AMS Device Manager Toolkit for FOUNDATION™ Fieldbus Device Developers

Preliminary Beta 0.1 Release

EMERSON™
Process Management

April 2004

Disclaimer

Copyright and Trademark Information

AMS Suite: Intelligent Device Manager is part of the PlantWeb field-based architecture from Emerson Process Management.

Table of Contents

Overview of Related Toolkits

Section 1 Introduction

Section 2 Getting Started

Section 3 Modifying Supporting Files

Section 4 Modifying the Device and Block Resource Files

Section 5 Installing and Testing Your Device DLL

Section 6 Integrating the Device into AMS Device Manager

Overview of Related Toolkits

What is AMS Suite: Intelligent Device Manager?

AMS Device Manager is a PC-based software application developed by Emerson Process Management that allows users to monitor, manage, and regulate the devices running their process, as well as the process itself.

A plant’s assets include the valves, and transmitters that keep its processes running. Traditionally, these assets (or devices) have been managed off-line, one-by-one, using separate non-integrated PCs or software packages. Because of this, device configuration data, change logs, and device diagnostic data storage were decentralized and could not be cross-referenced. Common tasks were duplicated in separate applications and there was no easy way to view data on all devices in a plant simultaneously.

AMS Device Manager provides an efficient solution to these asset management issues by integrating the various device management software packages into a unified system with a common interface.

AMS Device Manager: An Integrated Solution for Asset Management

AMS Device Manager is an easy-to-use, fully integrated system that operates in a Microsoft® Windows® operating system environment. It combines the various asset management tasks into applications with a common user interface and centralized data storage. It helps users configure, manage calibrations, and monitor devices.

To accomplish the goal of integrated device management, AMS Device Manager leverages Emerson Process Management industry-leading technology in FOUNDATION™ fieldbus.

AMS Device Manager provides access to Resource and Transducer blocks functionality available through fieldbus communications interface for any fieldbus-compatible field device with a Device Description (DD).

AMS Device Manager communicates with SMART field devices and improves access to the information they supply. Control data is sent to the control system, but asset management data is sent directly to AMS Device Manager, enabling users to communicate digitally with all types of devices and store the information in a common database.
AMS Device Manager users can:

- Configure a device and automatically create a record of what has been done
- Monitor conditions online and set alert limits for critical parameters
- Confirm the existence of a problem in a field device and diagnose the problem from a PC located at a distance from the device
- Correlate process control and asset management data in the same time domain; make decisions and process changes based on real-time information

From the safety and comfort of the control room or the maintenance shop, operators can monitor and work with the data supplied by field devices, improving asset management and thereby reducing maintenance and downtime costs.

AMS Device Manager Functions

Non-Device-Specific Functions

Some AMS Device Manager functions are common to all field devices (i.e., non-device-specific), allowing the user to view multiple devices and perform common functions without switching applications or interfaces. Examples include:

- Browsers to find devices on the network
- Device status monitoring
- Event lists

Device-Specific Functions

For device-specific functions, AMS Device Manager uses a device's DD, but provides an easy-to-understand interface that is similar to other device interfaces. Examples include:

- Device configuration
- Device methods
- Device diagnostics

Extending AMS Device Manager Core Functionality

The core functionality of AMS Device Manager can be extended by integrating additional devices into the AMS Device Manager application or through the use of SNAP-ON™ applications.

A manufacturer or developer can integrate a FOUNDATION fieldbus device into the AMS Device Manager application by supplying the required set of device files detailed in this document.

The Development Tools

The AMS Device Manager toolkits provide the specific tools required by developers to integrate a field device into AMS Device Manager, and to provide additional device-specific functionality through AMS Device Manager. Currently, these toolkits include:

- The AMS Device Manager Toolkit for FOUNDATION Fieldbus Device Developers
- The AMS Device Manager Toolkit for SNAP-ON Developers
- The AMS Device Manager Toolkit for HART Device Developers

AMS Device Manager Toolkit for FOUNDATION Fieldbus Device Developers

The development tools in the *AMS Device Manager Toolkit for FOUNDATION Fieldbus Device Developers* include:

- *AMS Device Manager Toolkit for FOUNDATION Fieldbus Device Developers* — Step-by-step procedures so that a device will work with AMS Device Manager.
- Ancillary files, including sample files

Section 1. "Introduction" provides more information about what this Toolkit contains and how to use it.

AMS Device Manager Toolkit for SNAP-ON Developers

You can achieve further functionality with fieldbus devices integrated in AMS Device Manager through the use of SNAP-ON applications. These device-specific applications are layered on the core AMS Device Manager platform to perform additional functions for a connected fieldbus device. The tools required to develop and test a SNAP-ON application are available in the *AMS Device Manager Toolkit for SNAP-ON Developers*.

Familiarity with the AMS Device Manager OPC Server architecture is required for writing SNAP-ON applications.

NOTE: The AMS Device Manager SNAP-ON Developer’s Toolkit, is available separately through your Emerson Process Management representative.

Prerequisites for Using the AMS Device Manager Toolkits

To be able to effectively use the AMS Device Manager toolkits to integrate fieldbus devices, developers must have the knowledge base described in the following paragraphs.

Required Knowledge Base

Developers must understand how to create a Device Description (DD) in the Device Description Language (DDL). For more information about DDs and the DDL, refer to the FOUNDATION Fieldbus *Device Description Language Specification* (FF-900).

Developers also must have a basic understanding of Microsoft Windows programming, specifically the rudiments of using Microsoft Visual Studio .NET 2002 user development skills. The AMS Device Manager toolkits do not teach Microsoft Visual Studio .NET 2002 or other programming concepts. For more information, refer to Microsoft Visual Studio .NET 2002 documentation or the Microsoft Windows Software Development Kit (SDK).

Other Things a Developer Should Know

AMS Device Manager applications should follow the Microsoft Windows user interface guidelines. These can be found in the *Official Guidelines for User Interface Developers and Designers* in the Microsoft Development Library, located in the MSDN. This guide explains most of the needed user interface design details. (Note: Much of the user interface typical in DLL type applications in this document is provided by the AMS Device Manager Parameter Control interface.)

For information on developing an application for foreign-language installations, see *Developing International Software*, available from Microsoft Press, second edition; and contact the AMS Device Manager Development Group.

For basic information about installing, starting up, and using AMS Device Manager, refer to the *AMS Suite: Intelligent Device Manager Installation Guide*, available in hard copy and electronic versions. Electronic versions are supplied on the AMS Device Manager installation CD. Following the AMS Device Manager installation, they are available by selecting Start | Programs | AMS Device Manager | Help | Installation Guide.

Where to Find More Information

The following table summarizes the reference material available for AMS Device Manager developers.

Reference	**Part Number**	**Source**	**Address**	**Contact Information**
AMS Device Manager Toolkit for FOUNDATION Fieldbus Device Developers		Emerson Process Management — Asset Optimization Division	12001 Technology Drive Eden Prairie, MN 55344	AMSDeviceManagerToolkit@emersonprocess.com
AMS Suite: Intelligent Device Manager Installation Guide, Software Version 6.5	10P58246000	Emerson Process Management — Asset Optimization Division	12001 Technology Drive Eden Prairie, MN 55344	Contact your site representative or call 1-800-833-8314
AMS Device Manager Books Online		Emerson Process Management — Asset Optimization Division	12001 Technology Drive Eden Prairie, MN 55344	Contact your site representative or call 1-800-833-8314
Developing International Software		Microsoft Press, second edition	One Microsoft Way, Redmond, WA 98052-6399	www.microsoft.com
Official Guidelines for User Interface Developers and Designers		Microsoft Development Network Library		www.msdn.microsoft.com

Technical Support for AMS Device Manager

For technical assistance in using AMS Device Manager contact your site representative or call 1-800-833-8314 or 1-512-832-3774.

Technical Support for AMS Device Manager Toolkit

For technical assistance in using this toolkit contact AMSDeviceManagerToolkit@emersonprocess.com.

AMS Device Manager-Related Abbreviations and Acronyms

DD	Device Description
DDL	Device Description Language
DLL	Dynamic Link Library
FF	FOUNDATION fieldbus
HART	Highway Addressable Remote Transducer
OPC	OLE for process control
SDK	Software Development Kit (Microsoft Windows)
UI	User Interface
WRF	Windows Resource File

NOTE: A glossary of terms used in AMS Device Manager can be found in AMS Device Manager Books Online.

Section 1

Introduction

This section introduces the *AMS Device Manager Toolkit for Foundation Fieldbus Device Developers* and highlights the following information:

- Explains what's in the toolkit and who should use it
- Briefly describes the AMS Device Manager system components and how you can extend the AMS Device Manager functionality
- Describes the purpose of this document, how it is organized, and the conventions used in the document
- Directs you to the next steps in customizing the AMS Device Manager application

What's in this Toolkit?

This toolkit includes documents and associated electronic files, which together provide the software development and support services tools you need to integrate a FOUNDATION fieldbus device into the AMS Device Manager application.

Toolkit Documents

The toolkit documents include the *AMS Device Manager Toolkit for Foundation Fieldbus Device Developers* which provides step-by-step procedures for field device developers to follow in preparation for integrating a field device into the AMS Device Manager application. It gives instructions for creating a device installation kit and verifying the device interface.

Toolkit Files

The toolkit files make your development tasks easier by providing resources such as precoded functions and libraries, thus alleviating many of the labor-intensive development tasks. These files include Device Windows Resource project files.

Information about these toolkit files and how to use them is provided in the remainder of this document.

User Interface Guidelines

Devices integrated with AMS Device Manager should follow the Windows user interface guidelines. These guidelines can be found in the Microsoft® Development Library software Development Kit, *Official Guidelines for User Interface Developers and Designers*. This guide covers most of the details of user interface design you will need.

Internationalizing Your Device Application

If you are developing your device application for foreign-language installations, we recommend the book *Developing International Software*; see the *Overview of AMS Device Manager-Related Toolkits* for more information. You should also consult with the AMS Device Manager Development Group before you begin internationalizing your application.

Who Should Use This Toolkit?

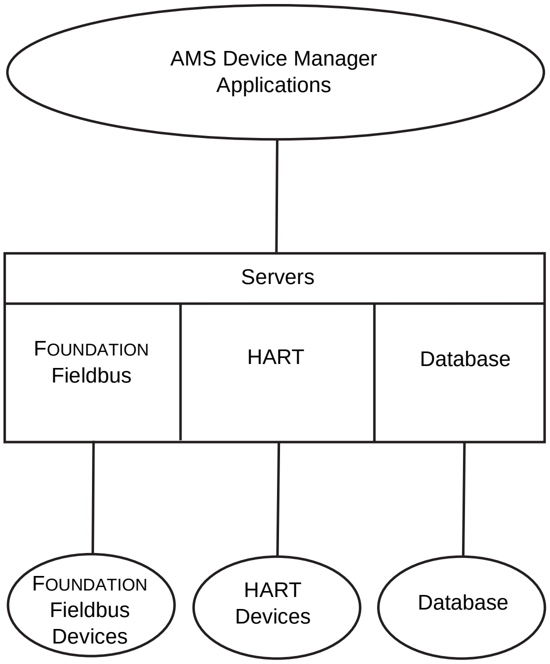
The *AMS Device Manager Toolkit for Foundation Fieldbus Device Developers* was developed for and is intended specifically for use by the FOUNDATION fieldbus device developer who wants to integrate a FOUNDATION fieldbus device files into the AMS Device Manager application. The level of the information provided in this toolkit assumes that the user:

- Has experience with Microsoft Visual Studio .NET 2002
- Understands how to write device descriptions (DDs) in the Device Description Language (DDL)

This toolkit does not provide instruction in programming, creating DDs, or tokenizing. If you need reference material on those subjects, refer to the Foundation Fieldbus specifications.

AMS Device Manager Architecture

The following diagram is a high-level view of the AMS Device Manager system components.



AMS Device Manager applications communicate with the AMS Device Manager servers, which in turn communicate with the device network and the database. Users conduct all device management tasks through the various AMS Device Manager applications.

Access to the device network and the database information is managed by the servers. Because AMS Device Manager uses a client-server architecture, applications do not have to physically reside on the same computer as the servers. This allows you to run AMS Device Manager applications (clients) on machines remote from the servers over a network. The device network in the diagram includes the actual field devices.

The database is the repository for historical device data and device configuration information.

You can integrate a fieldbus device into the AMS Device Manager application by supplying a DD and generating the required device support files. This document explains how to do this and provides files to assist you.

Purpose of this Document

This document provides a detailed, step-by-step guide for FOUNDATION fieldbus device developers to follow for integrating a field device into AMS Device Manager. It does not explain the basic functions of AMS Device Manager or the details behind the AMS Device Manager interfaces. For more information about basic AMS Device Manager functionality, refer to the *AMS Suite: Intelligent Device Manager Installation Guide* or the AMS Device Manager Books Online for related information.

This Toolkit is for AMS Device Manager versions 6.5 and AMS Inside 6.3.2 and newer.

This document describes the creation of Windows Resource files (WRF) for FOUNDATION fieldbus devices used in AMS Device Manager applications.

System Interface	Device Icon, Resource and Transducer blocks (WRF)	Device Descriptor (DDL)
DeltaV	This Toolkit	FOUNDATION fieldbus
Ovation	This Toolkit	FOUNDATION fieldbus
Rosemount Model 3420	This Toolkit	FOUNDATION fieldbus

NOTE: AMS Device Manager only interacts with Resource and Transducer blocks.

What You Should Know

This document assumes that you understand how to use the features and functions of Windows 2000 or Windows XP. If you need more information about these operating systems, refer to the documentation that came with your Operating System.

Creating custom resources for FOUNDATION fieldbus devices in AMS Device Manager requires the use of Microsoft Visual Studio .NET 2002. This document assumes that you understand how to create Resource files using Microsoft Visual Studio .NET 2002. All examples and instructions in this document use Microsoft Visual Studio .NET 2002. If you are using a different software application, please contact your AMS Device Manager representative for compatibility issues and information.

AMS Device Manager via OPC and other interfaces can display your Device Descriptor variables names to users. It is suggested to use variable names that are descriptive of the data they represent.

Refer to the appropriate programming documentation for detailed information about specific language concepts.

If you intend to have your DD/Resource file(s) integrated into standard AMS Device Manager production releases, please contact your site representative (or e-mail AMSDeviceManagerToolkit@emersonprocess.com) for specific production quality requirements. These requirements are probably no different than what your company already institutes, but this step may preclude unpleasant surprises at the time of submittal.

If you intend to localize your device resource file, contact your site representative (or e-mail AMSDeviceManagerToolkit@emersonprocess.com) to obtain information about a third-party translation house that is experienced in translating AMS Device Manager integrated devices.

Organization

This remainder of this document consists of the following sections:

- Section 2, “Getting Started,” presents an overview of the procedures to follow when customizing AMS Device Manager for a device, and introduces the supporting files used in the procedures.
- Section 3, “Modifying Supporting Files,” explains the steps you must follow to prepare supporting files.
- Section 4, “Modifying the Device and Block Resource Files,” explains how to make modifications to the example resource file to customize it for your device; and how to compile the modified resource file into the DLL files detailed in this document.
- Section 5, “Installing and Testing Your Device DLL,” explains how to install your device into an existing AMS Device Manager system and provides information on how to test it using AMS Device Manager.
- Section 6, “Integrating the Device into AMS Device Manager,” explains how to integrate your device into AMS Device Manager once the installation kit has been created.

Conventions

This document uses the typographic conventions described below.

Coding Examples

Examples of script, provided throughout the document, are shown in `Courier` (“typewriter”) monospaced font, with a rule line above and below the script. Script examples are aligned to the left margin of the page to provide space for longer lines of script. The following is an example:

```
// Style bits used by AMS controls
//
#define PS_EDIT                        0x00000001L // Edit control
#define PS_COMBO                       0x00000002L // Combo box
```

Keys on the Keyboard

Keyboard keys are shown in SMALL CAPITAL letters:

> ENTER

Key combinations are shown using the plus (+) sign:

> SHIFT + F1 indicates that you should press and hold down the SHIFT key, press F1, then release both keys.

Key sequences are shown using a comma (,):

> ESC, F1 indicates you should press ESC, release it, then press F1.

Filenames and Directories

All files and filenames in this document use MS-DOS or Windows naming conventions. Filenames and directories are represented in ALL CAPITAL letters:

> C:\WINDOWS\FMS.INI refers to the file FMS.INI, located in the WINDOWS directory on the C drive.

Commands

Commands you must enter are shown in bold Courier type, within the normal text margins:

> **`DIR C:\AMSBLD\RELEASE\*.*`** means you should type the command shown in the example, letter for letter.
>
> The phrase “Enter the following command:” means that you should type the command shown, then press the RETURN or ENTER key on the keyboard.

Variables

Variables in commands are shown in *`italicized Courier`* type:

> COPY *`source destination`* means you should enter the command “copy,” but replace source and destination with your specific information.
>
> Where necessary, the variables are defined in the text.

Variables other than in commands are shown in *italicized* text:

> *DEVICE*.RC refers to a file with the extension “.RC,” but you should replace “*DEVICE*” with the actual name of your file.

Titles of Other Publications

Titles of other publications appear in italicized text, with the publication number in parentheses following the title:

> *Device Description Language Specification* (FF-900).

What’s Next?

Section 2, “Getting Started,” guides you through the development process step-by-step.

Integrating a Field Device into the AMS Device Manager Application

To successfully customize the integration of your device into the AMS Device Manager application, you must thoroughly understand your device’s DD. You must also have a basic understanding of Microsoft Windows programming (specifically, the rudiments of UI development in Microsoft Visual Studio .NET 2002).

Section 2

Getting Started

This section provides an overview of the customization procedures and introduces the files you will be working with.

Overview of Procedures

This guide explains how to create custom resources in AMS Device Manager for a fieldbus device. Customizing AMS Device Manager for a fieldbus device means creating resource files for a device using an example resource file and the device's Device Description (DD) files.

To provide a custom interface to a fieldbus device, AMS Device Manager must have a set of Windows Resource files (detailed in this document) for the device. This is generated from a resource file project, which you create by editing the example resource file project provided.

NOTE: To accurately edit the resource files, you must have an in-depth knowledge of the device's DD.

Customizing AMS Device Manager for a Field Device: Overview of Steps

The following general steps are necessary to customize AMS Device Manager for a particular field device:

1. Modify supporting files. Several files must be modified in order to integrate the resource file into AMS Device Manager; see Section 3, “Modifying Supporting Files”. The supporting files are included with the Toolkit.
2. Modify the example resource files using information from the device’s DD; see Section 4, “Modifying the Device and Block Resource Files”.
3. Compile, integrate, and test your modified files. After you have made any additional modifications necessary, you must compile them and add them to the AMS Device Manager interface in order to test them.
4. Install and test your device DLL (detailed in this document); see Section 5, “Installing and Testing Your Device DLL”.
5. Create a “device installation kit” for installing your device into an existing AMS Device Manager system; see Section 6, “Integrating the Device into AMS Device Manager”. You must also test the installation kit itself.

Files Needed for Customizing AMS Device Manager

In the course of customizing device support for AMS Device Manager, you need the types of files listed below. You will need to create some, modify some, and simply be aware of others, as noted in their descriptions.

The Toolkit includes actual files or samples of the files you need to work with in customizing AMS Device Manager for a fieldbus device. The filenames provided are generic and must be changed to match your device following the conventions listed in the “Device Files and Directory Naming Convention” on page 2-5.

NOTE: DEVICE, as part of a filename, refers to the base name of the device.

NOTE: RESOURCE BLOCK, as part of a filename refers to the base name for a device’s Resource Block.

NOTE: TRANSDUCER BLOCK, as part of a filename refers to the base name for a device’s Transducer Block. A particular device may have more than one Transducer Block.

***XXYY*.FFO and *XXYY*.SYM**	These files are generated as output by the Fieldbus Foundation DD Tokenizer (*XX* is the device revision and *YY* is the DD revision). You will need these files when you install your device into AMS Device Manager.
***DEVICE*.CFF**	This file is a FOUNDATION fieldbus Common File Format file that contains all the information about the capabilities of a device. Creation of this file is outside the scope of this document. Refer to the *FOUNDATION Fieldbus Capability File* (FF-103) for creation of this file.
***DEVICE*.REG**	Registry files define which resource files correspond to which blocks within your device. This file is a standard Windows registry entries file and contains information that gets loaded into the Windows registry when your device support files are installed.
***DEVICE*.RC**	The device resource file contains the script that defines the device's icon and the other information related to the device interface. Create the device resource script file by modifying the sample resource file included with the Toolkit (DEVICE.RC), using the device's DD as a reference.
***DEVICE*.DLL**	This is the file that actually integrates with AMS Device Manager so that AMS Device Manager works with your device. Create this file when you compile the device resource file.
***DEVICE*.VCPROJ**	This supporting project file is used to generate the DLL (detailed in this document) from the resource file and other supporting files. Modify this file to match the specific device.
***DEVICE*.ICO**	This supporting file is the device icon of your device and needs to be referenced correctly in the resource script. For details about how this file is structured; see Section 3, "Modifying Supporting Files".
***RESOURCE BLOCK*.RC**	The device's Resource Block resource file contains the script that defines the Resource Blocks status and Configuration Displays, method names, Property Page Labels and internal strings. Create the device resource script file by modifying the sample resource file included with the Toolkit (RESOURCE BLOCK.RC), using the device's DD as a reference.
***RESOURCE BLOCK*.DLL**	This is the file that actually integrates with AMS Device Manager so it works with your device's Resource Block. This file is created when you compile the block resource file.
***RESOURCE BLOCK*.VCPROJ**	This supporting project file is used to generate your device's Resource Block DLL (detailed in this document) from the resource file and other supporting files. Modify this file to match the specific device.

TRANSDUCER BLOCK.RC	This Transducer Block(s) resource file contains the script that defines the Transducer Blocks Status and Configuration Displays, Method Names, Property Page Labels and Internal Strings. Create the device resource script file by modifying the sample resource file included with the Toolkit (TRANSDUCER BLOCK.RC), using the device's DD as a reference. **NOTE:** There is a separate project for each block.
TRANSDUCER BLOCK.DLL	This is the file that actually integrates with AMS Device Manager so that AMS Device Manager works with your devices Transducer Block(s). This file(s) is created this when you compile the block resource file.
TRANSDUCER BLOCK.VCPROJ	This supporting project file is used to generate your device's Transducer Block(s) DLL (detailed in this document) from the resource file and other supporting files. Modify this file to match the specific device.
RESOURCEBLOCK.ICO and TRANSDUCERBLOCK.ICO	These supporting file are the block icons of your device and need to be referenced correctly in the resource script. Do not modify these files.
RESOURCE.H	This file is created and used by Resource Editor to define new resources created in the resource script. **NOTE:** A RESOURCE.H file is needed for the corresponding Device, Resource Block, and Transducer Block(s) resources.
FMSDEVRC.H	This file defines the resource identifiers for the device resource script.
FMSCTL.H	This file defines constants that allow you to customize the appearance of the AMS Device Manager controls.
AFXRES.H	This standard header file, shipped with Microsoft Visual Studio .NET 2002, defines common constants in most resource scripts. Do not modify this file. **NOTE:** Since this file is shipped with Microsoft Visual Studio .NET 2002, it is not provided in this toolkit.

Device Files and Directory Naming Convention

The following four parameters in a device's Resource Block determine the location and filename of the DD files for the device: (FF-891-1.4 section 3.1.2 and section 5.3, also *Tokenizer Users Guide* (FD100)

- MANUFAC_ID (datatype unsigned32 per FF Spec)
- DEV_TYPE (datatype unsigned16 per FF Spec)
- DEV_REV (datatype unsigned8 per FF Spec)
- DD_REV (datatype unsigned8 per FF Spec)

NOTE: These parameter values are also listed in the CFF file.

For correct functionality of DD services all device DDs are placed beneath the base directory in the following format:

The <Base DD directory> for AMS Device Manager is typically = **\AMS\DEVICES\FF.**

NOTE: The format specification listed below refers to the printf format specifier notation.

and thus the directory structure to the DD files is:

<Base DD directory>\%06x<MANUFAC_ID>\%04x<DEV_TYPE>\

The example filenames below are all based on a Rosemount 3051 rev20 device with revision 2 of the DD.

FFO File

%02x<DEV_REV>%02x<DD_REV>.ffo

Ex: C:\AMS\DEVICES\FF\001151\3051\1402.FFO

SYM File

%02x<DEV_REV>%02x<DD_REV>.sym

Ex: C:\AMS\DEVICES\FF\001151\3051\1402.SYM

CFF File

%02X<DEV_REV>%02X<DD_REV>%02X<CFF_REV>.cff

Ex: C:\AMS\DEVICES\FF\001151\3051\140201.CFF

Device DLL File

%06X<MANUFAC_ID>_%04X<DEV_TYPE>_%02X<DEV_REV>.dll

Ex: C:\AMS\DEVICES\FF\001151\3051\001151_3051_14.dll

Block DLL Files

%06X<MANUFAC_ID>_%04X<DEV_TYPE>_%02X<DEV_REV>_%s.dll

The %s is really left up to the device manufacturer, here is an example of how they are named for a 3051 device:

- 001151_3051_14_R.dll (this is Resource block)
- 001151_3051_14_T.dll (this is Transducer block)
- 001151_3051_14_L.dll (this is LCD block)
- 001151_3051_14_D.dll (this is Diagnostics block)

Ex: C:\AMS\DEVICES\FF\001151\3051\001151_3051_14_R.DLL

C:\AMS\DEVICES\FF\001151\3051\001151_3051_14_T.DLL

C:\AMS\DEVICES\FF\001151\3051\001151_3051_14_L.DLL

C:\AMS\DEVICES\FF\001151\3051\001151_3051_14_D.DLL

Reg File

%06X<MANUFAC_ID>_%04X<DEV_TYPE>_%02X<DEV_REV>.reg

Ex: C:\AMS\DEVICES\FF\001151\3051\001151_3051_14.reg

The First Step

Your first step in customizing AMS Device Manager is to modify the supporting files; see Section 3, “Modifying Supporting Files”.

Section 3

Modifying Supporting Files

This section describes the steps you must follow to modify the files that support the Windows resource file.

Overview

You must modify or create the following supporting files, as described in this section:

- *DEVICE*.VCPROJ
- *DEVICE*.ICO
- RESOURCE BLOCK.VCPROJ
- TRANDUCER BLOCK.VCPROJ (There may be more than one Transducer Block in your device.)

Modify DEVICE.VCPROJ, RESOURCE BLOCK.VCPROJ and TRANSUCER BLOCK.VCPROJ

Required Modification

Change all occurrences of the string “DEVICE” in the file to *DEVICE*, where *DEVICE* is the base name of the device (for example, 001151_3051_14 for Rosemount 3051 Revision 20), see “Device Files and Directory Naming Convention” on page 2-5. Also, you must search for and replace the path, 465253\0001, with the manufacturer and device path to match your device.

NOTE: Make copies of the Transducer project for each unique transducer in your device.

Create or Modify the DEVICE.ICO File

This file contains a variety of icons used to visually represent the device in AMS Device Manager. Edit this file using a graphics program that can create exact-sized icon files so that they look like your device. The following defines the way this file should be structured.

DEVICE.ICO

This file should contain the following icon representations of your device:

- 16x16x16 color
- 32x32x16 color
- 16x16x256 color
- 32x32x256 color

Although it is not required, higher resolution displays can use 48x48x256 color and 64x64x256 color icons. You may want to include these representations in your DEVICE.ICO file to avoid having to develop these higher resolution images later.

These files must be created in a program that can save .ICO files. Use the following as a guide for creating your icon and bitmap files.

When taking a picture of the device to create an icon, here are a few tips to keep in mind:

- Natural daylight works best, as opposed to a flash. A flash tends to create a less-sharp image, and the light will generally reflect and overwhelm the picture.
- Use a light, uncluttered background to enhance the natural lighting and make it easier to remove the background in later editing.
- Take numerous pictures of the device at various angles and with various exposure times.

Once you have obtained a suitable picture of the device, use a photo-editor to erase all background from around the device itself. Convert this edited photo into the various sizes of icons you need.

NOTE: A transparent background is recommended for most icons in AMS Device Manager.

The Next Step

Once you have made all the necessary modifications to the supporting files, you can modify the resource file; see Section 4: “Modifying the Device and Block Resource Files”.

Section 4

Modifying the Device and Block Resource Files

After you have modified the supporting files, you must modify the example resource (RC) files to customize the AMS Device Manager display for your device.

This section describes the resource files in more detail and explains the sections you can modify. It also explains the effect modifications have on the AMS Device Manager interface to your device.

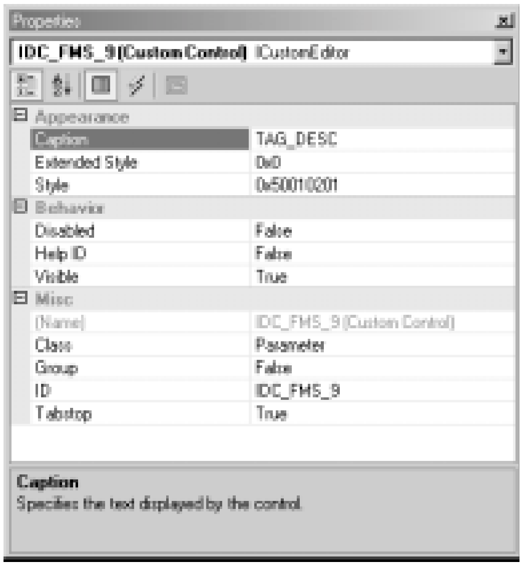
Properties Dialog

A parameter control is a custom control type in the Resource Editor. When adding or modifying a parameter custom control in the Resource Editor, the properties in Table 4-1 must be modified in the Properties dialog for each custom control; see Figure 4-1 on page 4-2.

Table 4-1. Property value

Property	Value
Caption	Must be the parameter name from the DD
Style	Determines the type and attributes of the parameter control; see “Style bits used for Control Types” on page 4-3
Class	Must be set to Parameter
ID	Must be from the fmsdevrc.h file and should be of the format IDS_FMS_XXX. Note that the order of the ID numbers determine the tab order
Tabstop	Must be set to True

Figure 4-1. Parameter Control Properties Dialog Box in Microsoft Visual Studio .NET 2002



IDC_FMS_9 (Custom Control) ICustomEditor

Appearance	
Caption	TAG_DESC
Extended Style	0x0
Style	0x50010201
Behavior	
Disabled	False
Help ID	False
Visible	True
Misc	
(Name)	IDC_FMS_9 (Custom Control)
Class	Parameter
Group	False
ID	IDC_FMS_9
Tabstop	True

Caption
Specifies the text displayed by the control.

Header Files

There are two header files that must be included by all device/block resource files. These files contain definitions that are used by the resource files: FMSCTL.H and FMSDEVRC.H.

FMSCTL.H

The first header file you will use, FMSCTL.H, is located in the INC directory. It contains the style bit constants used by the AMS Device Manager control. These constants let the device manufacturer specify the appearance of the AMS Device Manager controls on their device displays.

Style bits used for Control Types

The following style bits are used to control the appearance and behavior of your device variables on AMS Device Manager screens; see Figure 4-1).

```
//////////////////////////////////////////////////////////////////////////////////////////////
//
// Style bits used by AMS controls
//
#define PS_EDIT                             0x00000001L // Edit control
#define PS_COMBO                            0x00000002L // Combo box
#define PS_BITENUM                          0x00000010L // Bit enumerated control
#define PS_NOTPARAM                         0x00000020L // Do NOT append !Param to moniker.
#define PS_BITENUM_BOX                      0x00000040L // Bit enumerated control as check box (FF only)
#define PS_MULTILINE                        0x00000080L // MLE
#define PS_NOLABEL                          0x00000100L // Do not display Label
#define PS_NOUNITS                          0x00000200L // Do not display units
#define PS_PERCENT                          0x00000400L // Graphic Control: Percent Bar Graph.
#define PS_NOMETHODS                        0x00000800L // Suppress running pre/post edit methods.
#define PS_TEMP                             0x00001000L // Graphic control: Thermometer
#define PS_GAUGE                            0x00002000L // Graphic Control: Gauge meter
#define PS_READONLY                         0x00004000L // Force read only control
#define PS_SHORTLABELS                      0x00008000L // Use shortened label and units strings.
#define PS_LONGLABEL                        0x00040000L // Use long label length.
#define PS_NORMALLONGLABEL                  0x00080000L // Use between normal and long label.
#define PS_EX_SORT                         0x00100000L // Used to sort combo boxes
```

NOTE: The Style Property must be at least 0x50010000. Additional style bits described in “Style bits used for Control Types” must be OR’d bit-wise to this Style Property.

Each constant defines a standard control. For example, PS_EDIT defines the edit box control style bit (0x00000001). The style bits defined in FMSCTL.H are OR'd together with the standard Windows dialog control style bits to define the appearance and behavior of the AMS Device Manager controls. It is not necessary to understand how the standard Windows dialog control style bits are used in order to customize the displays for your device. For more information, refer to the Platform SDK: Windows User Interface Start Page.

The constants PS_NOLABEL and PS_NOUNITS define runtime behavior for the AMS Device Manager edit controls. If included in the overall style bits, they disable the display of the parameter label and units, respectively. Examples later in this document further explain the use of the control styles.

In addition to the style bits, AMS Device Manager controls require the resource file to provide the control with the name of the parameter that a particular instance of the control will represent. This information is stored in the caption property. Examples later in this section further explain the use of the caption property.

FMSDEVRC.H

The FMSDEVRC.H header file, located in the INC directory, contains the resource identifiers (that is, the compiler definitions of the resources) for the resources you will use. It helps AMS Device Manager ensure that all device resources are assigned the same ID for the same resource. The file contains many of the resource identifiers used in device/block resource files. For example, it assigns the resource name IDI_DEVICE to the device icon, so that all device icon filenames do not have to be hard-coded into AMS Device Manager.

It is not necessary to completely understand the details of the resource definitions and identifiers, but this file must be included in the Resource Includes dialog.

The following is a portion of the FMSDEVRC.H file, with comments that explain the different definition sections. Use this information as a reference. Do not make modifications to this file.

```
/////////////////////////////////////////////////////////////////////////////////
// Device View global resource definitions.

// Generic device resource IDs
#define IDA_DEVICE            100   // Device accelerator table
```

Bitmap and Icon Definitions

The following two lines define the IDs for the device bitmap and the device icon.

```
#define IDB_DEVICE            101   // Device bitmap
#define IDI_DEVICE            102   // Device icon (Standard 32x32)
```

Device Information

The next five lines define the IDs for the specific strings that identify the device. The actual values of the strings are set in the string table section of the resource file; see “Internal Strings” on page 4-12).

```
#define IDS_MANUFACTURER      103   // Device manufacturer string
#define IDS_TYPE              104   // Device type string
#define IDS_REVISION          105   // Device revision string
```

Device Resource File (Device.rc)

Once you understand the header files, you can examine the default resource script and determine where you need to modify it. Use Microsoft Visual Studio Resource Editor to modify the files.

NOTE: This section makes extensive use of the resources defined in “Header Files” on page 4-2.

The resource script defines the interface, as well as the device functions. The following text defines each section of the default resource file, with a representation of the resulting interface, where applicable. See “DEVICE.RC” on page 2-3.

Header

The first section of the resource script is a default header that defines the necessary auxiliary header files.

```
// Prototype Resource File script.
//
#include "resource.h"
```

Microsoft Visual Studio .NET 2002 creates RESOURCE.H when defining resources. If you add new resources to the resource script (not new AMS Device Manager controls), Resource Editor places the resource IDs in RESOURCE.H. If you want to add AMS Device Manager controls to the dialog templates, use the IDC_FMS_XXX identifiers provided in FMSDEVRC.H.

NOTE: Occasionally, when you add controls to a screen and recompile your DLL (detailed in this document), you may encounter a warning message from Resource Editor indicating that you have multiple defined controls. In this case, Resource Editor is incorrectly adding an AMS Device Manager control (IDC_FMS_X) to RESOURCE.H that already exists in FMSDEVRC.H. Simply delete the offending control from your RESOURCE.H file, recompile, and the warning message will go away.

```
#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
#include "fmsctl.h"
#include "fmsdevrc.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS
```

NOTE: The header file AFXRES.H ships with Microsoft Visual Studio .NET 2002 as a standard header file used to define common constants in most resource scripts.

Device Icon

The next line of the resource file defines the icon for the device.

```
/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

IDI_DEVICE              ICON    DISCARDABLE     "device.ico"
```

The icons are used to represent your device on various displays. You must create the icon file. For more information about creating these files; see “Create or Modify the DEVICE.ICO File” on page 3-2.

Block Resource Files (ResourceBlock.rc and TransducerBlock.rc)

This section describes the contents of a block resource file. See “RESOURCE BLOCK.RC” and “TRANSDUCER BLOCK.RC” on page 2-4.

Header

See “Header” on page 4-5 for more information.

Block Item

The next section of the resource file defines the icon for the block. The icons are used to represent the block on various displays. In order to provide consistency across devices, you should use the block icons provided by the AMS Device Manager group.

This line and the resourceblock.ico should not be changed.

Figure 4-2. ResourceBlock.rc file example

```
/////////////////////////////////////////////////////////////////////////////
//
// Icon
//
IDI_DEVICE              ICON    DISCARDABLE     "resourceblock.ico"
```

Status Display

The Status Display appears when you right-click the device’s context menu within the AMS Device Manager and select the “Status / Conditions” menu item.

The first property page must be identified using the IDD_STATUSDISPLAY_10. Additional pages are identified by incrementing the 'tens' position. In other words, the second and third pages are identified using IDD_STATUSDISPLAY_20 and IDD_STATUSDISPLAY_30, respectively. The coordinates of each property page must be (0,0) and the recommended size of each property page is 416x200.

The caption field of the parameter control specifies the status bit to be displayed. The general form of the captions is as follows:

parameter!Y

where parameter is the name of the status parameter and Y refers to the actual bit-enumerated value within the parameter. The parameter name must match the name of the parameter as it appears in the parameter list in the device's Device Description.

You must use the parameter control identifiers IDC_FMS_XXX. These identifiers are defined in FMSDEVRC.H. There are 100 identifiers available for use, defined as IDC_FMS_1 through IDC_FMS_100.

The example below demonstrates the use of GROUPBOX controls to group similar status bits. Note that you cannot place a control on top of another control. The controls must always be accessible at run-time by the mouse pointer.

In addition to WS_TABSTOP, the controls use the 0x310 style, where 0x0010 defines the control as a bit-enumerated type, 0x0100 suppresses any labels, and 0x0200 suppresses units.

The following example shows the definition of the Alarms property page of the standard Resource Block. A complete listing of Status display examples can be found with the Toolkit.

```
/////////////////////////////////////////////////////////////////////////////
//
// Status Display - Alarm Page
//
IDD_STATUSDISPLAY_10 DIALOGEX 0, 0, 416, 200
STYLE WS_CHILD
EXSTYLE WS_EX_CONTROLPARENT
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "BLOCK_ERR!16384",IDC_FMS_15,"Parameter",WS_TABSTOP | 0x310,20,20,170,13
    CONTROL         "BLOCK_ERR!2",IDC_FMS_2,"Parameter",WS_TABSTOP | 0x310,20,40,170,13
    CONTROL         "BLOCK_ERR!4",IDC_FMS_3,"Parameter",WS_TABSTOP | 0x310,20,61,170,13
    CONTROL         "BLOCK_ERR!8",IDC_FMS_4,"Parameter",WS_TABSTOP | 0x310,20,82,170,13
    CONTROL         "BLOCK_ERR!16",IDC_FMS_5,"Parameter",WS_TABSTOP | 0x310,20,103,170,13
    CONTROL         "BLOCK_ERR!32",IDC_FMS_6,"Parameter",WS_TABSTOP | 0x310,20,124,170,13
    CONTROL         "BLOCK_ERR!256",IDC_FMS_9,"Parameter",WS_TABSTOP | 0x310,20,145,170,13
    CONTROL         "BLOCK_ERR!512",IDC_FMS_10,"Parameter",WS_TABSTOP | 0x310,20,166,170,13
    CONTROL         "BLOCK_ERR!1024",IDC_FMS_11,"Parameter",WS_TABSTOP | 0x310,225,21,170,13
    CONTROL         "BLOCK_ERR!64",IDC_FMS_7,"Parameter",WS_TABSTOP | 0x310,225,42,170,13
    CONTROL         "BLOCK_ERR!128",IDC_FMS_8,"Parameter",WS_TABSTOP | 0x310,225,63,170,13
    CONTROL         "BLOCK_ERR!2048",IDC_FMS_12,"Parameter",WS_TABSTOP | 0x310,225,83,170,13
    CONTROL         "BLOCK_ERR!4096",IDC_FMS_13,"Parameter",WS_TABSTOP | 0x310,225,104,170,13
    CONTROL         "BLOCK_ERR!8192",IDC_FMS_14,"Parameter",WS_TABSTOP | 0x310,225,126,170,13
    CONTROL         "BLOCK_ERR!32768",IDC_FMS_16,"Parameter",WS_TABSTOP | 0x310,225,147,170,13
    CONTROL         "BLOCK_ERR!1",IDC_FMS_1,"Parameter",WS_TABSTOP | 0x310,225,168,170,13
    GROUPBOX        "",IDC_STATIC,215,6,191,184
    GROUPBOX        "",IDC_STATIC,9,6,191,184
END
```

Configuration Display

The Configuration display appears when you right-click the device's context menu within the AMS Device Manager and select “Configure”. Use Microsoft Visual Studio Resource Editor to modify the files.

The first property page must be identified using the IDD_CONFIGDISPLAY_10. Additional pages are identified by incrementing the 'tens' position. In other words, the second and third pages are identified using IDD_CONFIGDISPLAY_20 and IDD_CONFIGDISPLAY_30, respectively. The coordinates of each property page must be (0,0) and the recommended size of each property page is 416x200.

The caption field must contain the name of the parameter to be displayed by the control. This must match the name of the parameter as it appears in the parameter list in the device's Device Description.

You must use the control identifiers IDC_FMS_XXX. These identifiers are defined in FMSDEVRC.H. There are 100 identifiers available for use, defined as IDC_FMS_1 through IDC_FMS_100. You cannot place a control on top of another control. The controls must always be accessible at run-time by the mouse pointer.

The following example shows the definition of the Identification property page for the Configuration display of the standard Resource Block. A complete listing of the Configuration Display can be found with the Toolkit.

```
/////////////////////////////////////////////////////////////////////////////
//
// Configuration Display - Identification Page
//
IDD_CONFIGDISPLAY_10 DIALOGEX 0, 0, 416, 200
STYLE WS_CHILD
EXSTYLE WS_EX_CONTROLPARENT
FONT 8, "MS Sans Serif", 0, 0, 0x1
BEGIN
    CONTROL         "MANUFAC_ID",IDC_FMS_6,"Parameter",WS_TABSTOP | 0x301,86,
                    22,65,14
    CONTROL         "DEV_TYPE",IDC_FMS_1,"Parameter",WS_TABSTOP | 0x301,86,
                    49,65,14
    CONTROL         "DEV_REV",IDC_FMS_2,"Parameter",WS_TABSTOP | 0x301,86,76,
                    65,14
    CONTROL         "DD_REV",IDC_FMS_7,"Parameter",WS_TABSTOP | 0x301,86,103,
                    65,14
    CONTROL         "Characteristics!Member!BLOCK_TAG",IDC_FMS_13,"Parameter",
                    WS_TABSTOP | 0x321,86,130,156,14
    CONTROL         "TAG_DESC",IDC_FMS_9,"Parameter",WS_TABSTOP | 0x301,86,
                    157,156,14
    LTEXT           "Manufacturer",IDC_STATIC,25,22,50,15,SS_CENTERIMAGE |
                    NOT WS_GROUP
    LTEXT           "Device Type",IDC_STATIC,25,49,50,15,SS_CENTERIMAGE |
                    NOT WS_GROUP
    LTEXT           "Device Revision",IDC_STATIC,25,76,54,15,SS_CENTERIMAGE |
                    NOT WS_GROUP
    LTEXT           "DD Revision",IDC_STATIC,25,103,50,15,SS_CENTERIMAGE |
                    NOT WS_GROUP
    LTEXT           "Tag ",IDC_STATIC,25,130,57,15,SS_CENTERIMAGE
    LTEXT           "Tag Description ",IDC_STATIC,25,157,57,15,
                    SS_CENTERIMAGE
END
```

Transmitter Mode Validity

Multiple Mode Devices

A device block can have multiple "modes" in which sets of variables become invalid while other variables are valid. The AMS Device Manager allows the block property pages to be defined in a way that adapts to these various modes of the device block. Separate sets of Configuration and Status display property pages can be created for each mode of the device block. A maximum of eight modes is supported. The "validity" of specific parameters can be used to inform AMS Device Manager the mode of a device block, which in turn causes AMS Device Manager to load the appropriate set of Configuration or Status property pages.

Up to three Primary Validity Variables can be defined in the block resource string table. Each validity variable defined will be checked for validity and used to determine the mode of the device block. With no validity variables defined the block has only one mode. If one validity variable is defined in the block resource file, the block has two possible modes depending on whether that variables validity property is true or false. Two validity variables defined in the block string table give four possible modes and three validity variables allow for eight modes. Each validity variable defines one bit (from least to most significant) of the mode of the block.

Figure 4-3. Mode ID's defined in FmsDevRc.h

```
#define IDS_PVV_0 900
#define IDS_PVV_1 901
#define IDS_PVV_2 902
```

Below is a sample using mode IDs used in the TransducerBlock.rc String Table and an explanation of the updates to the Configuration and Status displays.

```
IDS_PVV_0 param!SENSOR_CONNECTION_1
IDS_PVV_1 param!SENSOR_CONNECTION_2
```

If SENSOR_CONNECTION_1 is valid but SENSOR_CONNECTION_2 is not, the name of the first property page of the Configuration display is “IDD_CONFIGDISPLAY_11”. The tens digit specifies that it is the first page; the ones digit is 1, because only bit 1 (or SENSOR_CONNECTION_1) is set.

Similarly, if SENSOR_CONNECTION_1 is valid and SENSOR_CONNECTION_2 is also valid, the name of the second property page of the Status display would be “DD_STATUSDISPLAY_23”. The tens digit specifies that it is the second page; the ones digit is three, since bit one and bit two are set (i.e. SENSOR_CONNECTION_1 and SENSOR_CONNECTION_2 are both valid).

Context Menu

The device developer may add items to its device context menu to launch the method wizard. The menu items defined in the block context menu will be merged with the standard items that appear on every device context menu.

Methods are referenced from the context menu using a string number for the help string associated with that method. An offset into another table then maps that string to the actual method name used in the DD.

The first method help string is defined as CM_METHOD1, which is defined in FMSDEVRC.H. The first method referenced in the context menu should have this value. Each additional method referenced in the context menu should have an incremental value.

```
/////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDM_DEVICE MENU DISCARDABLE
BEGIN
    POPUP ""
    BEGIN
        MENUITEM "&Status...",                  CM_STATUSDISPLAY
        POPUP "Calibrate"
        BEGIN
            MENUITEM "Zero trim",                   CM_METHOD3
            POPUP "Sensor Trim"
            BEGIN
                MENUITEM "Lower sensor trim",           CM_METHOD1
                MENUITEM "Upper sensor trim",           CM_METHOD2
            END
            MENUITEM "Factory Calibration",         CM_METHOD4
        END
    END
END


/////////////////////////////////////////////////////////////////////////////
```

Context Menu Help Strings

The following string table entries provide AMS Device Manager with suitable help strings to display on the status bar while the user scrolls through the block's context menu. Note that the string identifiers must precisely match the context menu command identifiers.

```
/////////////////////////////////////////////////////////////////////////////
//
// Help Strings
//
STRINGTABLE DISCARDABLE
BEGIN
    CM_STATUSDISPLAY    "Display selected device's status."
END
STRINGTABLE DISCARDABLE
BEGIN
    CM_CONFIGDISPLAY    "Display selected device's configuration view."
END
STRINGTABLE DISCARDABLE
BEGIN
    CM_METHOD1          "Master Reset- Commands the electronics to reset and perform a self
                                    test."
    CM_METHOD2          "Self Test- Commands the electronics to perform a self test."
END
```

Method Names

When the user selects a method from the block's context menu, AMS Device Manager locates a string table entry that corresponds to the command identifier associated with the selected method. It does this by taking the difference between the command identifier of the given method and CM_METHODS and then adds it to the constant IDS_METHODS to obtain a string table identifier. This string contains the name of the method.

```
/////////////////////////////////////////////////////////////////////////////
//
// Methods Names
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_METHOD1             "Method!master_reset"
    IDS_METHOD2             "Method!self_test"
END
```

Property Page Labels

The property page labels are the labels that appear on the tabs of the Status and Configuration displays.

```
////////////////////////////////////////////////////////////////////////////
//
// Status Tabs
//

STRINGTABLE DISCARDABLE
BEGIN
    3010                    "Alarms"
    3020                    "Process"
END
////////////////////////////////////////////////////////////////////////////
//
// Property Tabs
//

STRINGTABLE DISCARDABLE
BEGIN
    7010                    "Identification"
    7020                    "Process"
    7030                    "Alarms"
    7040                    "Hardware"
    7050                    "Options"
END
```

Internal Strings

These strings are used internally by AMS Device Manager. They should not be altered and are shown here for informational purposes only.

```
////////////////////////////////////////////////////////////////////////////
//
// Internal Strings
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_CONFIGDISPLAY       "IDD_CONFIGDISPLAY_"
END
STRINGTABLE DISCARDABLE
BEGIN
    IDS_STATUSDISPLAY       "IDD_STATUSDISPLAY_"
END
```

User Interface Guidelines

General Guidelines

The following guidelines apply to all displays.

1. Abbreviations should not be used.
2. All parameter labels should end with a colon.
3. The order of the IDS_FMS_XXX IDs controls the tab order in the dialog so they should be numbered to correctly control the tab order. Tab order should flow top to bottom and left to right. A group box should be considered one control, but the controls within it should follow the same rules.
4. Whenever possible static labels should not be used in the dialog. If possible these should be filled in with values from the DDs. If static labels are used they should match up with the values that are in the DDs so differences are not seen between the Configuration and Compare screens as an example.
5. Checkbox parameters should include checkbox and label/description to allow focus to be set to checkbox controls.
6. Checkbox parameters should not be allowed to be checked if the option is not even available - as an example Block Execution and Features checkboxes.

Status Display Guidelines

These guidelines apply to the Status display.

1. Similar status conditions should be grouped together in a group box.
2. All status conditions should have help defined for them in the device's Device Description.
3. This display may show parameters that are changing dynamically.
4. The size of each property page should be 416x200.

Configuration Display Guidelines

The guidelines apply to the Configuration display.

1. Similar device conditions should be grouped together in a group box.
2. All device parameters should have help defined for them in the device's Device Description.
3. This display should not show parameters that are changing dynamically.
4. The size of each property page should be 416x200.

Compiling the Resource File into a DLL

When you have made the necessary modifications to the resource file, you can compile it into a .DLL. There are two ways to do this: through Microsoft Visual Studio .NET 2002, or from the command line.

Compiling Through Visual Studio .NET 2002

To compile the resource file into a .DLL through Microsoft Visual Studio .NET 2002, first load the file *DEVICE*.VCPROJ (where *DEVICE* is the base name of the device—for example, 001151_3051_14.VCPROJ). Use the Project / Build menu to compile the file.

Compiling From the Command Line

To compile the file through the command line, use the following command:

```
nmake /f “DEVICE.VCPROJ”
```

where *DEVICE* is the base name of the device (for example, 001151_3051_14.VCPROJ).

NOTE: In order for Microsoft Visual Studio .NET 2002 to build your DLL (detailed in this document), the INC files needed by the resource file must be properly installed. See “Modify DEVICE.VCPROJ, RESOURCE BLOCK.VCPROJ and TRANSUCER BLOCK.VCPROJ” on page 3-2 for instructions on setting up the INC directories.

Creating additional supporting files

Registry file

Registry files define which resource files correspond to which blocks within your device. This file is a standard Windows registry entries file and contains information that gets loaded into the Windows registry when your device support files are installed.

There must exist a section for the device resource file and each of your block resource files. The names of the sections all begin with HKEY_LOCAL_MACHINE\Software\FRSI\DeltaV\CurrentVersion\AMS. The remainder of the section name specifies the device or block for which it pertains. The remainder of the section name takes format M.T.R where M, T, and R are the manufacturer id, device type and device revision respectively. Using the printf format specifier notation this is: %06x<MANUFAC_ID>.%04x<DEV_TYPE>.**%02d<DEV_REV>**

Each section contains a single key. For the device section, this key is called "DeviceDLL". For the block sections, this key is called "BlockDLL". Each key specifies the path to the resource file to be used for the given device or block. This path could be relative to the DeltaV DVData\amsdevices directory.

The following is an example of a Rosemount 3051 revision 20 registry file (001151_3051_14.reg).

NOTE: It is very important to note that even though the device's resource filenames show the device revision in a hexidecimal notation, the contents of the registry file KEYS must show them in decimal notation.

```
REGEDIT4
[HKEY_LOCAL_MACHINE\SOFTWARE\FRSI\DeltaV\CurrentVersion\AMS\001151.3051.20]
"DeviceDLL"="001151\\3051\\001151_3051_14.DLL"
[HKEY_LOCAL_MACHINE\SOFTWARE\FRSI\DeltaV\CurrentVersion\AMS\001151.3051.20\Block1000]
"BlockDLL"="001151\\3051\\001151_3051_14_R.DLL"
[HKEY_LOCAL_MACHINE\SOFTWARE\FRSI\DeltaV\CurrentVersion\AMS\001151.3051.20\Block1100]
"BlockDLL"="001151\\3051\\001151_3051_14_T.DLL"
[HKEY_LOCAL_MACHINE\SOFTWARE\FRSI\DeltaV\CurrentVersion\AMS\001151.3051.20\Block1200]
"BlockDLL"="001151\\3051\\001151_3051_14_L.DLL"
[HKEY_LOCAL_MACHINE\SOFTWARE\FRSI\DeltaV\CurrentVersion\AMS\001151.3051.20\Block1300]
"BlockDLL"="001151\\3051\\001151_3051_14_D.DLL"
```

The Next Step

After you have modified the supporting files and the resource file, and have compiled the resource file into a DLL (detailed in this document), and have created a registry file you are ready to install and test your device; see Section 5, “Installing and Testing Your Device DLL”.

Section 5

Installing and Testing Your Device DLL

This section explains how to install your device into an existing AMS Device Manager system to test your device interface.

Files and Directories You Will Use

Use the following files and directory structure to install and test your device with AMS Device Manager:

***DEVICE*.DLL**	DEVICE.DLL is the DLL file you created when you compiled the device resource file and supporting files; see "Compiling the Resource File into a DLL" on page 4-14.
RESOURCE BLOCK.DLL	RESOURCE BLOCK.DLL is the DLL file you created when you compiled the device resource file and supporting files; see "Compiling the Resource File into a DLL" on page 4-14.
TRANSDUCER BLOCK.DLL	TRANSDUCER BLOCK is the DLL file you created when you compiled the device resource file and supporting files; see "Compiling the Resource File into a DLL" on page 4-14. **NOTE:** You may have more than one Transducer Block DLL.

***XXYY*.FFO and *XXYY*.SYM**	These files are generated as output by the FOUNDATION fieldbus DD Tokenizer, where *XX* is the device revision (in hex) and *YY* is the device description revision (in hex).
DEVICE.REG	This is the registry file you created in the previous section which points to the location of the devices resource files.
DEVICE.CFF	This file, is a FOUNDATION fieldbus Capability file that contains all the information about the capabilities of a device.

Hardware You Will Use

You need the following hardware to test your device's DLLs:

1. An AMS Device Manager supported FOUNDATION fieldbus communication interface, e.g., a 3420 Fieldbus Interface Module or a DeltaV digital automation system.
2. A fieldbus terminator to connect your device.
3. A device connected to AMS Device Manager.

Assumptions

This section assumes that:

- You have installed AMS Device Manager.

  If you need more information about installing AMS Device Manager; see the *AMS Suite: Intelligent Device Manager Installation Guide*.

- You have created a DEVICE.DLL, RESOURCE BLOCK.DLL, and TRANDUCER BLOCK.DLL files.

  If you need more information about creating device DLL files (detailed in this document). See "Modifying the Device and Block Resource Files" on page 4-1.

- You have created a DEVICE.REG file; see "Registry file" on page 4-14.
- You have FOUNDATION fieldbus tokenized DD files XXYY.FFO and *XXYY*.SYM files.
- You have a DEVICE.CFF file.

Overview of Device Installation and Test Steps

The following is an overview of the steps required for installing and testing your device. Each step is explained in detail in the next part of this section. (The steps are numbered T-1 through T-5, to distinguish them as “Test” steps.)

1. Add your device files to AMS Device Manager.
2. Start AMS Device Manager Explorer and test your device.
3. Modify your resource file (if required) and rebuild your DLLs.
4. Copy your new DLLs into the AmsDevices directory.
5. Repeat steps T2 - T4 as needed to complete your DLL tests.

Step T-1: Add your device files to AMS Device Manager

Follow these steps carefully:

1. Create a folder anywhere on your computer and place all your device support files into that folder. It does not matter what you call the folder. You can also copy them to a floppy disk if you wish.
2. From the Start menu, select Start | Programs | AMS Device Manager | Add Device Type Manually, see the AMS Device Manager Release Notes for detailed instructions on FOUNDATION fieldbus device installation.
3. A wizard appears and guides you through the process of installing your device support files. The first thing you will be asked to do is to browse to the directory containing your support files. This is the directory that was created in Step 1. After that simply follow the direction presented by the wizard. Once the wizard completes, your device support files have been installed. At this point you should be able to connect your device to AMS Device Manager and test whether your support files have been developed correctly.

Step T-2: Start AMS Device Manager Explorer and Test Your Device

Once your device is connected to AMS Device Manager, test the functionality of your device DLL (detailed in this document):

1. Right-click your device to open the device context menu. Verify that the context menu looks as intended.
2. From the device context menu, click Status/Conditions. Verify the screens appear as intended. The device’s resource block and transducer block icons appear in the left pane of the Status/Conditions screen. Verify the correct tabs and contents in the right pane for all the blocks displayed in the left pane of the screens. If possible, cause your device to report an error status and verify that the correct status is displayed.
3. From the device context menu, click Configure. Verify that the Configuration Properties screen looks as intended. Verify that each parameter intended to be editable from the Configuration Properties screen is editable. Using the Block Navigator in the left pane of the screen, verify the correct tabs and content for all the blocks on the right pane. Ensure that all writable values on the screen actually write to the device successfully when you select **Apply**. Verify labels, tab order, and parameter Help.
4. From the device context menu, click Compare... Verify that the Compare screens look as intended. Verify that each parameter intended to be editable from the Compare screen is editable. Using the Block Navigator in the left pane of the screen, verify the correct tabs and content for all the blocks on the right pane. Verify that you can write values from the placeholder to the physical device and vice versa. Verify that any differences between the placeholder and physical device have their tabs colored green. Verify labels, tab order, and parameter Help.
5. From the device context menu, click Audit Trail and verify that changes to your device have been recorded here.
6. From the device context menus test all methods available and verify they complete without errors. Verify method Help on the status bar.

NOTE: These steps for testing are not meant to be all-inclusive, but only a guideline for testing your device DLL (created from this document).

Step T-3: Modify Your Resource File and Rebuild Your DLL

If you discover a problem with your DLL in Step T-2, edit your resource file to fix the problem and rebuild your DLL; see “Modifying the Device and Block Resource Files” on page 4-1 for more information on modifying the resource file and building the DLL.

Step T-4: Copy Your New DLL into the AMSdevices Directory

Once you have modified your resource file and rebuilt your DLL as needed, test your changes by adding your updated device files to AMS Device Manager; see “Step T-1: Add your device files to AMS Device Manager”.

Step T-5: Repeat Steps T-2—T-4 as Needed

While you are testing your device DLL files (detailed in this document), you may decide to make some changes. If so, repeat Steps T-2 through T-4 until you are satisfied with the functioning of your device with AMS Device Manager.

As part of this testing, you may need to modify the DD and/or CFF file for your device. If so, make the modifications, tokenize the DD, and add the updated files to AMS Device Manager; see “Step T-1: Add your device files to AMS Device Manager”.

Log Files for Debugging Your Device

The Windows Event Viewer can be used to view error messages that are not necessarily sent to the user interface.

Troubleshooting

If you have any problems testing your DLL (detailed in this document), you may find the following troubleshooting suggestions helpful:

- If the AMS Device Manager application issues error messages (indicating no moniker or no path to parameter) when it tries to display a dialog, do the following:
  - Check the parameter name in the resource file.
  - Check to make sure the DLL parameters match the resource file parameters.
- If the AMS Device Manager application issues an error message (indicating resource not available) when it tries to display a dialog, do the following:
  - Check the string table entries for the dialog headers. Ensure that they match the configuration tabs. It is probable that either a string table header entry has been deleted and the corresponding configuration dialog template is still around, or vice-versa.

The Next Step

The next step in customizing AMS Device Manager for your device is to integrate your device with AMS Device Manager, Section 6, “Integrating the Device into AMS Device Manager,”

Section 6

Integrating the Device into AMS Device Manager

This section explains how to integrate your device into AMS Device Manager once the installation kit has been created.

Methods for Integrating into AMS Device Manager

There are two methods for integrating the device into AMS Device Manager. The first is through an independent integration which calls for distribution by your company of the device support files created in the previous sections and sending this disk directly to the customer. The second method is through a full integration, whereby the device is distributed with AMS Device Manager to all customers automatically.

Independent Integration into AMS Device Manager

This method is useful when putting out modifications or new releases of devices which don't coincide with a scheduled release of AMS Device Manager.

Additionally, Emerson Process Management is currently developing a web site to serve as a convenient location for customers to locate Device Installation Kits and for device developers to make updates available more quickly to their customers.

This service is currently under development. Contact your representative or e-mail: AMSDeviceManagerToolkit@emersonprocess.com.

Full Integration into AMS Device Manager

This method allows the device to be included with the release of AMS Device Manager to all customers. The AMS Device Manager build process requires all device support files be supplied to Emerson Process Management. This includes the following files:

DEVICE.DLL

RESOURCE BLOCK.DLL

TRANDUCER BLOCK.DLL(s)

DEVICE.FFO

DEVICE.SYM

DEVICE.CFF

DEVICE.REG

NOTE: Refer to the Naming Conventions guidelines in this document for actual filenames.

The Integration Timeline

Contact the AMS Device Manager Development Team for a schedule of releases via AMSDeviceManagerToolkit@emersonprocess.com.

The Next Step

If you have questions about the integration process or any issues with the integration of your device into AMS Device Manager, contact the AMS Device Manager Development Team. If you don’t have a site contact, e-mail AMSDeviceManagerToolkit@emersonprocess.com.