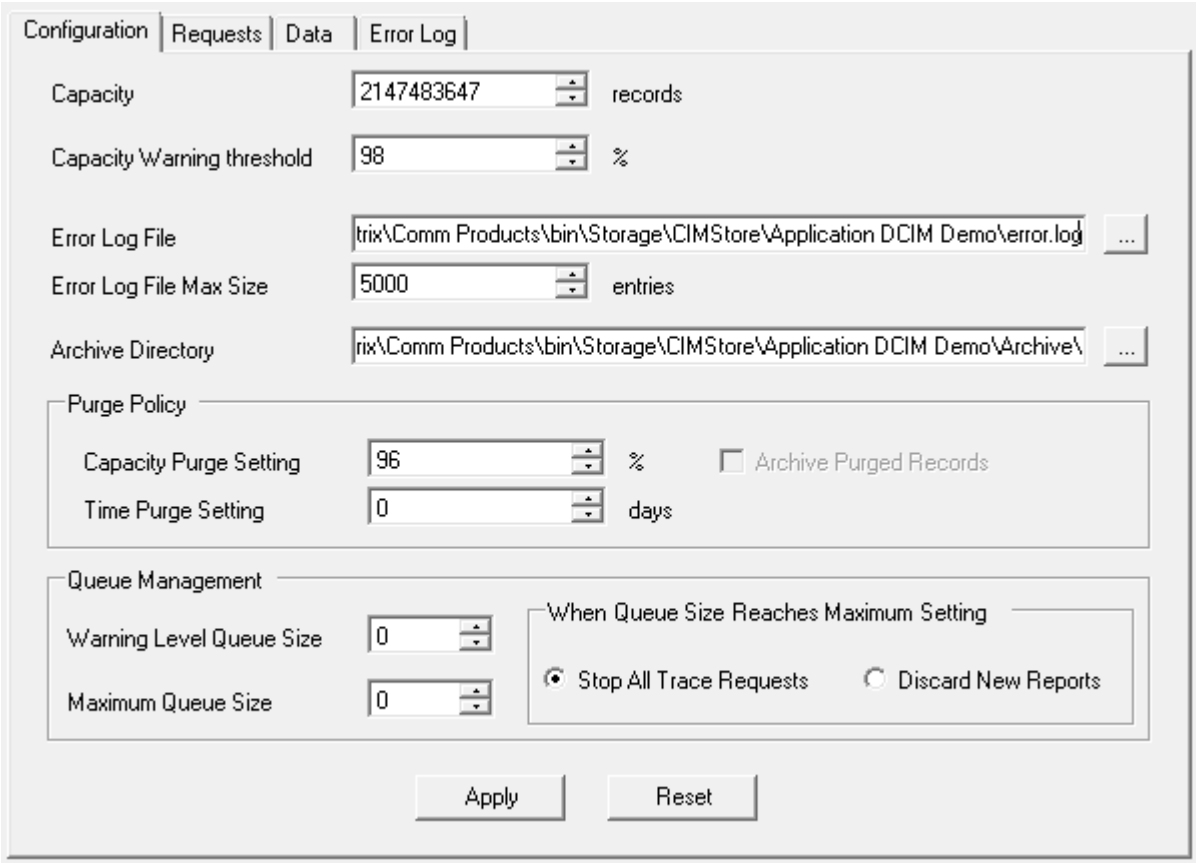
This view contains one tab page for each CIMPortal Plus equipment controlled by CIMStore.

The Equipment View contains Status View and following tabs:

- Equipment Configuration
- Requests
- Error Log
- Report Data

The Equipment Status View contains capacity indicator in form of progress bar, number of free and used records and the timespan of report data. This view also contains three database state indicators: FULL, CW and ERR.

4.5.1 CIMStore Equipment Configuration



Equipment Configuration tab contains the following configurable items:

- **Capacity** - Maximum number of report data records in the database
- **Capacity Warning threshold** - Turns CW indicator when database capacity reaches the specified percentage from the maximum capacity
- **Error Log File** - Location and the name of a file where all database related errors are written
- **Error Log File Max Size** - Maximum number of log entries to be written into log file
- **Archive Directory** - Folder location where CIMStore service will create files with report data after deleting from the database. It's recommended to specify this directory on a hard drive other than the one being used for database
- **Capacity Purge Setting** - When database capacity reaches the specified percentage from the maximum capacity, CIMStore service will start deleting old records

- **Time Purge Setting** - Delete report data records older than the specified range. Set to '0' to turn off. The two purge settings "Time Purge Setting" and "Capacity Purge Setting" are in a logical **or** relationship. That is, purge occurs if either condition is met.
- **Archive Purged records** - Tells CIMStore service to store records being deleted into files in the specified **Archive Directory** on this tab page.
- **Warning Level Queue Size** - Size of the CIMStore Report Queue at which CIMStore logs a warning message and calls CIMPortal Plus's ICxCPEstablishedSession::ClientError which in turn calls all registered application ICxCPEquipmentCB::EqClientError callbacks. Set Warning Level Queue Size to 0 to turn off. Warning Level Queue Size value must be less than Maximum Queue Size value.
- **Maximum Queue Size** - Size of the CIMStore Report Queue at which CIMStore logs an error message and, depending on MaxReportQueueSizeHandling setting, either deactivates all active Trace requests or discards all new reports from CIMStore until the Report Queue clears. Set Maximum Queue Size to 0 to turn off.
- **When Queue Size Reaches Maximum Setting** - Specifies CIMStore's behavior when Report Queue size reaches the MaxReportQueueSize setting:
  - **Stop All Trace Requests -** All active Trace requests are deactivated
  - **Discard New Reports -** All new reports from CIMPortal Plus are discarded until Report Queue is cleared

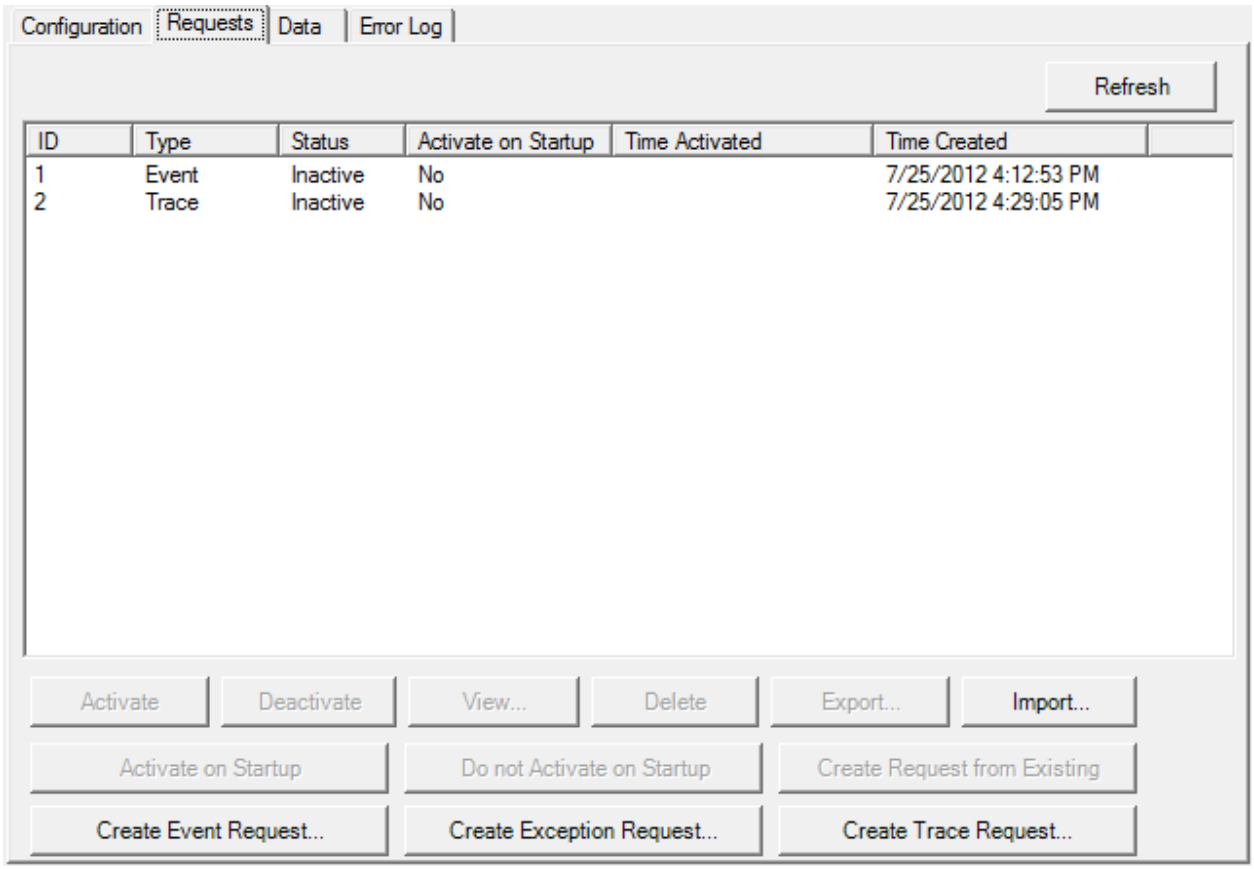
4.5.2 Requests

Requests are what make it possible for CIMStore to receive report data from CIMPortal Plus. There are three types of requests:

Type	Description
Event request	CIMStore requests CIMPortal Plus to generate a report whenever the specified event(s) occurs.
Exception request	CIMStore requests CIMPortal Plus to generate a report whenever the specified exception(s) changes to the specified "set" state or "clear" state.
Trace request	CIMStore requests CIMPortal Plus to generate a series of reports in a specified time interval whenever the specified trigger event(s) occurs or whenever the specified trigger exception(s) changes to the specified "set" state or "clear" state.

The user specifies in a request what parameters will be gathered in a report and the triggers to start the data gathering. When a request is created, it is stored in the CIMStore database. After a request is defined, it can be activated. If the conditions specified in an active request are met, CIMPortal Plus will generate a report (or reports). CIMPortal Plus sends the report(s) to CIMStore and CIMStore stores the report(s) to its Historical Data database.

CIMStore provides a Request tab to allow the user to define requests for equipment. When requests are defined, they are listed in the request tab similar to the following:

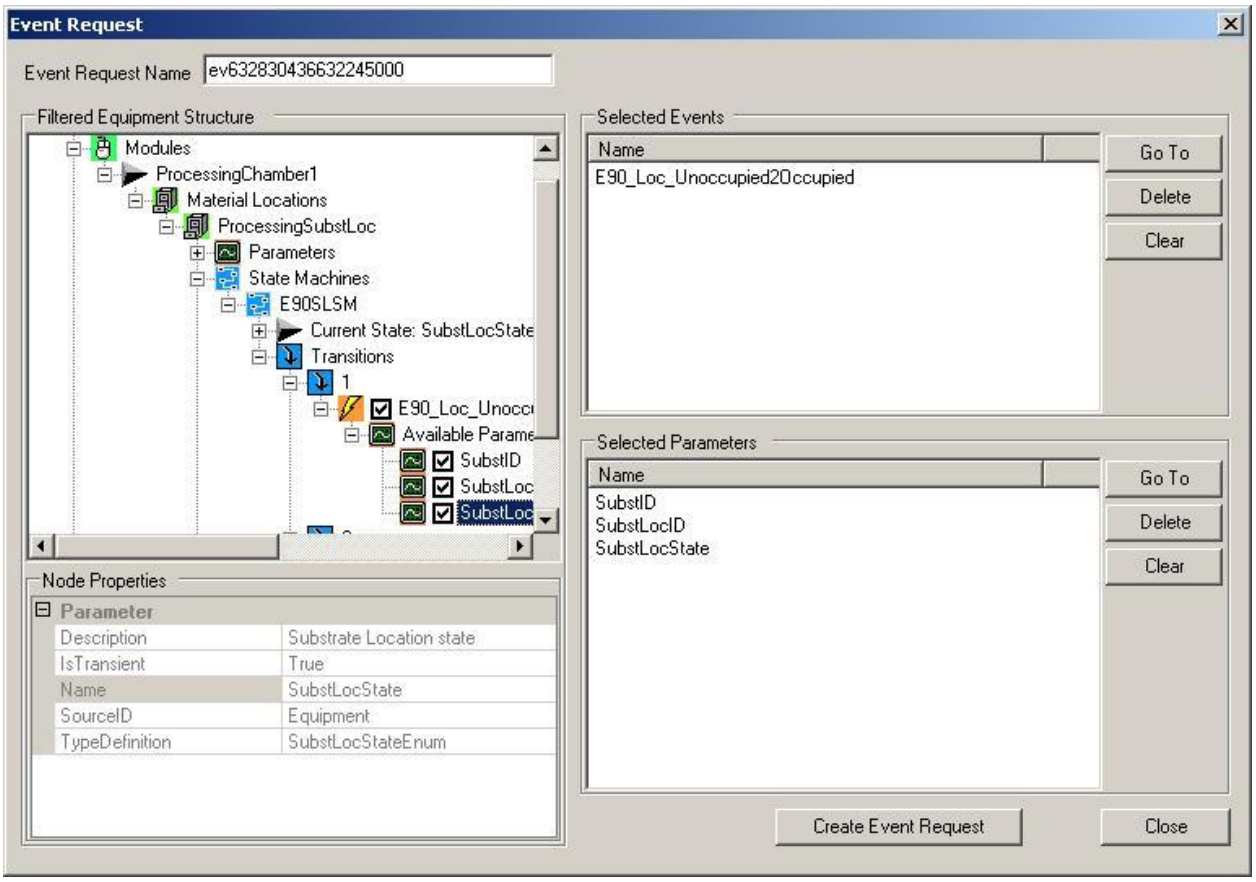


The list view displays:

- ID Request ID generated when the request was added to the database
- Type Event, Exception, Trace to indicate the type of request
- Status Active or Inactive
- Activate on Startup True or False indicating whether the request is activated or not when CIMStore starts
- Time Activated UTC time the request was activated. Empty if the request is Inactive
- Time Created UTC time the request was created

4.5.3 Create Event Requests

Pressing the Create Event Request... button on the Request tab displays the event request dialog.



The Filtered Equipment Structure displays the partial structure of the Equipment as it was queried from CIMPortal Plus service. The equipment model only displays branches of the model that contain parameters or events. If a branch does not contain a parameter or event, it is not displayed in the tree. Selecting any node in the tree displays its properties in the Node Properties at the bottom left of the dialog.

This tree view's context menu contains the following menu items:

Action	Description
Find	Brings up a 'Find Item' dialog to search for a node by its name in the model. The search may be case sensitive or case insensitive and looks for substring in node names. This dialog may be activated by using keyboard shortcut 'Ctrl+F'.
Find Next	Repeat the last search for a node inside the model. This action may be performed using the 'F3' keyboard shortcut.
Collapse Nodes	Collapses the currently selected node all of its child nodes in the model
Expand Nodes	Expands all the currently selected node and all of its child nodes in the model
Toggle Nodes	Changes the expanded/collapsed state of the currently selected node and all of its children in the model
Check Parameters	Traverses the currently selected node and all of its children in the model and checks the parameters that are not checked. These parameters are added to the Selected Parameters list.
Uncheck Parameters	Traverses the currently selected node and all of its children in the model and unchecks the parameters that are checked. These parameters are removed from the Selected Parameters list.
Toggle Parameters	Traverses the currently selected node and all of its children in the model and changes the state of each parameter encountered. If the parameter is changed to

	checked, it is added to the Selected Parameters list. If the parameter is changed to unchecked, it is removed from the Selected Parameters list.
Check Events	Traverses the currently selected node and all of its children in the model and checks the events that are not checked. These events are added to the Selected Events list.
Uncheck Events	Traverses the currently selected node and all of its children in the model and unchecks the events that are checked. These events are removed from the Selected Events list.
Toggle Events	Traverses the currently selected node and all of its children in the model and changes the state of each event encountered. If the event is changed to checked, it is added to the Selected Events list. If the event is changed to unchecked, it is removed from the Selected Events list.

- Event Request Name

The event request name is a human readable name for the event request. It must start with an alphabetic character and contain only alphanumeric characters.

- Selecting Events

An event request may have multiple events, but must have at least one. In the tree view, each event has a check box next to it that allows it to be selected. Once selected, the event is added to the Selected Events list. The events in this list are the ones that will be added to the request. The context menu items listed above may also be used to add to or remove events from the Selected Events list.

- Adding Parameters

In the tree view each parameter has a check box that allows it to be selected. Once selected, the parameter is added to the Selected Parameters list on the right side of the dialog. These are the parameters that will be collected into a report when the event is triggered.

- Selected Events/Parameters list

The Selected Events and Selected Parameters list views display all currently checked events and parameters in the model respectively. The order of items in the request may be re-arranged by using context menu or keyboard shortcuts.

Action	Description
Move Up	Shifts the current item up in the list. Keyboard shortcut 'Ctrl+U' may also be used.
Move Down	Shifts the current item down in the list. Keyboard shortcut 'Ctrl+D' may also be used.

The view contains following buttons:

Action	Description
Go To	Finds the currently selected item in the model tree-view
Delete	Removes currently selected item from the list. This action has the same effect as un-checking a check box next to the item in the model.
Clear	Removes all items from the list.

- Saving Event Requests

To add the event request to the CIMStore database, press the 'Save Event Request' button.

The event request is validated in the following areas:

- The event request name starts with an alphabetic character and contains only alphanumeric characters.

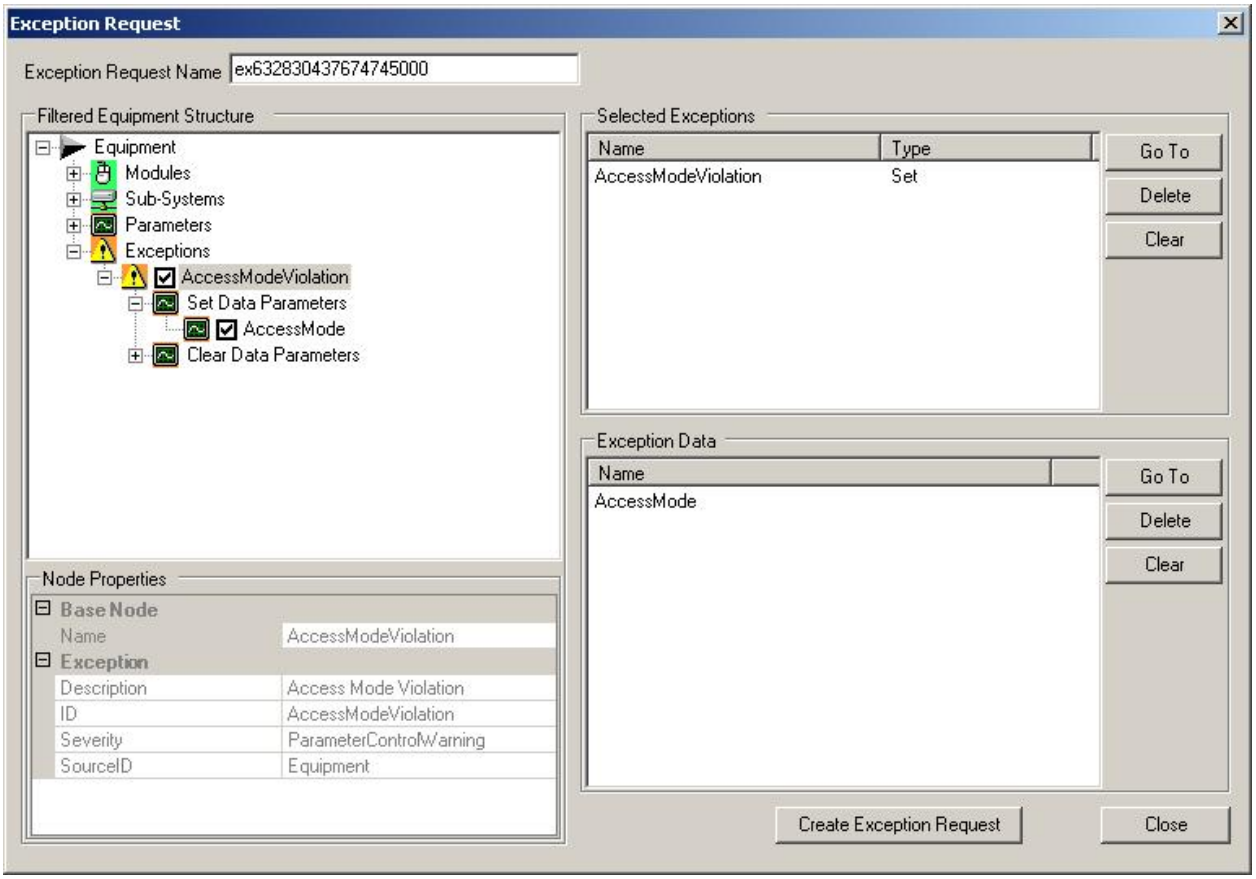
- The request has at least one event in the Selected Events list
- If a parameter is of type StringType and the max size is set to 0 or 255, the user will be prompted to enter the real max length for the parameter. This length may be applied to all other StringType parameters with a max size of 0 or 255.
- There are fewer than 1000 events and parameters in the request
- The total report size (based on parameter type size) is less than 64KB.
- The request is validated by CIMPortal Plus.

If any of these validations fail, the user is notified and the Event Request form remains visible.

If all validations succeed, an entry is added to the database and is displayed in the list view on the Request tab.

4.5.4 Create Exception Requests

Pressing the Create Exception Request... button on the Request tab displays the exception request dialog.



The Filtered Equipment Structure displays the partial structure of the equipment as it was queried from CIMPortal Plus service. The equipment model only displays branches of the model that contain exceptions and parameters. If a branch does not contain an exception or parameter, it is not displayed in the tree. Selecting any node in the tree displays its properties in the Node Properties at the bottom left of the dialog.

This tree view's context menu contains the following menu items:

Action	Description
Find	Brings up a 'Find Item' dialog to search for a node by its name in the model. The search may be case sensitive or case insensitive and looks for substring in node

	names. This dialog may be activated by using keyboard shortcut 'Ctrl+F'.
Find Next	Repeat the last search for a node inside the model. This action may be performed using the 'F3' keyboard shortcut.
Collapse Nodes	Collapses the currently selected node all of its child nodes in the model
Expand Nodes	Expands all the currently selected node and all of its child nodes in the model
Toggle Nodes	Changes the expanded/collapsed state of the currently selected node and all of its children in the model
Check Parameters	Traverses the currently selected node and all of its children in the model and checks the parameters that are not checked. These parameters are added to the Exception Data list.
Uncheck Parameters	Traverses the currently selected node and all of its children in the model and unchecks the parameters that are checked. These parameters are removed from the Exception Data list.
Toggle Parameters	Traverses the currently selected node and all of its children in the model and changes the state of each parameter encountered. If the parameter is changed to checked, it is added to the Exception Data list. If the parameter is changed to unchecked, it is removed from the Exception Data list.
Check Exceptions	Traverses the currently selected node and all of its children in the model and checks the exceptions that are not checked. These exceptions are added to the Selected Exceptions list.
Uncheck Exceptions	Traverses the currently selected node and all of its children in the model and unchecks the exceptions that are checked. These exceptions are removed from the Selected Exceptions list.
Toggle Exceptions	Traverses the currently selected node and all of its children in the model and changes the state of each exception encountered. If the exception is changed to checked, it is added to the Selected Exceptions list. If the exception is changed to unchecked, it is removed from the Selected Exceptions list.

- Event Request Name

The exception request name is a human readable name for the exception request. It must start with an alphabetic character and contain only alphanumeric characters.

- Selecting Exceptions

An exception request may have multiple exceptions, but must have at least one. In the tree view, each exception has a check box next to it that allows it to be selected. Once selected, the exception is added to the Selected Exception list. The exceptions in this list are the ones that will be added to the request. The context menu items listed above may also be used to add to or remove exceptions from the Selected Exceptions list.

By default, exceptions are added to the Selected Exceptions list with the type set to ‘Both’. This means that an exception report will be generated when the exception is both set and cleared. The type can be changed to ‘Set’ so that an exception report is only generated when the exception is set. It can also be changed to ‘Clear’ which generates an exception report only when the exception is cleared. The type of the exception is changed by:

1. Selecting the exception to change
2. Right-clicking in the Selected Exceptions dialog
3. Selecting the desired type from the context menu

- Adding Parameters

In the tree view each parameter has a check box that allows it to be selected. Once selected, the parameter is added to the Exception Data list on the right side of the dialog. These are the parameters that will be collected into a report when the exception is set or cleared.

- Selected Exceptions/Parameters list

The Selected Events and Exception Data list views display all currently checked exceptions and parameters in the model respectively. The order of items in the request may be re-arranged by using context menu or keyboard shortcuts.

Action	Description
Move Up	Shifts the current item up in the list. Keyboard shortcut 'Ctrl+U' may also be used.
Move Down	Shifts the current item down in the list. Keyboard shortcut 'Ctrl+D' may also be used.

The view contains following buttons:

Action	Description
Go To	Finds the currently selected item in the model tree-view
Delete	Removes currently selected item from the list. This action has the same effect as un-checking a check box next to the item in the model.
Clear	Removes all items from the list.

- Saving Exception Requests

To add the exception request to the CIMStore database, press the Save Exception Request button.

The exception request is validated in the following areas:

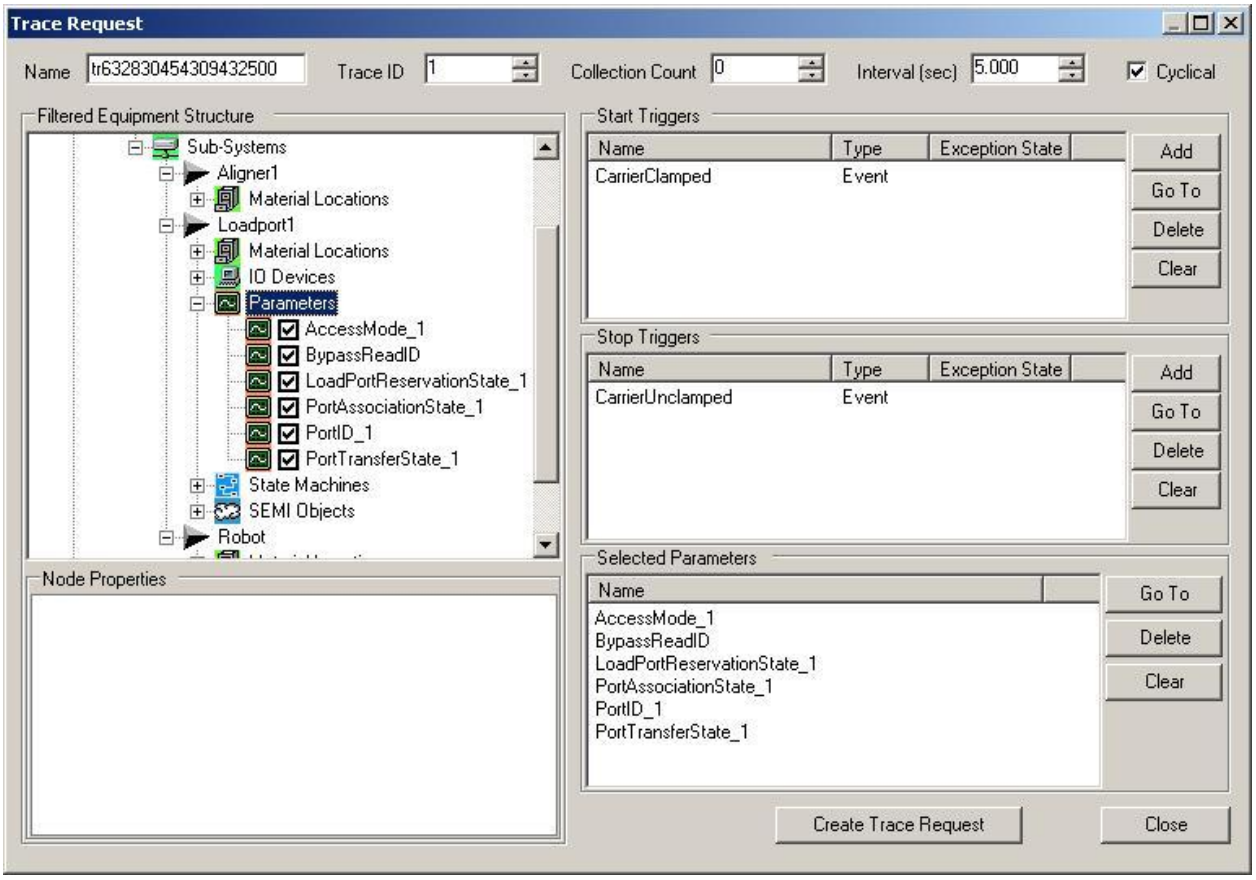
- The exception request name starts with an alphabetic character and contains only alphanumeric characters.
- The request has at least one exception in the Selected Exceptions list
- If a parameter is of type StringType and the max size is set to 0 or 255, the user will be prompted to enter the real max length for the parameter. This length may be applied to all other StringType parameters with a max size of 0 or 255.
- There are fewer than 1000 exceptions and parameters in the request
- The total report size (based on parameter type size) is less than 64KB.
- The request is validated by CIMPortal Plus.

If any of these validations fail, the user is notified and the Exception Request form remains visible.

If all validations succeed, an entry is added to the database and is displayed in the list view on the Request tab.

4.5.5 Create Trace Requests

Pressing the Create Trace Request... button on the Request tab displays the trace request dialog.



The Filtered Equipment Structure displays the partial structure of the equipment as it was queried from CIMPortal Plus service. The equipment model only displays branches of the model that contain parameters, events or exceptions. If a branch does not contain a parameter, event or exception, it is not displayed in the tree. Selecting any node in the tree displays its properties in the Node Properties at the bottom left of the dialog.

This tree view's context menu contains the following menu items:

Action	Description
Find	Brings up a 'Find Item' dialog to search for a node by its name in the model. The search may be case sensitive or case insensitive and looks for substring in node names. This dialog may be activated by using keyboard shortcut 'Ctrl+F'.
Find Next	Repeat the last search for a node inside the model. This action may be performed using the 'F3' keyboard shortcut.
Collapse Nodes	Collapses the currently selected node all of its child nodes in the model
Expand Nodes	Expands all the currently selected node and all of its child nodes in the model
Toggle Nodes	Changes the expanded/collapsed state of the currently selected node and all of its children in the model
Check Parameters	Traverses the currently selected node and all of its children in the model and checks the parameters that are not checked. These parameters are added to the Selected Parameters list.
Uncheck Parameters	Traverses the currently selected node and all of its children in the model and

	unchecks the parameters that are checked. These parameters are removed from the Selected Parameters list.
Toggle Parameters	Traverses the currently selected node and all of its children in the model and changes the state of each parameter encountered. If the parameter is changed to checked, it is added to the Selected Parameters list. If the parameter is changed to unchecked, it is removed from the Selected Parameters list.

- Adding Parameters

In the tree view each parameter has a check box that allows it to be selected. Once selected, the parameter is added to the 'Selected Parameters' list on the right side of the dialog. These are the parameters that will be returned in the trace report.

- Start/Stop Triggers

A trigger may be added to a list of Start or Stop Triggers by selecting an event or exception in the model view and clicking the Add button inside the Start Triggers or Stop Triggers box. For exception triggers the optional exception state may be specified by using context menu.

When all of the start triggers for a trace request reach the appropriate state (event triggered for events, exception state specified) the trace request starts generating reports. If no start trigger is specified, trace reports are immediately generated upon activation of the request.

When all of the stop triggers for a trace request reach the appropriate state, CIMPortal Plus stops generating reports for that request.

- Trace Request Properties

The trace request dialog has several other fields used for configuring the trace request. These fields are located across the top of the trace request dialog.

Field	Description
Name	The trace request name is a human readable name for the trace request. It must start with an alphabetic character and contain only alphanumeric characters.
Trace ID	Integer ID of the trace. This is not the same as the request ID displayed request list from the database
Collection Count	Number of reports to generate for this request. Zero is open ended.
Group Size	Indicates the number of reports to group together before sending out the reports.
Interval	The interval in seconds at which the reports will be generated for the request.
Cyclical	If a trace request is cyclical, it will automatically set itself up to be triggered again after the stop trigger occurs. If the trace request is not cyclical, the user must manually deactivate then reactivate the request for the start trigger to generate more reports.

- Selected Parameters list

The Selected Parameters list view displays all currently checked parameters in the model. The order of parameters in the request may be rearranged by using context menu or keyboard shortcuts.

Action	Description
Move Up	Shifts the current parameter up in the list. Keyboard shortcut 'Ctrl+U' may also be used.
Move Down	Shifts the current parameter down in the list. Keyboard shortcut 'Ctrl+D' may also be used.

The Selected Parameters, Start Triggers and Stop Triggers controls contain following buttons:

Action	Description
Go To	Finds the currently selected list item in the model tree-view
Delete	Removes currently selected item from the list. If the item is a parameter, it will be unchecked in the tree view as well.
Clear	Removes all items from the list.

- Saving Trace Requests

  To add the trace request to the CIMStore database, press the Save Trace Request button.

  The trace request is validated in the following areas:

  - The trace request name starts with an alphabetic character and contains only alphanumeric characters.
  - The request has at least one parameter in the Selected Parameters list
  - If a parameter is of type StringType and the max size is set to 0 or 255, the user will be prompted to enter the real max length for the parameter. This length may be applied to all other StringType parameters with a max size of 0 or 255.
  - There are fewer than 1000 events exceptions and parameters in the request
  - The total report size (based on parameter type size) is less than 64KB.
  - The request is validated by CIMPortal Plus.

  If any of these validations fail, the user is notified and the Event Request form remains visible.

  If all validations succeed, an entry is added to the database and is displayed in the list view on the Request tab.

4.5.6 Activate/Deactivate Requests

Once a request has been created and added to the database, the user can change the state of the request to make it active or inactive.

- Activate a Request
  1. Select the desired request in the list view on the Request tab (**Note:** Multiple requests may be selected at the same time)
  2. Press the Activate button

  If the request definition is invalid for any reason the error is reported and the request remains inactive. After the request is activated, it will start generating report data when the conditions or triggers of the request are met. The state of the request in the list box is changed from Inactive to Active and the activation time is displayed.

- Deactivate a Request
  1. Select the desired request in the list view on the Request tab (**Note:** Multiple requests may be selected at the same time)
  2. Press the Deactivate button

  Each selected request will be deactivated in CIMPortal Plus. The state of the request in the list box is changed from Active to Inactive and the activation time is cleared.

4.5.7 View Requests

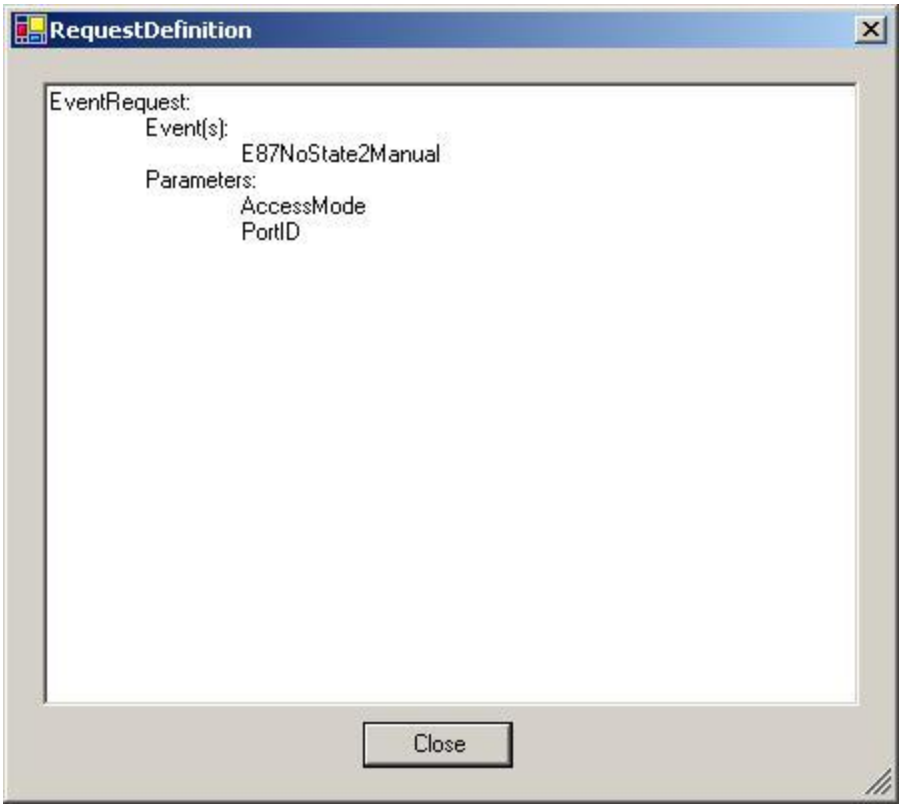
After a request has been created and stored in the database, it might be necessary or desirable to view the definition of the request.

To view request definition:

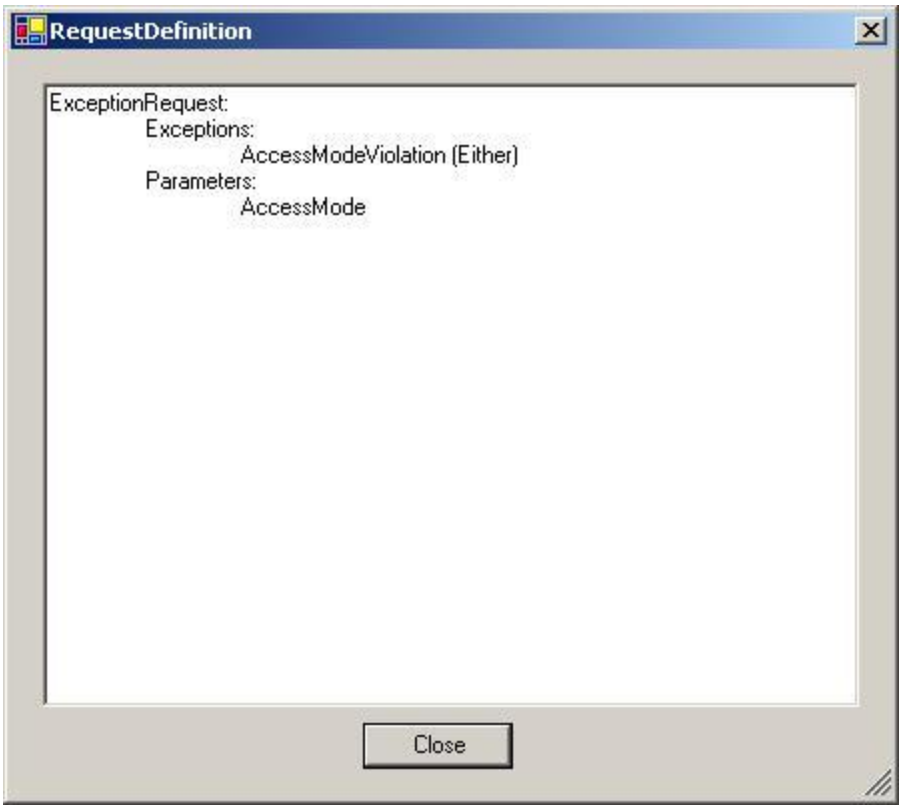
1. Select the request to view (only a single request may be viewed at a time)

2. Click the View... button on the request tab.

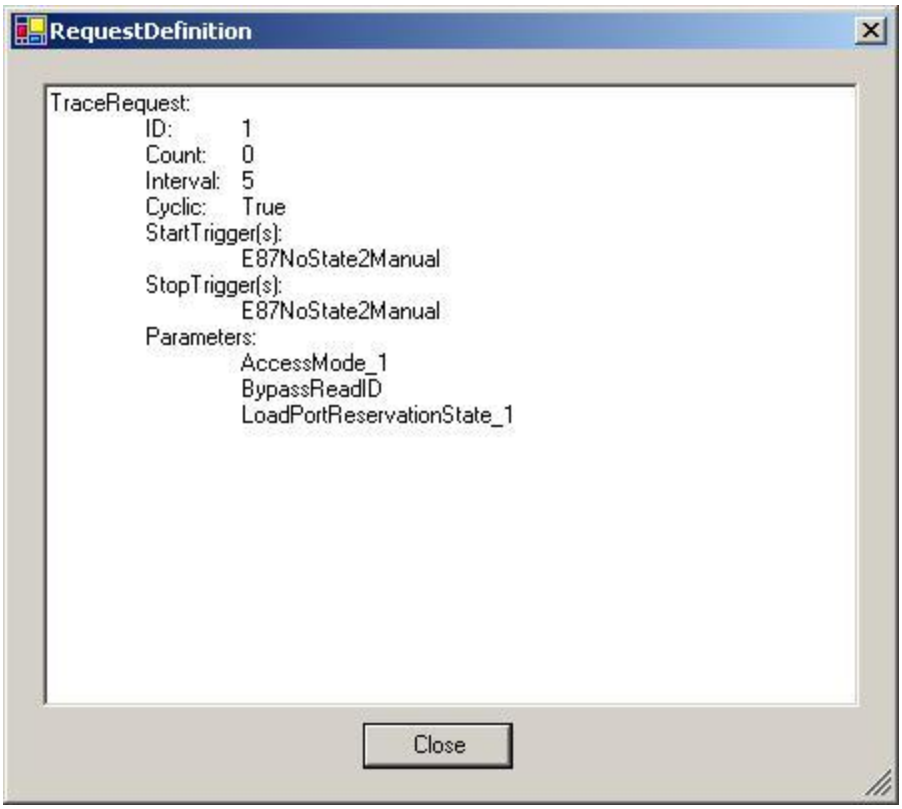
If the selected request is an event request, the definition might appear as follows:



If the selected request is an exception request, the definition might appear as follows:



If the selected request is a trace request, the definition might appear as follows:



4.5.8 Delete Requests

To delete a request:

1. Select the desired request in the list view on the Request tab (Note: Multiple requests may be selected at the same time)
2. Press the Delete button

Each selected request will be removed from the CIMStore database and from the list view on the request page. If a request to be deleted is active, it will be deactivated and then deleted.

4.5.9 Activate Requests on Startup

After a request has been added to the database, it can be set to be activated automatically when CIMStore starts.

- Activate on Start

  To set a request as active on startup:

  1. Select the desired request in the list view on the Request tab (**Note:** Multiple requests may be selected at the same time)
  2. Press the 'Activate on Startup' button

  The Activate on Startup field of the request will change to true. That request will activate automatically when CIMStore starts.

- Inactive on Start

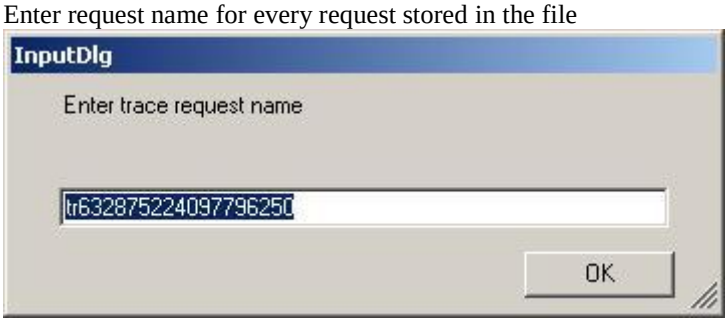
  To set a request as inactive on startup:

1. Select the desired request in the list view on the Request tab (**Note:** Multiple requests may be selected at the same time)
2. Press the 'Do not Activate on Startup' button

The Activate on Startup field of the request will change to false. That request will no longer activate automatically when CIMStore starts.

4.5.10 Export and Import Requests

- Export Requests
  1. Select the desired request in the list view on the Request tab (**Note:** Multiple requests may be selected at the same time)
  2. Press the 'Export...' button
  3. Select file name and location for the Request Definition File and click 'Save'
- Import Requests
  1. Press the 'Import...' button
  2. Browse for the Request Definition File and click 'Open'
  3. Enter request name for every request stored in the file



4.5.11 CIMStore Equipment Setup

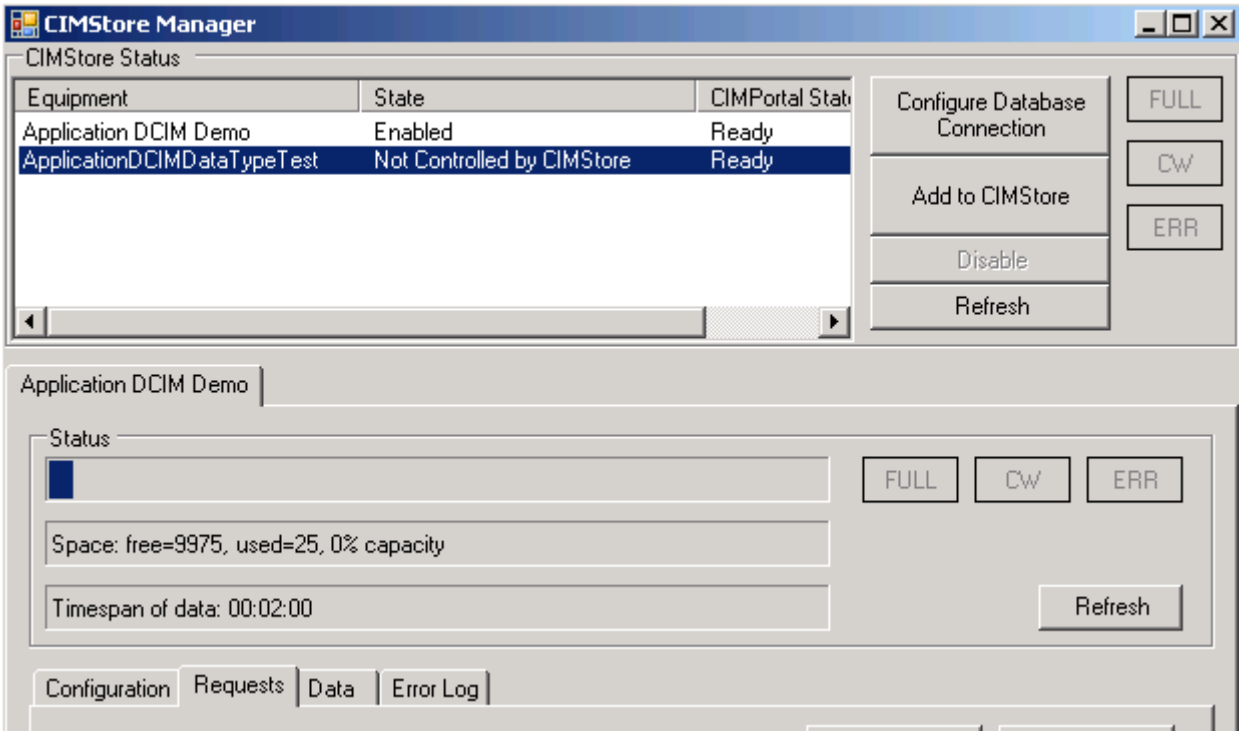
After equipment has been deployed in CIMPortal Plus, then CIMStore has the ability to collect data from that equipment. This document describes how to get started.

- Prerequisites
  - CIMStore's connection to the database has already be configured and established.
  - Equipment .pkg file has been deployed in CIMPortal Plus.
- Setup CIMStore to Collect Data

Initially, CIMStore will not be setup to collect any data from deployed equipment. In order to set up data collection follow these steps:

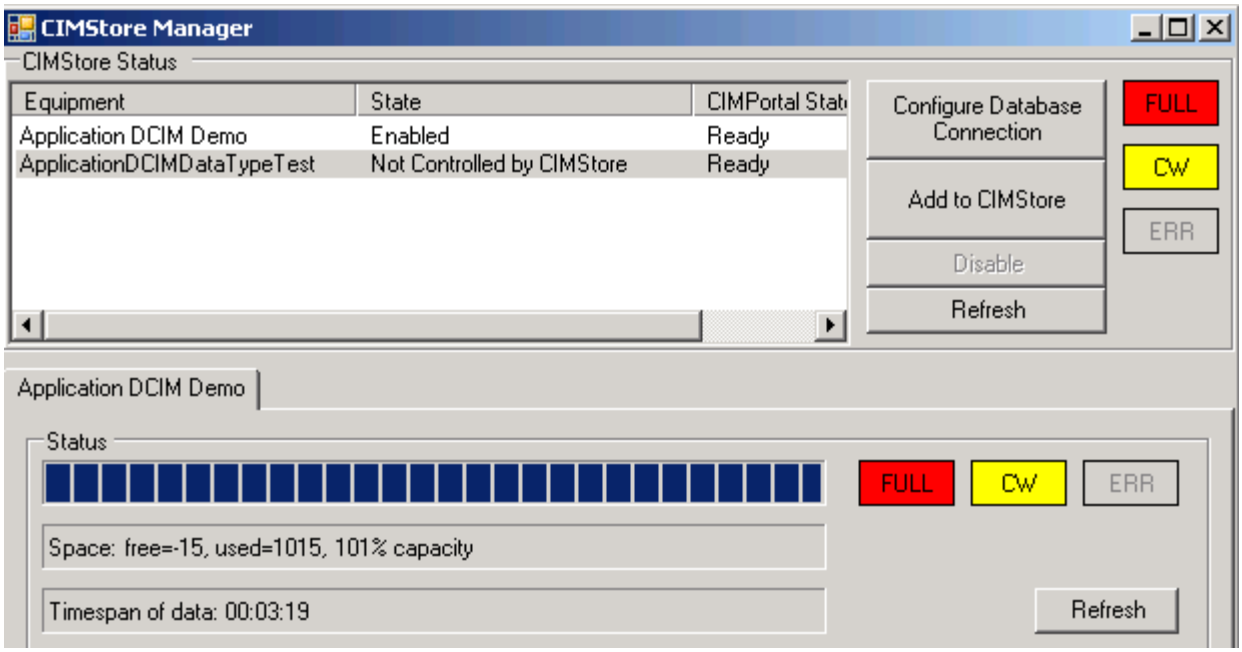
1. Run CIMStore Manager
2. Select the equipment from the list so that it is highlighted. Typically, there will only be one.
3. Click on the Add to CIMStore button. If the equipment had previously been added to CIMStore, then the button will have the message Delete From CIMStore.

CIMStore Manager will create a new tabbed window in the lower portion of its window. For example, in this image, there are two deployed equipment. One, the Application DCIM Demo, has been added to CIMStore. The other has not.



- View the Data Collection Setup

Once equipment has been added to CIMStore, then it is possible to configure the database, create and activate data collection requests. The data collection status is displayed in the top portion of the equipment's tabbed window. Following is a description of each field.



- **Refresh**
  Click on the refresh to update all of the status fields.

NOTE: When there is a capacity warning, then the display will refresh automatically at a high frequency until the capacity warning is cleared. It is best to configure capacity purge setting such that capacity warnings do not normally occur.

- **Status**

  This is a visual indication of the database capacity. This is relative the database configuration.

- **Space**

  The space text field is indicates how many of the allocated capacity records are free (available for new data) and used (consumed by gather data). The text field also lists the current percentage of capacity the existing data consumes.

- **Timespan**

  The timespan field indicates in hours, minutes and seconds the duration of time spanned by the collected data.

- **Full**

  By default the Full indicator has a gray color. If the database fills to capacity, then the Full indicator will be red.

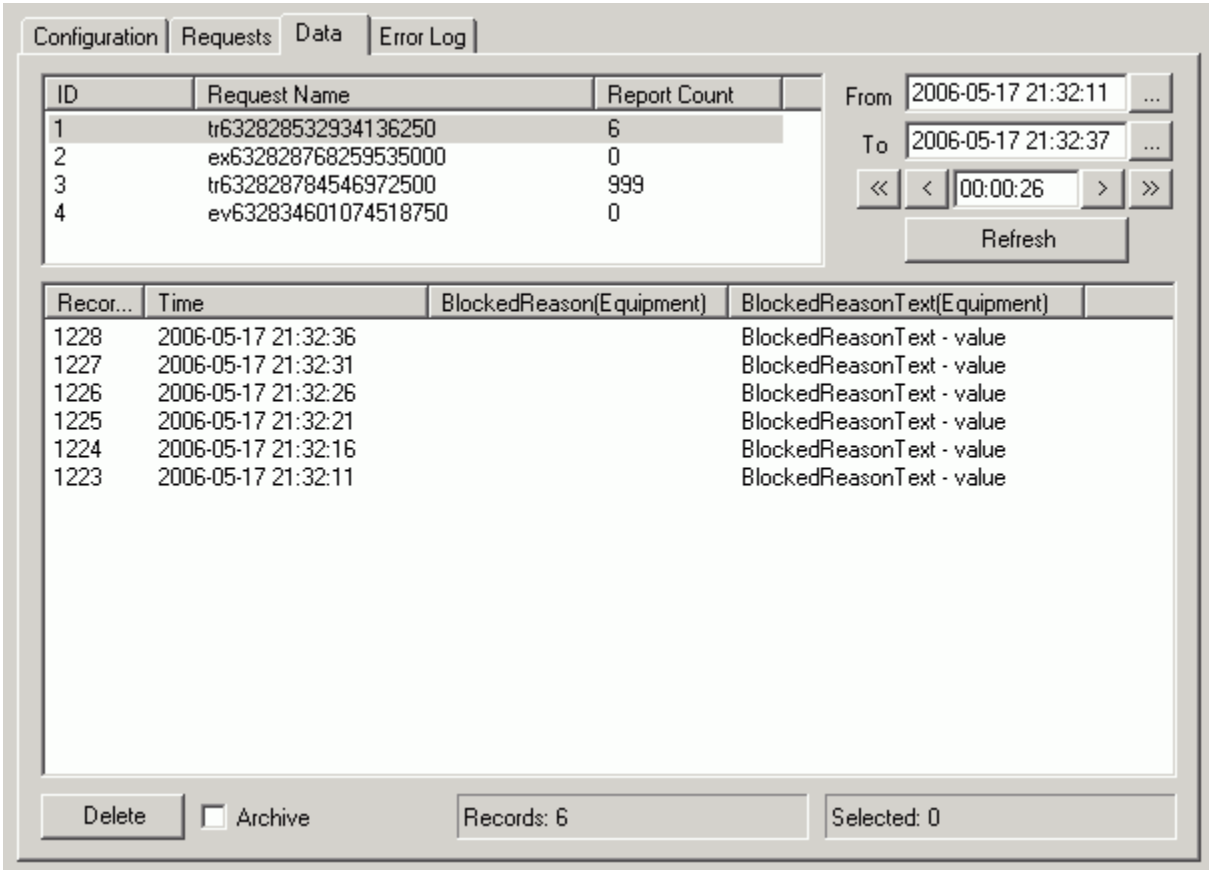
- **CW**

  By default the Capacity Warning (CW) indicator is a gray color. If the database fills beyond the capacity warning threshold, then the CW indicator will be yellow.

- **ERR**

  By default, the Error (ERR) indicator is a gray color. If the database returns an error when calling its API functions, then the error indicator will change color.

4.5.12 CIMStore Report Data View



Data tab allows users to view and delete report data records.

In the top left corner of the Data tab the list of all defined requests is located. This list displays all defined requests with its ID, name and total number of report records.

When a request is selected, the report data view is filled with approximately 100 latest report records, bound to the nearest second of the report's time. The 'From', 'To' and timespan fields in the top right corner of the Data tab are initialized to maximum, minimum and the timespan for the displayed report timestamps. The number of records in the current view is displayed on the bottom of the Data tab in 'Records: {number}' status view.

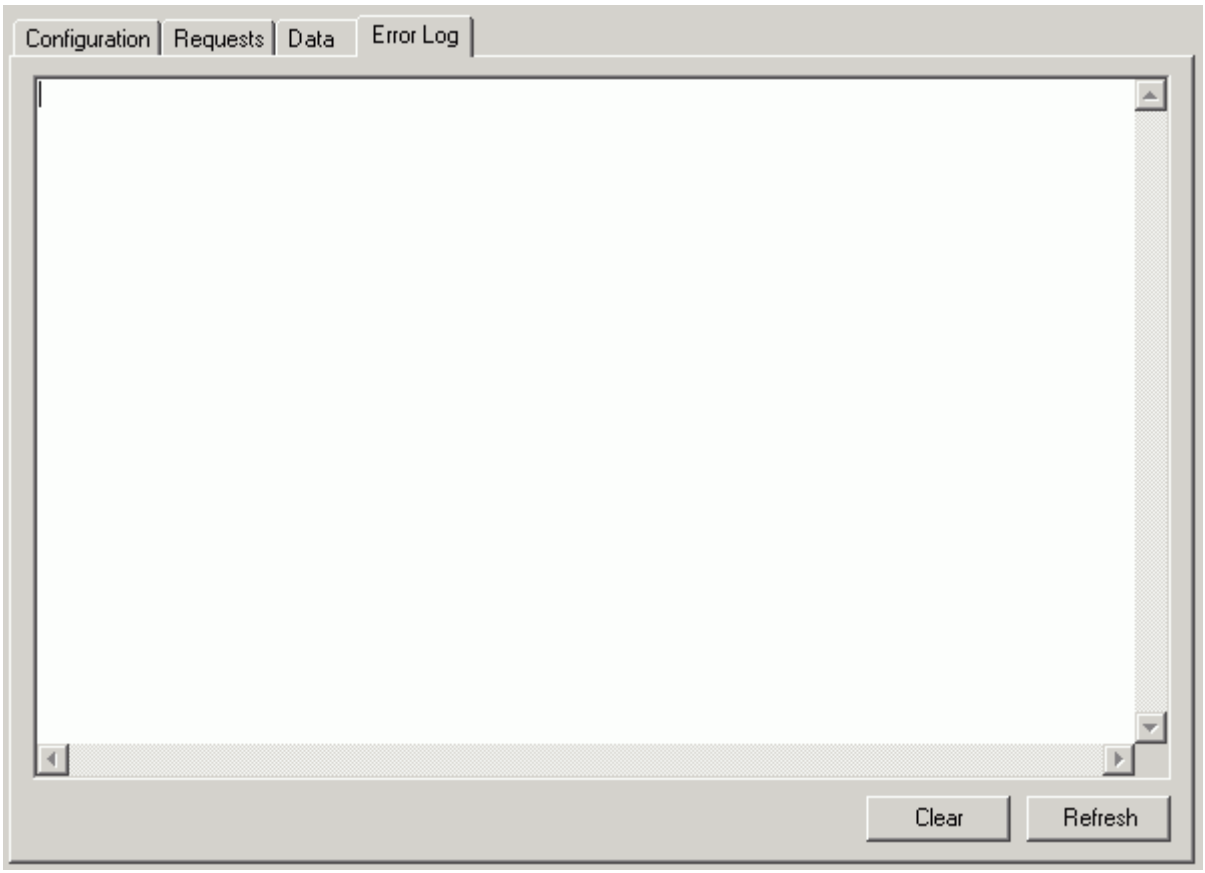
The report records may be navigated using the << - First, < - Previous, > - Next and >> - Last navigation buttons. The Previous and the Next buttons move the current view to the previous or the next set of report records in the database within the specified timespan. The First and the Last buttons move the current view to the beginning or to the end of the report table.

User may manually choose the 'From' - the earliest and/or 'To' - the latest report record times to display after clicking on 'Refresh' button.

To delete records user first must select report records from the list first. The number of selected records is displayed on the bottom of the Data tab in 'Selected: {number}' status view. Once records are selected, click on 'Delete' button to delete selected records. If desired, the deleted records may be archived by checking the 'Archive' checkbox before click the 'Delete' button.

To delete all records of a request or set of requests, select one or more requests from the request list view instead of selecting records from the report data view and then use the 'Delete' button. Note that when multiple requests are selected, the report view will be left empty.

4.5.13 CIMStore Database Error Log



Error Log tab displays the database related errors. CIMStore uses 'Error Log File' and 'Error Log File Max Size' configuration items to determine where and how many error log entries to keep.

Click on 'Refresh' button to display latest error log or "Clear' to clear this view as well as the log file.

4.6 Archive File Format

Cimetrix CIMStore has ability to archive purged/deleted records into files. This document describes the format of these files.

The archive file always has 3 lines:

1. Report information
   - Event Reports:
     <timestamp>, <request type>, <source id/event id>
   - Exception Reports:
     <timestamp>, <request type>, <source id/exception id(set/clear flag)>
   - Trace Reports:
     <timestamp>, <request type>

   Where timestamp is 64 bit integer as returned in the Windows API GetSystemTimeAsFileTime();
   request type - one of three values: 1-event, 2-exception, 3-trace;
   set/clear flag - 1-set, 2-clear.
2. Request parameter definitions: <source id/parameter name>, <source id/parameter name>, ...
3. Comma-delimited parameter values

Because values are comma-delimited, the string parameter values must not have commas.

5. Using the ISMI Metadata Conformance Analyzer (MCA)

5.1 What is MCA?

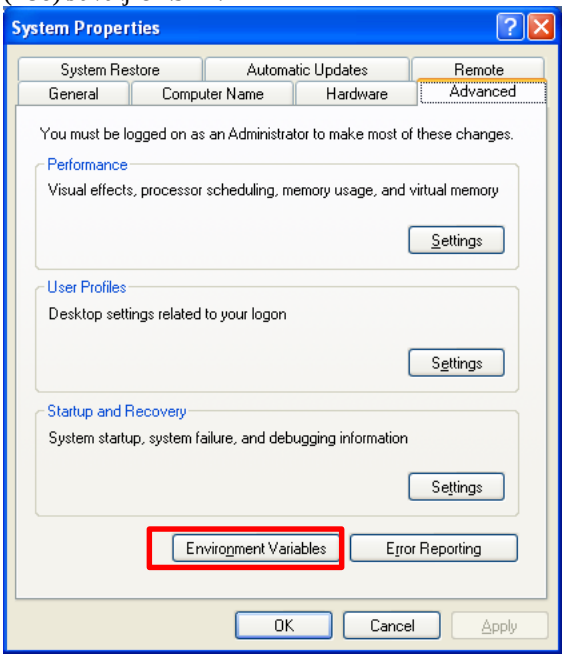
The Metadata Conformance Analyzer tool is an automated means to check for conformance of equipment metadata to SEMI EDA standards as well as the ISMI guidelines for model building. The tool is available at no charge from the ISMI website:
http://www.sematech.org/mtc/tools/metadata/conformance_analyzer/index.htm.
From this page, you may choose to download either the 0710 version or the 1105 version of the tool.

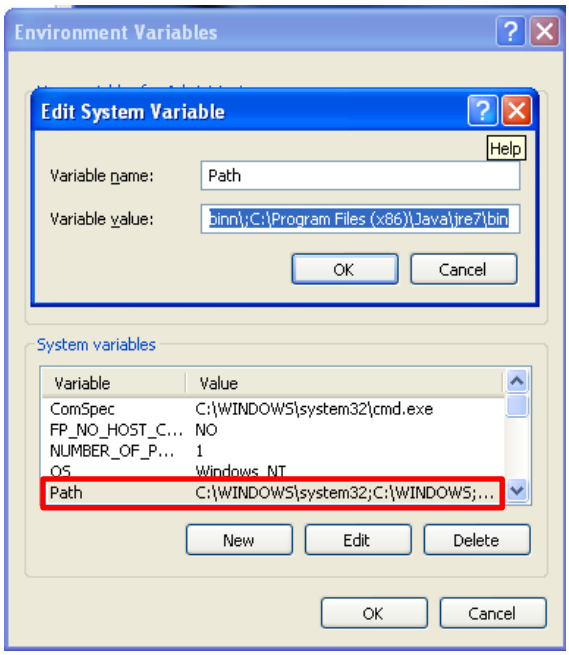
The ISMI model building guidance document is *Equipment Data Acquisition (EDA) Version 0710 Guidance* (Technology Transfer #10115124A-ENG) and is also freely available at
http://www.sematech.org/docubase/document/5124aeng.pdf.

The MCA verification process validates the model structure and content against the 0710 standards (including the new E164 standard) and the advice of the guidance document mentioned above. Note that the E164 standard basically replaces the 0710 Guidance document, but the document is still useful due to its detailed descriptions.

5.2 Setting Up MCA

Get the latest Java runtime for your operating system from http://java.com/en/download/index.jsp and install it. Right click "My Computer" and go to "Properties". Click the "Advanced" tab then click "Environment Variables". Under the "System Variables" group, find the "Path" variable, highlight it then click "Edit". Add the following text to the end of the "Variable value" field, without quotations: "C:\Program Files (x86)\Java\jre7\bin".



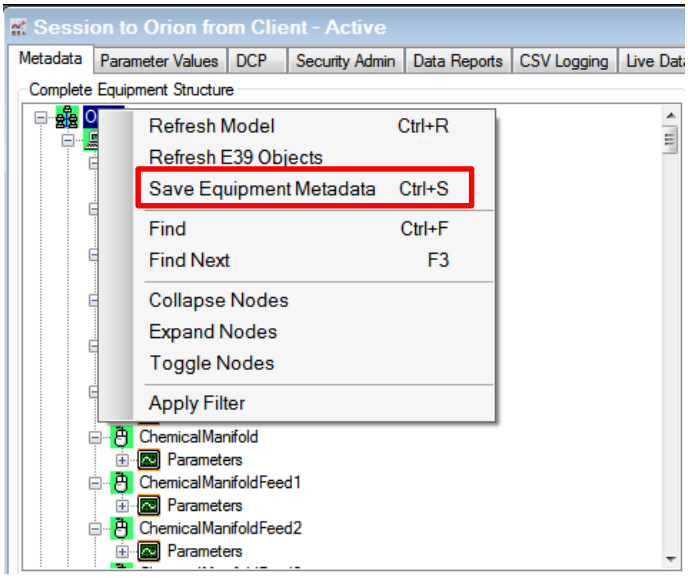


Install MCA.

5.3 Collecting Metadata

In order to collect metadata to be analyzed by MCA, users should go through the following steps:

1. **Create a valid equipment model package**. Using EMDeveloper, the user creates the desired model and completes final validation from the Packaging Menu. Once the model is valid, the user creates the equipment package by selecting Create Deployment Package from the Packaging menu and following the prompts to save the .pkg file.
2. **Deploy the equipment model into CIMPortal Plus**. From the CIMPortal Plus control panel's Configuration tab, the user presses the Deploy button and selects the model package created in the previous step. The user follows the prompts until the model is deployed and the State is set to ready.
3. **Set privileges for the model**. Once the equipment is ready, the user can change to the desired CIMWeb tab (CIMWeb0710 or CIMWeb1105) and the Access Control sub-tab. On the Access Control tab, the user should enter a Principal Name (similar to a login name that will be used to attach to the equipment) and check the boxes next to E132 All Privileges and E132 Admin Privilege.
4. **Launch ECCE and save the metadata**. Connect to the equipment using ECCE. If ECCE has not been used to connect to this specific equipment before, the user will need to create a new session definition using the Principal Name from the previous step as well as E125, E132 and E134 location URLs that can be obtained from the Addresses tab in the CIMPortal Plus Control Panel. Once a session definition has been created, the user connects to the equipment. Once the ECCE client has connected and the equipment metadata appears on the Metadata tab, right click the equipment node and select "Save Equipment Metadata."



Select the directory to store the metadata then click "OK."

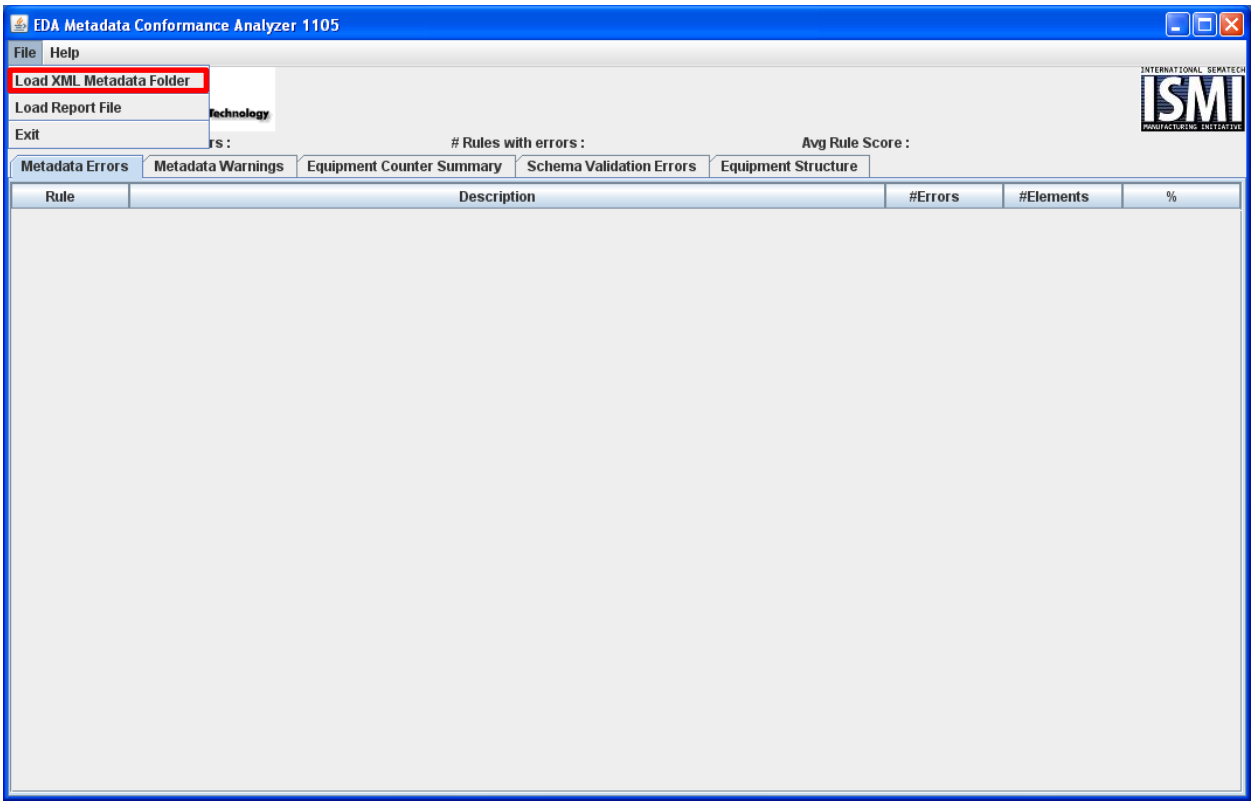
Your metadata can be found in the selected directory after the ECCE becomes responsive (approximately 2 seconds).

5.4 Analyzing Metadata

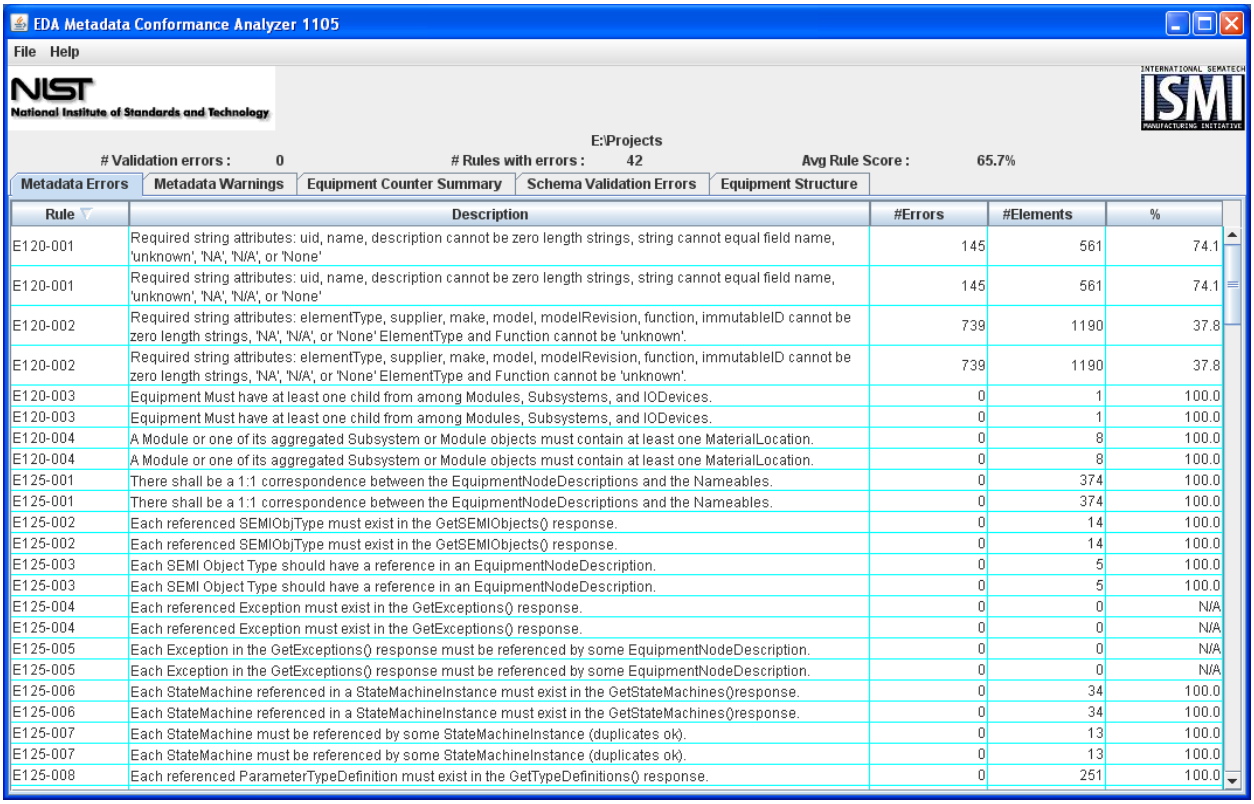
5.4.1 MCA 1105

Launch the MCA via the MCA1105V1.3.bat file.

Click File then Load XML Metadata Folder and select the directory that contains the desired metadata created above.



After the metadata is loaded, they will be displayed on the Metadata Errors tab. This tab displays all rules and the details associated with the rules. Rules that have no errors will display 100.0 under the % column.



Rule	Description	#Errors	#Elements	%
E120-001	Required string attributes: uid, name, description cannot be zero length strings, string cannot equal field name, 'unknown', 'NA', 'N/A', or 'None'	145	561	74.1
E120-001	Required string attributes: uid, name, description cannot be zero length strings, string cannot equal field name, 'unknown', 'NA', 'N/A', or 'None'	145	561	74.1
E120-002	Required string attributes: elementType, supplier, make, model, modelRevision, function, immutableID cannot be zero length strings, 'NA', 'N/A', or 'None' ElementType and Function cannot be 'unknown'.	739	1190	37.8
E120-002	Required string attributes: elementType, supplier, make, model, modelRevision, function, immutableID cannot be zero length strings, 'NA', 'N/A', or 'None' ElementType and Function cannot be 'unknown'.	739	1190	37.8
E120-003	Equipment Must have at least one child from among Modules, Subsystems, and IODevices.	0	1	100.0
E120-003	Equipment Must have at least one child from among Modules, Subsystems, and IODevices.	0	1	100.0
E120-004	A Module or one of its aggregated Subsystem or Module objects must contain at least one MaterialLocation.	0	8	100.0
E120-004	A Module or one of its aggregated Subsystem or Module objects must contain at least one MaterialLocation.	0	8	100.0
E125-001	There shall be a 1:1 correspondence between the EquipmentNodeDescriptions and the Nameables.	0	374	100.0
E125-001	There shall be a 1:1 correspondence between the EquipmentNodeDescriptions and the Nameables.	0	374	100.0
E125-002	Each referenced SEMIObjType must exist in the GetSEMIObjects() response.	0	14	100.0
E125-002	Each referenced SEMIObjType must exist in the GetSEMIObjects() response.	0	14	100.0
E125-003	Each SEMI Object Type should have a reference in an EquipmentNodeDescription.	0	5	100.0
E125-003	Each SEMI Object Type should have a reference in an EquipmentNodeDescription.	0	5	100.0
E125-004	Each referenced Exception must exist in the GetExceptions() response.	0	0	N/A
E125-004	Each referenced Exception must exist in the GetExceptions() response.	0	0	N/A
E125-005	Each Exception in the GetExceptions() response must be referenced by some EquipmentNodeDescription.	0	0	N/A
E125-005	Each Exception in the GetExceptions() response must be referenced by some EquipmentNodeDescription.	0	0	N/A
E125-006	Each StateMachine referenced in a StateMachineInstance must exist in the GetStateMachines()response.	0	34	100.0
E125-006	Each StateMachine referenced in a StateMachineInstance must exist in the GetStateMachines()response.	0	34	100.0
E125-007	Each StateMachine must be referenced by some StateMachineInstance (duplicates ok).	0	13	100.0
E125-007	Each StateMachine must be referenced by some StateMachineInstance (duplicates ok).	0	13	100.0
E125-008	Each referenced ParameterTypeDefinition must exist in the GetTypeDefinitions() response.	0	251	100.0

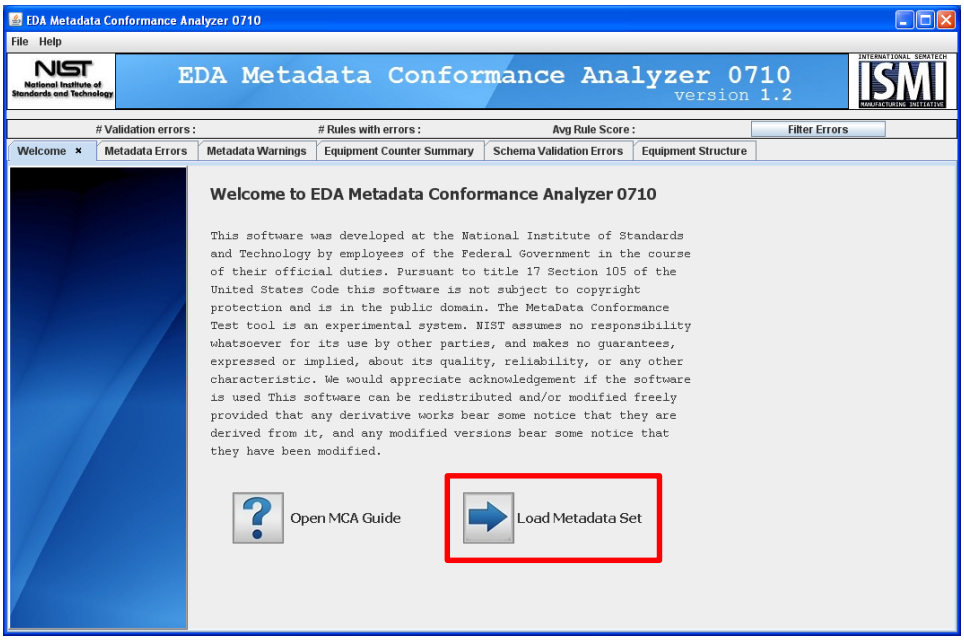
Double click a rule to get detail on the errors that exist for that rule.

Repeat for each rule in the list and correct the errors in the metadata via EMDeveloper if needed.

5.4.2 MCA 0710

Launch the MCA via the MCA0710V1.2.bat file.

Click the "Load Metadata Set" button and select the directory that contains the desired metadata.



Click the Metadata Errors tab and click the "Filter Errors" button. Each item in the list represents a rule in which one or more errors exist.

Double click a rule to get detail on the errors that exist for that rule.

In order to fix MCA reported problems, the user will need to go back to EMDeveloper and change the model so that the metadata will be compliant with the MCA rules. The user will then need to follow the steps listed in section 5.3 (Collecting Metadata) to recreate the metadata files to be analyzed again.

Hint: The process of updating the metadata files is much simpler if you leave EMDeveloper, CIMPortal Plus Control Panel, ECCE and MCA open while fixing the MCA rules compliance issues.

Note: MCA validation checking is not done in EMDeveloper because it would not be possible to keep our software in sync with the 3rd party testing software. Using the method described above will ensure that you have the most up to date E164 analysis available.

5.5 Resolving Errors and Warnings

Errors and Warnings will reference specific Rules from the EDA Guidance document, as well as the E164 requirement (RQ) for the 0710 freeze only. The Guidance document and E164 standard document should be consulted on each item to gain a better understanding what the reported issue is.

5.5.1 Known issues

At the time of release, the model created by the New Model Wizard, without any other changes, has multiple errors reported against it. Cimetrix believes that the model is in fact compliant and the MCA tool is misreporting these issues.

5.5.1.1 GL-005

The message for this error is:

```
Modules must be defined as child nodes of the Equipment, and Load Ports must be
defined as child nodes of a Subsystem with ElementType='EFEM'.
```

This error can occur when the equipment model defines the ElementType of the EFEM node as anything other than "EFEM". E120 suggests that an element with the role of transporting carriers or substrates should assign Element Type as "Carrier Handler" or "Substrate Handler", respectively. To avoid the error in MCA, any model created by New Model Wizard will set the EFEM ElementType to "EFEM".

5.5.1.2 GL-075

The message for this error is:

```
StateMachines mapped from a SEMI Standard state model must match the
corresponding states from the SEMI Standard.
```

There are six errors reported in this section, three for SubstrateLocation and three for BatchLocation. They are all the same class of error.

MCA is reporting these two state machines to be missing their states. But they in fact do exist. Cimetrix suspects the MCA tool is not taking into consideration the top-level state in the state machine when looking for these substates. This misreporting of these states is very similar to the reported problems in section GL-176 below.

5.5.1.3 GL-078

The message for this error is:

```
Exceptions should be a part of the equipment component to which that exception
pertains.

Module has no exceptions.
```

In the case of the reported nodes, there are no exceptions used, and so none defined.

The related standard is SEMI E164-0712, section 9.2.6.1 which simply states exceptions should be referenced by nodes that represent the component to which the exception pertains.

Cimetrix believes this to mean any exceptions should be close in the hierarchy to where they’re used.

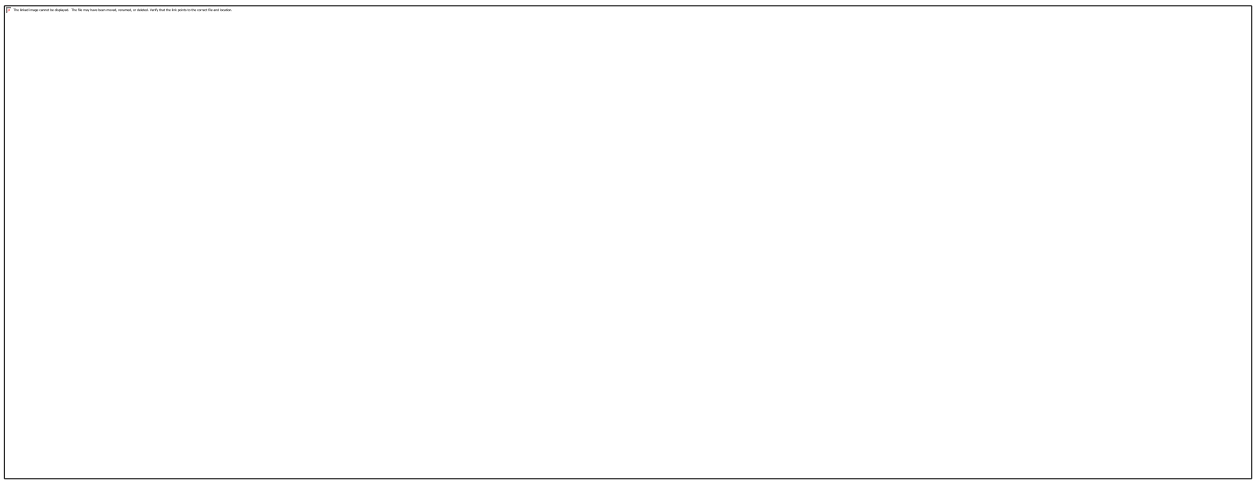
The MCA error seems to indicate all module/equipment nodes must have at least one exception. Cimetrix believes this to be a condition not imposed by the standard, nor can we envision any reasons why it would be. The error can be eliminated by creating exceptions that are not used, but that does not seem beneficial.

5.5.1.4 GL-176

The message for this error is:

```
For a StateMachine taken from a SEMI Standard, each State's id attribute value
shall be the name of that State from that SEMI Standard presented in CamelCase.
```

There are seven failures for this rule, as shown in the image below.



The first failure, “State ID Active does not have the correct corresponding State Name ProcessJob.NoState,” is due to MCA checking to see if the StateName of the ProcessJob.Active state is ProcessJob.NoState. This is incorrect. MCA should be checking to see if the StateName of the ProcessJob.Active state is ProcessJob.Active.

The remaining failures, seen in the image above, involve three state nodes in SubstrateLocation and three state nodes in BatchLocation. They are all the same class of error: “State ID <state> does not have the correct corresponding State Name: <state>”. Apparently for these two state machines, MCA expects the state ID to be the same as the state name. This is wrong. Per SEMI E125-0710, section 10.7.3.2.1, page 26, the state name is the Id of the current state prepended with the path of states to the current state. CIMPortal Plus correctly names all states this way and MCA correctly validates other states this way. These two state machines incorrectly do not validate against this rule.

5.5.1.5 E-125-017

The message for this error is:

```
Each Parameter referenced in an AvailableParameter construct for a
StateMachineInstance or SimpleEvent must have the isTransient parameter set to
Transient.
```

The MCA tool reports this for a number of parameters. The E164 XML files indicate these parameters are to be Unrestricted, not Transient There are no known failures for this rule.

5.5.1.6 GL-127

The message for this error is:

```
Enumeration values must be as defined in the EDA Common Metadata
```

The error detail shows

```
Enumeration EPTStateEnum is missing

Enumeration PreviousEPTStateEnum is missing
```

MCA is known to generate this error when it does not get all the expected enums in an E116GetTypeDefinitions reply. However if E116 is not implemented in the equipment model, the metadata will not have any reply for E116GetTypeDefinitions.

In the case that the equipment does not implement E116, the equipment model need not declare the StateMachines, StateMachineInstances, LogicalElements, Parameters, and ParameterTypes associated with E116. As such the ParameterTypes EPTStateEnum and PreviousEPTStateEnum will not be defined.

If the equipment model defines the ParameterType definitions EPTStateEnum and PreviousEPTStateEnum and these definitions are not referenced by any parameters in the model, then MCA will subsequently generate the error "Each TypeDefinition must be referenced by an equipment Parameter", per rule E125-010. Therefore there is no case in which both rules can be satisfied when the equipment does not implement E116.

Cimetrix believes that excluding E116-related metadata is a legitmate exception to MCA results for equipment that does not support E116.

5.5.1.7 GL-184

The message for this error is:

```
Within a StateMachine based on a SEMI Standard state model, each Event's id
attribute value shall be of the form [state machine]:[transition
number]:[source state]-[target state].
```

MCA is known to generate this error when a the Process Job state "ProcessComplete", a substate of "PostActive" is used as the EventName of the state transition #7 of a Process Job, from ProcessComplete to NoState state..

To avoid the error in MCA, New Model Wizard will name the state transition "PostActive-NoState" rather than "ProcessComplete-NoState".