

Access to Additional Content

for

CEA-860-A, Dated: December 2002

(Click [here](#) to view the publication)

This Page is not part of the original publication

This page has been added by IHS as a convenience to the user in order to provide access to additional content as authorized by the Copyright holder of this document

Click the link(s) below to access the content and use normal procedures for downloading or opening the files.

➤ [CEA-860-A Files](#)

Information contained in the above is the property of the Copyright holder and all Notice of Disclaimer & Limitation of Liability of the Copyright holder apply.

If you have any questions, or need technical assistance please contact [IHS Support](#).



IHS Additional Content Page



Consumer Electronics Association

1919 South Eads Street
Arlington, VA
22202 USA
(866) 858-1555 toll free
(703) 907-7600 main
(703) 907-7601 fax
www.CE.org

May 14, 2008

Dear Customer:

Thank you for purchasing CEA-860-A. Electronic files mentioned in Annex C can be found at www.ce.org/CEA860. If you have any questions, please contact standards@ce.org.

Sincerely,

CEA Technology & Standards

CEA Standard

Device Plug-In Interface to EIA/CEA-709.1 Network Tools

CEA-860-A

December 2002



CEA[®]
Consumer Electronics Association



www.CEA.org

Documents

Document Sharing Hub

NOTICE

Consumer Electronics Association (CEA®) Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of CEA from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than CEA members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by CEA in accordance with the American National Standards Institute (ANSI) patent policy. By such action, CEA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This CEA Standard is considered to have International Standardization implication, but the International Electrotechnical Commission activity has not progressed to the point where a valid comparison between the CEA Standard and the IEC document can be made.

This Standard does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this Standard to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

(Formulated under the cognizance of the CEA's **R7 Home Networks Committee**.)

Published by

©CONSUMER ELECTRONICS ASSOCIATION 2002
Technology & Standards Department
1919 S. Eads Street
Arlington, Virginia 22202

**PRICE: Please call Information Handling Services, USA and Canada (1-800-854-7179)
International (303-397-7956), or
<http://global.ihs.com>
All rights reserved
Printed in U.S.A.**

PLEASE!

DON'T VIOLATE
THE
LAW!

This document is copyrighted by the Consumer Electronics Association (CEA®)
and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies by
entering into a license agreement. For information contact:

Information Handling Services
15 Inverness Way East
Englewood, Colorado 80112-5704
or call U.S.A. and Canada 1-800-854-7179, International (303) 397-7956
See <http://global.ihs.com> or email global@ihs.com

Copyright Consumer Technology Association
Provided by IHS Markit under license with CTA
No reproduction or networking permitted without license from IHS

FOREWORD

This standard was developed under the auspices of the Consumer Electronics Association (CEA, formerly CEMA) Technology & Standards R7.1 HCS1 Subcommittee.

CONTENTS

Device Plug-in Interface to EIA-709.1 Network Tools	1
1 Scope	1
2 Normative References	1
2.1 Normative Reference List.....	1
2.2 Normative Reference Acquisition	1
3 Document Overview	2
3.1 Section 4: Definitions and Symbols.....	2
3.2 Section 5: Overview of Device Plug-in Interface Components	2
3.3 Section 6: Device Plug-in Creation Process	2
3.4 Section 7: Device Plug-in Naming Conventions	2
3.5 Section 8: Device Plug-in Control Interface	2
3.6 Section 9: Device Plug-in Services Interface	2
3.7 Section 10: Installation Specification	3
3.8 Section 11: Error Codes	3
3.9 Support Documentation	3
3.9.1 Annex A: Objectives and Benefits.....	3
3.9.2 Annex B: Usage Scenarios	3
3.9.3 Annex C: Support Files.....	3
4 Definitions and Symbols	3
4.1 Use of Terms	3
4.2 Definitions.....	3
4.3 Symbols and Graphical Representations.....	6
4.4 Abbreviations	6
5 Overview of Device Plug-in Interface Components.....	6
5.1 Architecture Diagram	7
5.2 Component Descriptions	8
5.2.1 Network Tool Computer running Windows® or Windows NT®.....	8
5.2.2 Network Tool Software (Plug-in Client)	8
5.2.3 Vendor-Specific Network Services API.....	8
5.2.4 Vendor-Specific Network Services Interface.....	9
5.2.5 Device Plug-in (COM Server).....	9
5.2.6 Factory Method.....	10
6 Device Plug-in Creation Process	10
6.1 Device Plug-in Types	10
6.1.1 Model-Type Plug-in	10
6.1.2 Device-Type Plug-in	10
6.1.3 Manufacturer-Profile-Type Plug-in	11
6.1.4 Profile-Type Plug-in	11
6.1.5 Manufacturer-Type Plug-in.....	11
6.2 Plug-in Object Creation Steps	11
7 Device Plug-in Naming Conventions.....	14
7.1 Plug-in Application Naming Overview	14
7.2 Plug-in Program Name	14
7.2.1 Prefix.....	15
7.2.2 Program ID Subset	15

7.2.3	User Level Tag	16
7.3	Plug-in Component Name	17
7.4	Support of Multiple Device Plug-ins	18
7.4.1	Plug-in Naming	18
7.4.2	Network Tool Name Usage	18
8	Device Plug-in Control Interface	19
8.1	Quick Reference	19
8.1.1	Interface Definitions	20
8.1.2	I709DevicePlugin Interface	20
8.1.3	I709PluginCommands Interface	20
8.1.4	I709PluginCommand Interface	21
8.1.5	I709DevicePluginDisplay Interface	21
8.1.6	I709NodeConnectionPoint Interface	22
8.1.7	I709DeviceFiles Interface	24
8.1.8	I709PluginFactory Interface	24
8.2	Interface Definition	25
8.2.1	709DevicePlugin Object	25
8.2.2	709PluginCommands Object	27
8.2.3	709PluginCommand Object	28
8.2.4	709DevicePluginDisplay Object	33
8.2.5	709NodeConnectionPoint Object	36
8.2.6	709DeviceFiles Object	39
8.2.7	709PluginFactory Object	40
8.2.8	E709UserLevel Enumeration	40
8.2.9	E709PluginType Enumeration	41
8.2.10	E709DRFType Enumeration	41
8.2.11	E709DRFScope Enumeration	41
8.3	IDL Specification	41
9	Device Plug-in Services Interface	42
9.1	Quick Reference	43
9.1.1	I709Node Interface	44
9.1.2	I709NodeCommStatus Interface	45
9.1.3	I709NetworkVariables Interface	45
9.1.4	I709ConfigurationProperties Interface	46
9.1.5	I709LMObjects Interface	46
9.1.6	I709ConfigurationProperty Interface	46
9.1.7	I709NetworkVariable Interface	47
9.1.8	I709LMObject Interface	47
9.2	Interface Definition	47
9.2.1	709Node Object	48
9.2.2	709NodeCommStatus Object	55
9.2.3	709NetworkVariables Object	57
9.2.4	709ConfigurationProperties Object	58
9.2.5	709LMObjects Object	60
9.2.6	709ConfigurationProperty Object	60
9.2.7	709NetworkVariable Object	63
9.2.8	709LMObject Object	66
9.2.9	E709NodeState Enumeration	69
9.2.10	E709NodeMode Enumeration	69
9.2.11	E709ServiceType Enumeration	69
9.3	IDL Specification	69
10	Installation Specification	70

10.1	Device Plug-in Component Registration.....	70
10.2	Additional Registry Entries	70
10.2.1	Overview.....	70
10.2.2	Device Plug-in Type Keys.....	71
10.2.3	Device Plug-in Keys	71
10.2.4	Device Plug-in String Values.....	72
10.2.5	Using Default Plug-ins	75
10.3	File Placement.....	76
10.3.1	CEA-860-A Type Library File Installation.....	76
10.3.2	CEA-860-A Plug-in Factory Component File Installation	77
10.3.3	Device Plug-ins Directory Structure.....	78
10.4	Device Plug-in Component Removal.....	81
11	Error Codes.....	82
11.1	Runtime Error '429': "ActiveX component can't create object"	82
11.2	Runtime Error '438': "Object does not support this property or method"	82
12	Annexes.....	83
	Bibliography	83
A	Annex A Objectives and Benefits (Informative).....	84
A.1	Overview	84
A.2	Statement of Purpose	84
A.3	Using a Plug-in Architecture.....	84
A.4	Interfaces Described.....	84
A.5	Interoperability and Upgrades	85
A.6	Objectives	85
B	Annex B Usage Scenarios (Informative).....	86
B.1	Overview	86
B.2	Plug-in Control Interface	86
B.2.1	Plug-in User Scenarios	86
B.2.2	Network Tool Developer Scenarios.....	87
B.3	Plug-in Services Interface	87
B.3.1	Network Tool Developer Scenarios.....	87
B.3.2	Plug-in Developer Scenarios.....	87
C	Annex C Support Files (Normative).....	88
C.1	Overview	88
C.2	Provided Software Support Files	89

Figures

Figure 1	Plug-in Architecture Diagram	7
Figure 2	Control Interface Placement	19
Figure 3	Services Client	42
Figure 4	Services Client/Server Relationship	43
Figure 5	Services Device Communication Relationship.....	43
Figure 6	Device Plug-in Registry Keys Sample	72
Figure 7	Registry Key String Values Samples	75
Figure 8	CEA-860-A Type Library File Placement	77
Figure 9	CEA-860-A Plug-in Factory File Placement.....	78
Figure 10	Device Plug-ins Sample Directory Structure	80
Figure 11	Device Plug-ins Sample File Placement	81

Tables

Table 1 Plug-in Object Creation Steps..... 13

Table 2 Plug-in Application COM Program ID Format..... 14

Table 3 Plug-in Program Name Format..... 15

Table 4 Plug-in Prefix Examples..... 15

Table 5 Plug-in Program Name Program ID Subsets 16

Table 6 Plug-in Program Name User Level Tags 17

Table 7 Plug-in Component Names..... 18

Table 8 Mandatory Registry Key Names..... 71

Table 9 Device Plug-in Registry Keys 72

Table 10 Device Plug-in Registry String Value Examples 74

Table 11 Device Plug-in Directory Structure 79

Table 12 Key for Codes Applying to Provided Software Files 88

Copyright Consumer Technology Association
Provided by IHS Markit under license with CTA
No reproduction or networking permitted without license from IHS

Device Plug-in Interface to EIA-709.1 Network Tools

1 Scope

This specification applies to a set of software interfaces between ANSI/EIA-709.1-B-2002 (EIA-709.1) Network Management Tools and device type specific software for device installation, configuration and commissioning, resulting in enhanced functionality of network management tools. This specification also provides backward compatibility to ANSI/EIA-709.1-A-1999 tools. Supported devices include those that meet the EIA-709.1 standard for nodes and the objects they contain which communicate on an EIA-709.1 standard network. Network tool software and device plug-in software properly implementing their appropriate sides of the interfaces described in this specification interoperate on Microsoft® Windows® and Windows NT® platforms using the Component Object Model (COM) technology for interoperability.

2 Normative References

The following standards contain provisions that, through reference in this text, constitute normative provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed in Section 2.1.

2.1 Normative Reference List

- [1] ANSI/EIA-709.1-B Control Network Protocol Specification
- [2] The Component Object Model (COM) Specification
- [3] LonMark® Application Layer Interoperability Guidelines, Version 3.2, Document No. 078-0120-01D, LonMark® Interoperability Association.
- [4] File Transfer, Document No. 005-0025-01D, Echelon Corporation.
- [5] MSDN, MSDN Online, July 2001, Microsoft Corporation

2.2 Normative Reference Acquisition

ANSI/EIA Standards:

- Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Phone 800-854-7179; Fax 303-397-2740; Internet: <http://global.ihs.com>; Email: global@ihs.com

COM Standard:

- The Component Object Model (COM) Specification, *Draft Version 0.9, October 24, 1995, Microsoft Corporation and Digital Equipment Corporation*, Copyright © 1992-95 Microsoft Corporation, One Microsoft Way, Redmond, WA, 98052-6399; Internet: <http://www.microsoft.com/com/resources/comdocs.asp>

IEC Standards:

- Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Phone 800-854-7179; Fax 303-397-2740; Internet: <http://global.ihs.com>; Email: global@ihs.com
- IEC Central Office, 3, rue de Varembe, PO Box 131, CH-1211 Geneva 20, Switzerland; Phone +41 22 919 02 11; Fax +41 22 919 03 00; Internet <http://www.iec.ch>; Email pubinfor@iec.ch

LonMark Interoperability Association:

- LonMark Interoperability Association, 550 Meridian Ave, San Jose, CA 95126-3422; Phone: +1-408-938-5266; Fax: +1-408-790-3838; Internet: <http://www.lonmark.org>; E-mail: info@lonmark.org

Echelon Corporation:

- Echelon Corporation, 550 Meridian Ave, San Jose, CA 95126-3422; Phone: +1-408-938-5200; Fax: +1-408-790-3800; Internet: <http://www.echelon.com>; E-mail: lonworks@echelon.com

MSDN, MSDN Online:

- Microsoft Corporation, One Microsoft Way, Redmond, WA, 98052-6399; Internet: <http://msdn.microsoft.com/default.asp>; E-mail: msdn@microsoft.com

3 Document Overview

This document contains both normative and informative information compiled during the development of a specification for device plug-in interfaces to ANSI/EIA-709.1-B-2002 network tools. The majority of this document consists of the technical definitions of the software interfaces required to allow third party network tools and device plug-ins to interoperate on a user's computer. In addition to the technical definitions, this document also contains background material for the rationale of the specified interfaces including usage scenarios. The interface design architecture and the requirements it was derived from are also included. This background and other informative material can be found in this document's annexes. The following sections describe the major areas of this document and their purpose and contents.

3.1 Section 4: Definitions and Symbols

This section identifies terminology and acronyms that are used throughout the document. Terms are defined either in their general context or in their context specific to this specification.

3.2 Section 5: Overview of Device Plug-in Interface Components

This section describes the software architecture encompassed by the interfaces specified in this document. A system architecture diagram shows the visual relationship of the software and hardware components affected by this specification. Many of the system's components are also described.

3.3 Section 6: Device Plug-in Creation Process

This section describes the software method that a tool will use to create device plug-in objects at runtime. The different types of plug-ins that can be found and invoked are described and the interface specification for this method is defined.

3.4 Section 7: Device Plug-in Naming Conventions

This section describes the areas in which naming conventions must be adhered to when implementing the interfaces in this specification. The naming conventions and their use is defined in this section.

3.5 Section 8: Device Plug-in Control Interface

This section contains the definition of the software interfaces required to allow the network tool to determine the feature set of the plug-in from the device plug-in software component. It begins with a discussion of the role of the interface set and provides a quick reference to the high level details of the interfaces. It follows with a detailed definition of the interfaces' properties and methods including syntax and examples of their implementation. The last area of this section describes the Interface Definition Language (IDL) file provided with this standard for the set of interfaces comprising the Control Interface.

3.6 Section 9: Device Plug-in Services Interface

This section contains the definition of the software interfaces required to allow a device plug-in software component to use the ANSI/EIA-709.1 network services of the network tool. It begins with a discussion of the role of the interface set and provides a quick reference to the high level details of the interfaces. It

follows with a detailed definition of the interfaces' properties and methods including syntax and examples of their implementation. The last area of this section describes the Interface Definition Language (IDL) file provided with this standard for the set of interfaces comprising the Services Interface.

3.7 Section 10: Installation Specification

This section describes the way a device plug-in must be installed to ensure interoperability with the network tool while the tool is running. Naming conventions and software registration specifications are identified.

3.8 Section 11: Error Codes

This section serves as a reference for the developer of device plug-ins and network tools when handling errors during integration at runtime.

3.9 Support Documentation

This document also provides a bibliography and some informative material intended to assist users of this specification with background and supportive documentation. Section 12: Annexes provides a list of the annexes found at the end of the document. Section 13 contains this document's Bibliography. The annex sections follow the bibliography and are summarized in this section.

3.9.1 Annex A: Objectives and Benefits

This section contains a description of the background rationale for developing a specification for interfaces between network tools and device plug-ins. It contains an overview of the objectives and a statement of purpose for the effort. It also comments on software architectures, interfaces, and upgrades. Finally, it lists the specific objectives this specification addresses for tools and device plug-ins in the field.

3.9.2 Annex B: Usage Scenarios

This section provides background information describing the target users of this specification. The target users are described in the context of usage scenarios for this specification. It breaks down the users into their roles as plug-in developers and network tool developers and describes their perspectives and objectives satisfied by implementing to this specification.

3.9.3 Annex C: Support Files

This section provides a table describing the files accompanying this specification. This section is a normative reference and several of the files described and included with this standard are substantive in support of the content of this document. The table names and describes each of the files including their context in support of this specification.

4 Definitions and Symbols

4.1 Use of Terms

The following section introduces the basic terminology employed throughout this document. Most of it is commonly used and the terms have the same meaning in both the general and the EIA-709 Device Plug-in context. For terms that have subtle differences, those differences are noted.

4.2 Definitions

ActiveX Object: Software packages using the OLE Automation and COM technology to expose their features and functions to other software packages known as ActiveX clients. In the context of this standard, the device plug-ins are ActiveX objects and the tools that run them and access their properties

and methods are the objects' clients. ActiveX objects are often called Servers, COM Servers, or ActiveX Servers.

ActiveX Client: Software package that invokes and accesses the features and functions of other software packages, specifically ActiveX Objects, using OLE Automation and COM technology. The client is the package that creates the ActiveX object from a component library. In the context of this standard, network tools are the ActiveX Clients to the device plug-in ActiveX objects.

COM Program ID, COMProgID: The user-friendly string ID registered in the Windows registry with the Globally Unique Identifier (GUID) of an ActiveX object. The ID contains the name of the software program that the object is contained in, the name of the class component from which the object is created, and the version number of the component. The IDs are listed in the registry as "HKEY_CLASSES_ROOT/CLSID" entries and each ActiveX Object's GUID is paired with an associated COM Program ID. COMProgID is the abbreviation used in this standard and its interfaces as an abbreviation for COM Program ID.

Compare: An optional feature that a device plug-in may support that will allow the tool to force the plug-in instance to compare its properties with another provided device or plug-in instance of the same type. This feature causes the plug-in to run a comparison of the contents of one device with another. Either of the device plug-in instances may contain persisted or live device properties. The implementation of the Compare feature would be contained solely in the plug-in and the call would come from the tool or the plug-in user. The plug-in may reflect the results of the compare in its user interface components. This feature would be used, for example, by a user wishing to determine the difference between a persisted and a live device.

Configuration: The non-volatile information used by the devices to customize their operation. There is configuration data for the correct operation of the protocol in each device, and optionally, for application operation. This document refers only to the application configuration information that is application specific.

Control Interface: The set of COM Interfaces shared by the tool and the device plug-in that allow the tool to identify the plug-in by descriptive information, as well as the features the plug-in supports. The implementation of the Control Interface methods and properties exist in the plug-in and the tool will make the calls to the plug-in to expose the details of the plug-in to the user. The control interface allows the user to control the plug-in through the network tool.

Download: The act of sending configuration data to a device that may cause the device to change its operation, settings, or operation limits. Download is a feature supported by the device plug-in Control Interface that allows the tool to force the plug-in to prepare and send its current data down to the device. The device plug-in Services Interface contains the call the plug-in uses to make the actual device download command through the tool. The plug-in contains the data and can initiate its own download through the Services Interface, as well.

Extensibility: The character of a set of COM interfaces which allow them to be broadened in the future to enhance the software's capabilities without causing the previous components to become inoperable. Extensible interface definitions imply that base properties and methods required to run a feature set already exist and that new functions and properties will merely enhance the base behavior.

Factoring: The design goal of identifying and defining a set of software interfaces that implies a focused responsibility for those interfaces. Factoring interfaces involves removing the interfaces to their own component allowing them to become more easily reusable as a collection. Unrelated interfaces are removed from the set so the set of interfaces can retain uniqueness and focused scope.

Factory Method: In general, an interface which has the responsibility of creating an object, but which allows other classes to determine which object to instantiate. In the context of this standard, the factory method of the device plug-in interface is called by the network tool with a COM Program ID of a device plug-in object the tool would like to attempt to create.

Host application: The host application, in the context of this standard is the network tool which instantiates and hosts the device plug-ins.

Interface Definition Language (IDL): The IDL file specifies a network contract between the client (network tool) and server (device plug-in)—that is, the IDL file specifies what is transmitted between the client and the server. Keeping this information distinct from the information about the operating environment makes the IDL file portable to other environments. (MSDN[5])

Network Services: The collection of EIA-709.1 commands and protocol a software network tool uses to communicate with, service, and configure an EIA-709.1 compliant physical network and its compliant nodes. This standard requires that the network tool expose a subset of these commands to the device plug-in to allow it to service EIA-709.1 nodes. The collection of interfaces between the tool and the device plug-in allowing for these network services is called the Services Interface.

Persistence: The ability of a software object to retain its value or state after the object itself is terminated by the host application. In the context of this standard, persistence is applied to the ability of a device plug-in object to retain its property values, data, and settings to be recreated at a later time. Software persistence requires that a number of special properties and methods of an object be implemented prior to being shut down. The flag in the Control Interface definition of this standard for Supports Persistence is set to True when the necessary methods and properties have been implemented in the device plug-in object to allow it to be persisted and recreated later from the persisted object.

Plug-in: In the context of this standard, a software package that can be created by and within another software package that adheres to a set of interfaces allowing the two packages to expect exact behaviors from the other. Network tools create software objects based on a standard creation method. When created, those software objects expose a standard known set of properties and methods that plug into the calls that the tool has implemented to access the software objects. The connection is two-way once it is made, and the plug-in object can also make calls on a standard known set of properties and methods implemented in the tool software.

Program ID: In the context of this standard and the EIA-709.1[1] standard, this refers to the program-identifying property of an EIA-709.1-compliant node. The program of a node is the application in that node model which is identified as unique and which causes its unique behavior. Each node model contains a unique program ID which identifies whether the node is based on the LonMark Interoperability Guidelines[3], the manufacturer of the node, the LonMark device class and subclass of the node, the Industry Use Classification of the node, and the node's model.

Services Interface: The set of COM Interfaces shared by the tool and the device plug-in that allow the tool to provide network services for a device to its plug-in. The plug-in makes calls on a node object provided by the tool which then passes the calls on to its network services layer or API for communication with the device. The network services provided by the tool via the Services Interface include access to the device's configuration properties and network variables, communication status of the device, and explicit messaging for the device. The implementation of the Services Interface methods and properties exist in the tool and the plug-in will make calls on the Node object provided by the tool to expose the details of the node to the user. The Services Interface allows the user to configure the node through the network tool via the plug-in.

Status: The interpretation of an EIA-709.1-compliant node's inputs and outputs that allow a user to determine the node's state of operation. In the context of a device plug-in, status is generally referenced as the collection of input/output network variables of the device. These are to be distinguished from the properties that allow the use to change the device's application configuration.

Upload: The act of calling for configuration data from a device for use in the display of the data for potential alteration. Upload, as the counter-feature to download, is supported by device plug-ins to allow the tool to force the plug-in to reload its data from the device, rather than continue to use its own

properties. Once the call to upload has been made, the plug-in contains the data and can prompt its own uploads and downloads, as well.

Z Order: The *Z order* of a window indicates the window's position in a stack of overlapping windows. This window stack is oriented along an imaginary axis, the z-axis, and extending outward from the screen. The window at the top of the Z order overlaps all other windows. The window at the bottom of the Z order is overlapped by all other windows. (MSDN[5])

4.3 Symbols and Graphical Representations

N/A

4.4 Abbreviations

The software industry and EIA-709 related abbreviations that appear in this document are:

API	Application Programming Interface
C++	A software development language
COM	Component Object Model
COMProgID	COM Program ID
CPT	Configuration Property Type
GUID	Globally Unique Identifier
IDL	Interface Definition Language
OEM	Original Equipment Manufacturer
SCPT	Standard Configuration Property Type
UCPT	User-defined Configuration Property Type
VB	Visual Basic – a software development language

5 Overview of Device Plug-in Interface Components

A solid architecture for a plug-in model is critical to its success in satisfying the goal of interoperability. It must anticipate the basic needs of network tool users when viewing and configuring a variety of device types. It must anticipate the need for plug-in modules to be upgraded more frequently than the tools that use them. It must allow device plug-ins to be implemented as in-process or out-of-process servers. Finally, the architecture must anticipate that vendors may build proprietary device plug-in/network tool solutions that implement an extended interface that incorporates, as part of the extended interface, the complete interoperable device plug-in interface defined by this standard.

5.1 Architecture Diagram

The overall structure of the interaction between the network tool and the plug-in can be depicted as shown in the figure Plug-in Architecture Diagram.

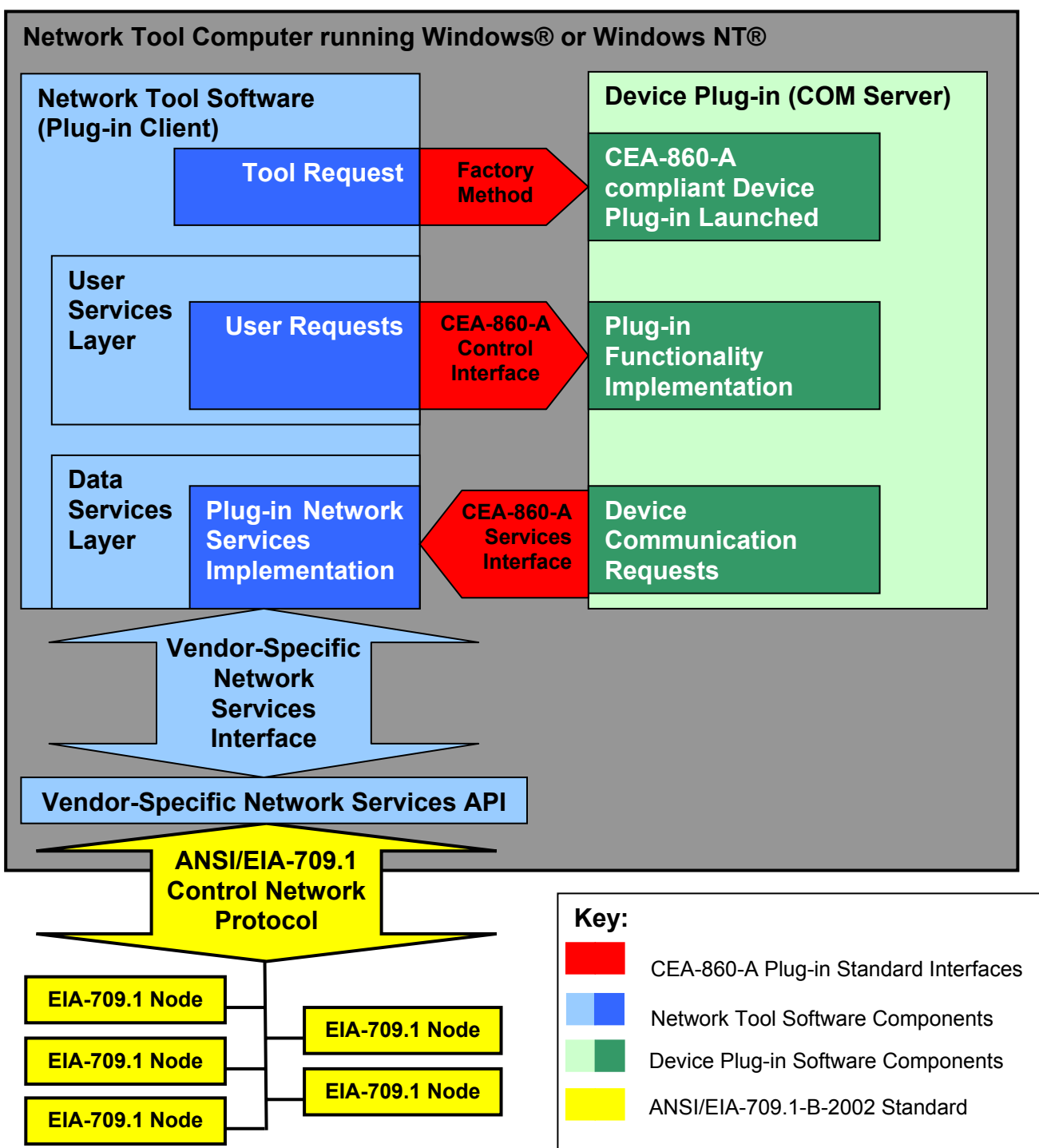


Figure 1 Plug-in Architecture Diagram

5.2 Component Descriptions

5.2.1 Network Tool Computer running Windows® or Windows NT®

This is the computer on which the Network Management Service Tool, the Device Plug-in(s), and the Network Management API are installed. This standard defines interfaces that are designed to run on the Microsoft® Windows® 95, Windows® 98, Windows® Me®, Windows® 2000, Windows® XP®, and Windows NT® 4.0 operating system platforms. The Network Management Service Tool and the Network Management API are integrated based on a proprietary interface mechanism provided by the tool vendor.

5.2.2 Network Tool Software (Plug-in Client)

This is the complete application that services EIA-709.1 networks. It is comprised of two major components, Tools and Services, and is jointly responsible for handling all user requests while communicating on an EIA-709.1 Network. The network tool is a COM client based on the Component Object Model (COM) specification revision 0.9. It may be implemented using any threading model appropriate to the application. In addition, it must adhere to the COM interface calling rules, as it is known that multi-threaded clients that do not properly marshal their threads can cause a non-thread-safe server to malfunction.

5.2.2.1 User Services Layer

This portion of the Network Tool provides the user interfaces and directs all functionality for the user communicating with an EIA-709.1 Network. It interacts with the Services component as needed to handle direct communications with the network. It also coordinates with the Device Plug-in component as needed to send requests for plug-in functionality. It uses the CEA-860-A Control Interface to access the device plug-in's functions and data. It also receives information back from the plug-in through this interface.

5.2.2.2 Data Services Layer

This portion of the Network Tool initiates all the network services requests and processes all returned information from a communicating EIA-709.1 network by interfacing with the Network Management API. It sends all requests through the Network Services Interface, as well as accesses all desired functionality from the API. It also interacts with the Device Plug-in component upon request to provide network services to the device plug-in. It uses the CEA-860-A Services Interface to receive communication requests from the plug-in and implement communication functionality using the Network Services Interface. It also returns information to the device plug-in through this interface.

5.2.2.2.1 CEA-860-A Services Interface

This interface consists of the properties and methods that expose a network tool's network services functionality available to a device plug-in. Requests made by the device plug-in through this interface are in the form of calls for properties and methods. Functionality required to service these requests are implemented in the host network tool using the arguments passed by the device plug-in. All passed and returned value types are defined in the interface allowing for complete compatibility between the components. The device plug-in component determines what functionality is available from the network tool, whereas the network tool provides the actual functional implementation.

5.2.3 Vendor-Specific Network Services API

This component contains the set of programming instructions required to communicate with EIA-709.1 networks. It uses the EIA-709.1 protocol to send and receive messages from nodes on a network, and provides mechanisms for other applications to view those messages. This component is part of the proprietary network services solution of the network tool and is not further defined by the Plug-in Standard.

5.2.4 Vendor-Specific Network Services Interface

This interface consists of the properties and methods that allow for integration of the network tool application with the Network Management API. This component is developed by either the vendor of the API, the tool vendor, or both. The result is a proprietary solution that has no effect on the device plug-in components of the network service tool. All communication requests received by the Services component of the service tool are handled by this component to make them compatible with the interface to the Network Management API. The device plug-in components need not know what underlying mechanism is used to send and receive the messages. This separation allows each device plug-in to work with any conforming EIA-709.1 network management tool. This component is part of the proprietary network services solution of the network tool and is not further defined by the Plug-in Standard.

5.2.5 Device Plug-in (COM Server)

This component of the network tool represents all device plug-ins as developed against the standard. There may be one or more plug-ins that will be used by any network tool installation. Each installed plug-in with a set of Plug-in interfaces conforming to this standard will be compatible with any Network Tool supporting device plug-ins implemented to this standard. The plug-in is a COM server based on the Component Object Model (COM) specification revision 0.9. It may be implemented using any threading model appropriate to the application. Nonetheless, it is recommended that the plug-in COM server be implemented to be thread-safe so as to be compatible even with COM clients that do not strictly adhere to COM's threading rules.

Each standard device plug-in has the ability to interface with the client Network Tool by handling requests for feature implementation. It will use the CEA-860-A Control Interface to receive requests, provide the necessary implementation required to satisfy those requests, and return any values required by the Network Tool application. Examples of feature implementation may include the ability to display the device's status data screen, display and modify the device's configuration data, save the device's data, or print device's data.

Each standard device plug-in also has the ability to interface with the host Network Tool by requesting network services functionality from the tool. It uses the CEA-860-A Services Interface to send requests and necessary data with the requests to the network tool. When the network tool has implemented communications functionality as requested by the device plug-in, it uses the interface to return any expected data to the device plug-in. Examples of communication requests may include instructions to get the device's unique node ID, return NV values, reset a node, return the status of a node, or set configuration values.

The Control and Services interfaces are tied together within the plug-in software implementation. When the network tool user makes a request for functionality for a device, the tool uses the CEA-860-A Control Interface to communicate the request to the plug-in. The device plug-in implementation of the feature usually requires communication with the device. The plug-in incorporates sending communication requests back to the network tool via the CEA-860-A Services Interface. The tool will complete the device communication request using its own network services and return any data and control to the device plug-in. The plug-in uses the returned data to complete the request initially made by the user of the network tool. The implemented interface components work together to share information and complete tasks. The control passes from one component to the other sequentially through the interfaces. The implementation of functionality thus is handled by the appropriate component without the other component needing any additional data or code compatibility than what the interfaces define.

5.2.5.1 CEA-860-A Control Interface

This interface consists of the properties and methods that expose a device plug-in's available data and functionality. Requests made by the network tool through this interface are in the form of calls for properties and methods. Functionality required to service these requests is implemented in the device plug-in using the arguments passed by the tool. All passed and returned value types are defined in the interface allowing for complete compatibility between the components. The network tool component

determines what functionality is provided by the device plug-in, whereas the plug-in contains the actual functional implementation.

5.2.6 Factory Method

This method is run by the network tool communicating with a device supported by a device plug-in. The network tool passes the program ID of the device to this method which attempts to locate and launch the device-supporting plug-in registered in the user's Windows Registry. This method provides levels of device plug-in support by parsing the program ID of the device if an exact match to the device model is not found. Once a device plug-in is launched using the Factory method, its function set is fully available to the network tool for loading, execution and display within the tool.

6 Device Plug-in Creation Process

Network tools must have a way to create the plug-in software objects before running the feature set supported by them. Tools associate a device on its network with a registered software component representing the device plug-in. The connection between the two is based on the Program ID of the device and the name of the plug-in program. There are five types of plug-ins a tool can invoke for a single device's program ID as described below.

The creation process is a multi-step process where the device's program ID is used to create the Plug-in object. This process occurs by the network tool providing the device's program ID to a 'Factory' method. The Factory returns the plug-in object associated with the device. The creation process executes up to seven major steps in order and the process exits when a device plug-in object is created. Subsets of steps may be undertaken to use a plug-in supporting different user levels. The tool may also implement caching of device plug-in objects to minimize creation time. The steps for plug-in object creation are detailed in the next section.

The creation process allows device manufacturers to develop a smaller number of plug-ins supporting multiple devices. If a device has a number of proprietary extensions, the manufacturer may decide to make a plug-in for a specific device version or make a more generic device plug-in that services all revisions of a given device. The process also allows a profile to include proprietary extensions. A device may be serviced by a standard device profile plug-in according to the applicable rules, or that plug-in may be 'overridden' by installing a specific device plug-in. This allows the service technician to service that device using the customized plug-in, while the profile would remain active for devices containing a different program ID.

6.1 Device Plug-in Types

This section lists plug-ins in their order-of-specificity of device support that is also the order of the steps the creation process will use to create the plug-in objects. The model-type plug-in offers the most specific support for a device and the manufacturer-type plug-in offers the least specific support.

6.1.1 Model-Type Plug-in

This device plug-in type offers the most specific support for the device. It is based on the model and the revision of the device and allows the user to access the device using the most specific knowledge of its data and application. It can be expected to contain all data needed to configure the device, as well as the latest defect fixes available in the model revision. This plug-in type would have its program name based on the entire program ID of the device – one that is model-specific.

6.1.2 Device-Type Plug-in

This device plug-in type offers device-level support for a family of devices. All devices that have the same starting fourteen (14) characters in their program ID can be supported by a single plug-in. Data common to all devices of the family can all be serviced by this plug-in. The plug-in can also have logic that can further interpret a specific device model and service it more specifically. The plug-in servicing this family

of devices would be launched for any device containing the same first fourteen characters of the program ID. A device-type plug-in, not specific to model, would have its program name based on the device family's program ID without the model identified. This standard only supports this plug-in level for LonMark® Guidelines-format EIA-709.1 devices.

6.1.3 Manufacturer-Profile-Type Plug-in

This plug-in type can service all devices made by a specific manufacturer that fit a certain LonMark device profile. The manufacturer of a device is identified by the second through sixth (2-6) characters of the device's program ID. The profile of a device is identified by the seventh through tenth (7-10) characters of a device's program ID. Every device fitting a profile can be expected to have a subset of its data described commonly to all devices in the same profile. This data can be implemented in an even more specific way by a single manufacturer. This level plug-in can service all profile devices of a single manufacturer specific to that manufacturer's common implementation of the profile. A manufacturer profile plug-in would have its program name based on the subset of the device's program ID representing both the manufacturer ID and the profile of the device. This standard only supports this plug-in level for LonMark® Guidelines-format EIA-709.1 devices.

6.1.4 Profile-Type Plug-in

This plug-in type can service all devices made by any manufacturer that fit a certain LonMark device profile as described in the previous section. This is a more general level of plug-in support because this plug-in would not be able to interpret a single manufacturer's implementation for a profile. This plug-in would be expected to service all mandatory and optional data in the profile device in a way that all profile devices could be configured and serviced in a common way. A profile-based plug-in would have its program name based on the plug-in manufacturer's Manufacturer ID and on the subset of the device's program ID representing profile of the device. This standard only supports this plug-in level for LonMark® Guidelines-format EIA-709.1 devices.

6.1.5 Manufacturer-Type Plug-in

This plug-in type can service all devices of any model, family, or profile manufactured by a single company. This is the most general type of device support covered by this specification. The plug-in can be expected to provide only the level of support common to all devices distributed by this manufacturer, or the plug-in itself will be able to determine the differences between the devices it will be expected to support. This plug-in type will be launched last in the creation order and if this plug-in type does not exist for a device, the tool will have failed to find a supporting plug-in for the device. A manufacturer plug-in would have its program name based on the subset of the device's program ID representing just the device's manufacturer ID. This standard only supports this plug-in level for LonMark® Guidelines-format EIA-709.1 devices.

6.2 Plug-in Object Creation Steps

Windows® COM Components are registered by COM Program IDs which follow the format <Program>.<Component>.<Version>. The tool will identify a single desired object to the factory method by building a string based on the program ID of the device. The Factory method will attempt to create the object by locating its registered program. The Factory will return a NULL if the object cannot be created. The tool will call the Factory once with a Program ID of a device and parameters indicating the plug-in type and user level that is desired. The factory method will return the first object it was able to create based on the provided parameters. If the calling tool has provided type and user level parameters, the factory will only return a corresponding plug-in if found.

In most cases, the caller will indicate a desire for any type or user level plug-in. The factory will attempt to return any found plug-in, moving in order of most-specific device support (Model level, Default user) to least specific plug-in support (Manufacturer level, Generic 5 user.) Once the factory locates a plug-in and returns an object, it stops its attempt to create plug-ins at any higher levels. If the Factory cannot return an object for the highest level, least specific plug-in type, it will return NULL and the tool will recognize the

device as being unsupported by a plug-in on the system. The naming conventions illustrated below are described in more detail in the following section Device Plug-in Naming Conventions.

The steps used to create the device plug-in object are spelled out in Table 1 below. A COM Server software component (DLL) file for this method has been provided as part of this specification document because the factory method described will be implemented nearly identically in all tools. Tool implementers may use the provided library or create their own component for the factory method. The source code used to create this component library has also been provided as a guideline for understanding and development. Follow these links to the [Factory Method DLL](#) and its [source code](#).

Step #	Action	COM Program Id Format/Example
1	Create a Device/Model plug-in object using the entire program ID of the device.	CEA860AMod _ <Full Program Id>_Def.Mod
		Program ID = "80002A5500030411" COMProgID = "CEA860AMod_80002A5500030411_Def.Mod"
2	Create a Device Plug-in object using the manufacturer and profile subsets of the program ID of the device and a component name reflecting a device-type plug-in.	CEA860ADev _<Device subset>_Def.Dev
		Program ID = "80002A5500030411" COMProgID = "CEA860ADev_0002A5500_Def.Dev"
3	Create a Manufacturer Profile Plug-in object using the manufacturer and profile subsets of program ID of the device and a component name reflecting a Manufacturer-Profile-type plug-in.	CEA860AMPf _<MFR+Profile subset>_Def.MPf
		Program ID = "80002A5500030411" COMProgID = "CEA860AMPf_0002A5500_Def.MPf"
4	Create a Profile Plug-in object using the plug-in manufacturer ID, the device profile subset of program ID of the device, and a component name reflecting a Profile-type plug-in.	CEA860APrf _<Profile subset>_Def.Prf
		Program ID = "80002A5500030411" COMProgID = "CEA860APrf_00001_5500_Def.Prf"
5	Create a Manufacturer Plug-in object using the manufacturer subset of program ID of the device.	CEA860AMfr _<MFR subset>_Def.Mfr
		Program ID = "80002A5500030411" COMProgID = "CEA860AMfr_0002A_Def.Mfr"
6	[Reserved for future]	
7	[Reserved for future]	

Table 1 Plug-in Object Creation Steps

The tool may perform additional steps between the major steps that will attempt to create plug-in objects that support a specific user level as described completely in the Plug-in Naming Conventions section below. Additionally, the tool can implement a solution for launching and caching multiple plug-ins of a certain type or user level and allowing the user to select the appropriate one based on the descriptive information the plug-in publishes about itself. The plug-in developer must be aware that a tool may launch multiple instances of a plug-in simultaneously. It will be the responsibility of the plug-in to manage any external data and dependencies in situations where more than one instance is being run and utilizing the same resources.

Devices that do not follow the program ID definitions described in the LonMark Interoperability Guidelines[3] are supported by this specification at the device model-specific level only. Parsing the

program IDs of these devices in order to find more generic plug-in support could result in unanticipated conflict with actual plug-ins which are unrelated to the device or the manufacturer. In order to determine that a device's program ID follows the format expected, a tool must identify whether the first character of the program ID is an '8' or a '9'. If any other character is found in this position, the tool must not attempt to parse the program ID as indicated in Steps 2 through 7 above. Only Plug-in Creation Step 1 may be applied to identify plug-in support for this device and if a plug-in is not created the tool must assume the device is not supported by any plug-ins on the user's system.

This specification allows for multiple Profile Plug-ins to be installed on a user's computer, however the factory method defined, and the sample plug-in factory provided, will always return the first profile plug-in found for a device. This is due to there being no built-in method for identifying which manufacturer's profile should be launched. A tool may provide more robust functionality for identifying which manufacturer's plug-in to launch by identifying all the Profile plug-ins installed for a device and allowing the user to select the one to launch. The tool identifies which plug-ins are installed by iterating the registry entries for those plug-ins that followed the recommend plug-in installation steps described in Section 10. In the absence of this functionality in the tool, the user will control which plug-in displays for a device in the way the plug-ins are installed. The user wanting only one selected plug-in to run each time the device is found will install only one Profile Plug-in at a time. Failing this, the user should expect that, when multiple Profile Plug-ins are installed, the Factory Method will always return the first found on the system. The order will be random and the launched plug-in may not be the same each time the tool is run.

7 Device Plug-in Naming Conventions

In order to achieve interoperability with no prior interaction between third party vendors of network service tools and plug-ins, portions of the device plug-in and the associated installation package must adhere to naming conventions for software components and their registration with the Windows operating system. This section defines the mandatory naming conventions for the plug-in applications and their components. Please refer to the section Installation Specification for the recommended file system locations for the plug-in applications.

7.1 Plug-in Application Naming Overview

As outlined in the previous section Plug-in Object Creation Steps, Network tools will contain implementation for creating plug-ins at runtime based on the device's program ID and the type of plug-in the tool would like to run. A tool may attempt to invoke all possible plug-ins for a device by following the steps outlined in the previous section Plug-in Object Creation Steps. It may also implement a subset of these rules as appropriate for the application. Because a tool's only means of creating plug-in objects may be based on the device's program ID at runtime, the tool must have a pre-identified set of prefixes and suffixes that installed plug-ins have incorporated into their names. As described above, Windows COM Components are registered by COM Program IDs that follow the format <Program>.<Component>.<Version>. The next two sections define the mandatory name extensions for the Program name of the plug-in and the Component name of the plug-in. The following table shows the format of the full COM Program ID of the first version of a registered plug-in supporting a device at the model level for a default user.

Format	Example
ProgramName.ComponentName.Version	CEA860AMod_80002A5500030411_Def.Mod.1

Table 2 Plug-in Application COM Program ID Format

7.2 Plug-in Program Name

The Program Name contains three sections. The Prefix is mandatory and variable for all plug-ins and identifies this standard and the plug-in type described by this standard. The Program ID Subset is mandatory and variable and is used to identify the plug-in type by program ID. The User Level Tag is

mandatory and variable and is used to identify the level of user expected to use the plug-in. The sections of the program name are delimited with an underscore character ('_') to provide visual separation around the program ID portion subset of the name. The following table shows the format of a Plug-in Program Name.

Format	Example
PrefixType_ProgramIDSubset_UserLevel	CEA860AMod_80002A5500030411_Def

Table 3 Plug-in Program Name Format

7.2.1 Prefix

Each device plug-in must identify this specification by number as the first section of the program name. This allows tools to search the registry for installed plug-ins and ensures that there are no conflicts with other installed applications. The first seven characters of a compliant plug-in program name must be "CEA860A". These seven characters are immediately followed by three pre-defined characters identifying the type of plug-in it is as described in the section Device Plug-in Types. These characters are identical to those used as the component name in the same plug-in and are described in the section Plug-in Component Name. They are listed here as a reference.

Plug-in Type	Prefix Characters	Complete Prefix Example
Model Level support: Most specific level of support of the device and model. Paired with the Device Model Program ID Subset.	Mod	CEA860AMod
Device Family support: Supports a family of devices without regard to the specific device model. Paired with the Device Family Program ID Subset.	Dev	CEA860ADev
Manufacturer Profile support: Supports device profiles of devices of a specific device manufacturer. Paired with the Manufacturer Profile Program ID Subset.	MPf	CEA860AMPf
Device Profile support: Supports device profiles without regard to the manufacturer of the device while showing the manufacturer of the plug-in. Paired with the Device Profile Program ID Subset.	Prf	CEA860APrf
Manufacturer support: Supports a manufacturer's device without regard to the profile or device family.	Mfr	CEA860AMfr

Table 4 Plug-in Prefix Examples

7.2.2 Program ID Subset

Each device plug-in must provide the subset of a device's program ID for the type of plug-in it will support. For a model-specific plug-in, this means that the entire program ID of the device must be included in the name of the plug-in program. At the least specific level of support, the manufacturer ID portion of the program ID must be included in the name of the plug-in program. The program ID subset must be delimited on each side in the program name by underscore characters. The following table shows the different Program ID Subsets that would be used for the different levels of support for the device with program ID "80002A5500030411".

In the case of the Profile plug-in type, a naming mechanism is needed to ensure that different manufacturers plug-in program names remain unique when installed on the user's system. An additional name subset must be added to the profile subset of the device's program ID and the name delimited by an additional underscore ('_') character to identify it as a profile-only plug-in. To ensure uniqueness, the additional section of the program name will be the LonMark manufacturer ID of the *plug-in manufacturer*. If a plug-in manufacturer that is not a LonMark Member creates and distributes a plug-in for a LonMark profile, that manufacturer may use an alphanumeric string in place of the manufacturer ID. The manufacturer must attempt to ensure global uniqueness when choosing the string. In no case may the manufacturer begin the string with any character or value that would cause the first nibble (MSB of the first byte, as ordered from left to right) to be 1000 or 1001. For profile plug-ins, there is no direct way to find the plug-in. To do this, the tool will need to iterate through the entire list of CEA860APrf_XXX entries to find desired entries.

Each of the examples in the tables below imply that manufacturer '00001' wrote each of the *profile-type* plug-ins listed to distinguish between the profile plug-in manufacturer and the third-party devices it supports.

Plug-in Type	Program ID Subset	Example (Without User Level)
Model	80002A5500030411	CEA860AMod_80002A5500030411_
Device	0002A5500	CEA860ADev_0002A5500_
Manufacturer-Profile	0002A5500	CEA860AMPf_0002A5500_
Profile	5500	CEA860APrf_00001_5500_
Manufacturer	0002A	CEA860AMfr_0002A_

Table 5 Plug-in Program Name Program ID Subsets

7.2.3 User Level Tag

Each device plug-in must include an indication of the level of user support it provides. Manufacturers providing only one plug-in of any one type are advised to use the default User Level tag indicating that it contains the only level of user support offered for that plug-in type. This is not required and plug-ins may be named indicating a different level of user support. Tools, however, at a minimum, should attempt to support default plug-ins at each level. More robust tools may implement mechanisms to support more advanced user support or support for multiple plug-ins. Multiple plug-ins may be distributed for a plug-in level and each must have different, predefined contents in the User Level section of the Plug-in Program Name. Multiple plug-in support is discussed in detail in the section Naming Multiple Plug-ins below. The following table shows the different User Level Tags and their advised usage context.

User Support Level	User Level Tag	Examples
Default User: Generic user support intended for all users and not specifying a higher or more sensitive level of application support of the targeted device or profile.	Def	CEA860AMod_80002A5500030411_Def CEA860ADev_0002A5500_Def CEA860AMPf_0002A5500_Def CEA860APrf_00001_5500_Def CEA860AMfr_0002A_Def
Technician/Expert User: Manufacturer-specific technician support for advanced or sensitive applications of the targeted device or profile. May be password protected.	Tech	CEA860AMod_80002A5500030411_Tech CEA860ADev_0002A5500_Tech CEA860AMPf_0002A5500_Tech CEA860APrf_00001_5500_Tech CEA860AMfr_0002A_Tech
Manufacturer Engineer: Highest level of support for engineering applications, testing, or development use of the plug-in for the targeted device or profile.	OEM	CEA860AMod_80002A5500030411_OEM CEA860ADev_0002A5500_OEM CEA860AMPf_0002A5500_OEM CEA860APrf_00001_5500_OEM CEA860AMfr_0002A_OEM
Generic Support 1-5: No user level support indicated; recommended for use when multiple plug-ins are distributed for a plug-in type. Maximum tag index number supported by this specification is 5. See “Naming Multiple Plug-ins” section for more information.	Gen1 Gen2 Gen3 Gen4 Gen5	CEA860AMod_80002A5500030411_Gen1 CEA860AMod_80002A5500030411_Gen2 CEA860AMod_80002A5500030411_Gen3 CEA860AMod_80002A5500030411_Gen4 CEA860AMod_80002A5500030411_Gen5

Table 6 Plug-in Program Name User Level Tags

7.3 Plug-in Component Name

The plug-in component name is the name of the component implementing the device plug-in interfaces and will be the object created when the service tool uses the plug-in creation method. The component name is mandatory and variable and indicates the plug-in's type – the level of device support contained in the plug-in. Each pre-defined name corresponds to a plug-in creation step described in the section Plug-in Object Creation Steps above. It is also identical to the plug-in type indicated in the prefix for the plug-in program name. The tool will match the program ID subset it is attempting to launch with its corresponding component name. The following table shows the different Plug-in Component Names and their mandatory usage context. Refer to the previous section Program ID Subset for the list of the subset these plug-in types are paired with.

Plug-in Type	Component Name	Example
Model Level support: Most specific level of support of the device and model. Paired with the Device Model Program ID Subset.	Mod	CEA860AMod_80002A5500030411_Def.Mod
Device Family support: Supports a family of devices without regard to the specific device model. Paired with the Device Family Program ID Subset.	Dev	CEA860ADev_0002A5500_Def.Dev
Manufacturer Profile support: Supports device profiles of devices of a specific device manufacturer. Paired with the Manufacturer Profile Program ID Subset.	MPf	CEA860AMPf_0002A5500_Def.MPf
Device Profile support: Supports device profiles without regard to the manufacturer of the device while showing the manufacturer of the plug-in. Paired with the Device Profile Program ID Subset.	Prf	CEA860APrf_00001_5500_Def.Prf
Manufacturer support: Supports a manufacturer's device without regard to the profile or device family.	Mfr	CEA860AMfr_0002A_Def.Mfr

Table 7 Plug-in Component Names

7.4 Support of Multiple Device Plug-ins

This specification supports the installation of multiple plug-ins for different plug-in types and user levels on the same service tool machine. Strict adherence to the naming conventions described above will allow multiple plug-ins to be installed and registered without conflict. Manufacturers are responsible for not naming two plug-ins with the same name if they are intended for uncontrolled distribution. Tools are responsible for forming the COM Program ID strings completely and correctly in order to launch the plug-in appropriate for the user's context.

7.4.1 Plug-in Naming

To successfully distribute multiple plug-ins for a plug-in type, create two or more plug-ins with program names that differ only in the User Level Tag. At the model level, these plug-ins would provide a different level of support for identical device types. If the plug-in manufacturer would like to identify that one should be used by a building owner, for example, the plug-in component name would include tags Def or Gen1. To identify that a service technician should use a set of plug-ins addressing the technician level of expertise, the plug-in component would include the name Tech.

7.4.2 Network Tool Name Usage

To successfully launch from a number of applicable plug-ins, tools would identify the order priority for their specific application and build strings and attempt to launch plug-ins in varying order based on different user-level tags. The tool developer would implement intermediate steps in the creation process outlined in the previous section that was based on the user-level tag of the plug-in, as well as the plug-in type. Tools can serve their users by adjusting the order of access for different plug-in user levels. If a tool will be used by technicians only, the tool could set an option so that all Tech plug-ins would be launched before default or general plug-ins. An in-house testing tool could be set to launch the OEM-tagged plug-ins exclusively. Passive tools in the field could be set up to only launch the default plug-ins for the model-only of a device.

8 Device Plug-in Control Interface

This specification allows for custom software applications to be created as device plug-ins to service tool applications that service EIA-709.1 control networks. Device manufacturers create custom, flexible device applications that are used within third party network tools selected by service technicians to suit their needs. Any compliant application will use this specification to invoke a compliant device plug-in and run its feature set. The described architecture allows for interoperability by ensuring that a standard set of software interfaces will be adhered to in both the tools and the device plug-ins, as well as by ensuring that extensions to the interfaces will continue to allow the base set of interfaces to operate as expected.

This specification provides the network tool the ability to interact with the device plug-in software as needed to provide a flexible environment to configure and monitor individual devices on a network. The next section, Device Plug-in Services Interface, describes the interfaces allowing the tool to accomplish the device configuration and browsing tasks required by device plug-ins. The feature set exposing these services to the user sits on top of the Services Interface layer and provides the user of the tool access to those features. This set of interfaces is the Control Interface of the device plug-in.

The responsibility of the Control Interface, defined in this section, is the feature interaction between the tool and the plug-in as directed by the user. The interfaces defined here are accessed by the tool to determine the tasks that can be exposed to the user. They identify how the user will identify the task within the tool, as well as the commands the tool must direct back to the plug-in to run those tasks. In running each feature, the plug-in will make calls to the Services Interface as needed for interaction with the device. The features themselves can provide a user interface component and expose device data deemed appropriate by the device manufacturer for configuring or monitoring the device. The Control Interface controls what the user can do with the device and utilizes the Services Interface to accomplish the underlying device services needed.

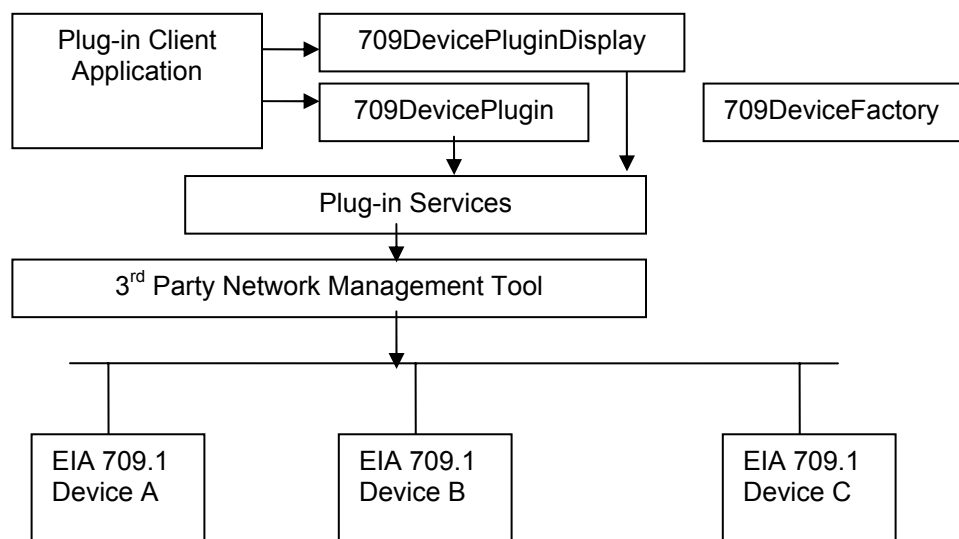


Figure 2 Control Interface Placement

8.1 Quick Reference

This section includes a summary of the methods and properties of the interfaces that make up the Control Interface component of the device plug-in. These COM interfaces, based on the Component Object Model (COM) specification revision 0.9, and their parameters and behavior are defined in later sections.