DM3 Standard Component Interface Messages

05-1040-001

Publication Date: April, 1998

Part Number: 05-1040-001

Dialogic Corporation
1515 Route 10
Parsippany NJ 07054

Technical Support
Phone: 973-993-1443
Fax: 973-993-8387
BBS: 973-993-0864
Email: CustEng@dialogic.com

For **Sales Offices** and other contact information, visit our website at **http://www.dialogic.com**

Table of Contents

1. Introduction 1

1.1. About This Guide 1

1.1.1. Information in This Guide 1

1.1.2. Other Relevant Guides and References................................. 1

1.1.3. Glossary Definitions 2

1.1.4. Typeface Conventions 2

1.2. Key DM3 Architecture Concepts 2

2. Standard Component Interface Messages 5

Std_MsgAck - the command has been received but not yet executed 10

Std_MsgArmRTC - arms a component instance for an RTC action 12

Std_MsgArmRTCCmplt - component instance is armed for an RTC action 15

Std_MsgArmxRTCs - arms a component instance for multiple RTC actions........ 16

Std_MsgArmxRTCsCmplt - component instance is armed for multiple RTC actions 19

Std_MsgCancelAllEvts - cancels the reporting of all events................................ 20

Std_MsgCancelAllEvtsCmplt - events will not be reported................................. 22

Std_MsgCancelEvt - cancels the reporting of a single event 23

Std_MsgCancelEvtCmplt - verifies the event will not be reported....................... 25

Std_MsgCancelxEvts - disables the reporting of multiple events 26

Std_MsgCancelxEvtsCmplt - verifies the events will not be reported 29

Std_MsgComtest - establishes a communications path with a component or instance 30

Std_MsgComtestCmplt - verifies a communications path with a component or instance 32

Std_MsgDetectEvt - configures a component instance to detect a specific event. 33

Std_MsgDetectEvtCmplt - component instance can detect a specific event 36

Std_MsgDetectxEvts - configures a component instance to detect multiple events 37

Std_MsgDetectxEvtsCmplt - verifies the component instance can detect multiple events 40

Std_MsgDisarmAllRTCs - disables all RTC actions.. 41

Std_MsgDisarmAllRTCsCmplt - component instance will not perform any RTC actions 43

Std_MsgDisarmRTC - disables an RTC action.. 44

Std_MsgDisarmRTCCmplt - component instance will not perform an RTC action 46
Std_MsgDisarmxRTCs - disables multiple RTC actions 47
Std_MsgDisarmxRTCsCmplt - component instance will not perform the RTC actions 50
Std_MsgError - indicates an error in executing a command message 51
Std_MsgEvtDetected - indicates an event has occurred 53
Std_MsgExecute - executes the function contained in the message 55
Std_MsgExecuteCmplt - component instance has executed the encapsulated function 57
Std_MsgExit - shuts down a component instance 58
Std_MsgExitCmplt - verifies the component instance has shut down 60
Std_MsgGetParm - requests a parameter value 61
Std_MsgGetParmCmplt - contains a requested parameter value 63
Std_MsgGetxParms - requests multiple parameter values 65
Std_MsgGetxParmsCmplt - contains requested parameter values 68
Std_MsgInit - initializes a component instance 70
Std_MsgInitCmplt - verifies the component instance is initialized 72
Std_MsgSetAllParmsDef - sets all parameters to their default values 73
Std_MsgSetAllParmsDefCmplt - verifies all parameters are set to their default value 75
Std_MsgSetParm - changes the value of a specific parameter 76
Std_MsgSetParmCmplt - verifies the parameter is set 78
Std_MsgSetParmDef - sets a specific parameter to its default value 79
Std_MsgSetParmDefCmplt - verifies the parameter is set to its default value 81
Std_MsgSetxParms - changes the value of multiple parameters 82
Std_MsgSetxParmsCmplt - verifies the parameters are set 85
Std_MsgSetxParmsDef - sets multiple parameters to their default values 86
Std_MsgSetxParmsDefCmplt - parameters are set to their default values 89

3. Event Notification 91

3.1. About Event Notification and RTC Actions 91
3.2. Event Registration 92
3.3. DM3 Resource Events 92

Appendix A: Parameters 93

Appendix B: Error Codes 95

How Error Notification Works 95
Standard Component Error Codes 95

Index 99

1. Introduction

1.1. About This Guide

1.1.1. Information in This Guide

This document describes the DM3 GlobalCall Resource, including the various parts of the resource, the message sets that the resource employs, and how to use the resource's features.

This document contains the following chapters:

Chapter 2 lists and explains the messages in the Standard Component Message Set.

Chapter 3 discusses event notification.

Appendix A describes the parameters of DM3 resources.

Appendix B lists and explains the error codes of DM3 resources.

1.1.2. Other Relevant Guides and References

This user's guide discusses topics pertaining globally to resources on the DM3 platform. Certain concepts are discussed in more detail in other publications, and are referred to throughout this resource guide:

- **the DM3 architecture:** For an overview of the DM3 mediastream architecture, see the *DM3 Mediastream Architecture Overview* guide (part number 05-0813-001).
- **resource-specific messages:** For detailed information about the proprietary messages used by each DM3 resource, see the user's guide for that resource.

1.1.3. Glossary Definitions

A term being defined for the first time will be in boldface, followed by its definition.

1.1.4. Typeface Conventions

The following typeface conventions are used throughout this guide:

- function names are boldface, lowercase, and followed by parentheses. Example: **qMsgRead( )**.
- filenames are italic and lowercase. Example: *coders.h*.
- message names are italic, lowercase, are proceeded by a prefix and underscore. Example: *Player_MsgAdjSpeed.*
- Parameter and field names are boldface. Example: "the **timeout** parameter".

1.2. Key DM3 Architecture Concepts

For more information about these concepts, see the *DM3 Mediastream Architecture Overview* (part number 05-0813-001).

- **DM3** is an architecture on which a whole set of Dialogic products are built. The DM3 architecture is open, layered, and flexible, encompassing hardware as well as software components.
- A **DM3 resource** is a conceptual entity that provides a specific functionality to a host application.

  A resource contains a well defined interface or message set, which the host application utilizes when accessing the resource. Resource firmware consists of multiple components that run on top of the DM3 core platform software (which includes the platform-specific DM3 kernel and device driver).
- A **component** is the entities that comprise a DM3 resource. A component runs on a DM3 control processor or signal processor, depending on its function. Certain components handle configuration and management issues, while others process stream data.

To access the features of a resource, the host exchanges messages and stream data with certain components of that resource. During runtime, components inside a resource communicate (via messages) with other components of that resource, as well as with components of other resources.

- A DM3 **message** is a formatted block of data exchanged between the host and various entities on the DM3 platform, as well as between the DM3 entities themselves.

  The DM3 architecture implements different kinds of messages, based on the functionality of the message sender and recipient. Messages can initiate actions, handle configuration, affect operating states, and indicate that events have occurred. They can be sent synchronously or asynchronously.

- A **cluster** is a collection of DM3 component instances that share specific TDM timeslots on the network interface or the SCbus, and which therefore operate on the same mediastream data. The cluster concept in the DM3 architecture corresponds generally but not exactly to the concept of a “group” in S.100 or to a “channel” in conventional Dialogic architectural terminology. Component instances are bound to a particular cluster and its assigned timeslots in an allocation operation.

- A **port** is a logical entity that represents the point at which PCM data can flow into or out of a component instance or interface in a cluster. The port abstraction provides a high-level means of defining potential data flow paths within clusters and controlling the actual data flow using simple protocols. Ports are classified and designated in terms of data flow direction and the type of entity that provides the port.

2. Standard Component Interface Messages

The messages in the DM3 Standard Component Message Set are contained in the header file *stddefs.h.*

Not all components will support the complete set of Standard Component Messages. The documentation for each DM3 component or resource will list the standard messages that it supports.

NOTE: The Standard Component Message Set is one of three types of message sets used by DM3 components and component instances. The other two set types are:

- the standard DM3 Kernel Message Set. These messages are sent by a DM3 component to the kernel or by the kernel to a component or component instance that called a kernel function. For information about the result messages returned by the kernel to the component that called the function, see the *DM3 Kernel Software Reference.*
- resource-specific message sets. Each DM3 resource has component and instance-level messages specific to that resource only, as created and defined by the developer of the resource's component(s). For more information about resource-specific message sets, see the appropriate resource user's guide.

Each instance of a DM3 component uses three types of messages:

- **command messages** are messages sent to the appropriate component instance from the host application.
- **reply messages** are a component instance's response to command messages and are sent from the instance to the host application. If an error occurs during the execution of the command, an error message (*Std_MsgError*) will be sent instead of the reply message.
- **event messages** are asynchronous messages that may be sent by the component instance to the host application when they are enabled by that host application.

The message sets that the host uses to communicate with the TSP resource are:

- a **Standard Component Message Set,** used by all DM3 resources for accessing standard features such as parameter setting, asynchronous event enable/disable, and Run Time Control. This chapter lists the standard messages used by all DM3 resources.
- one or more **resource-specific message sets,** which is used to access features specific to the resource. For more information about these resource-specific sets, see the appropriate user's guide for the DM3 resource.

Table 1 lists the messages in the standard component message set. Each of these messages is described in this chapter.

Table 1. The Standard Component Message Set

Command Message	**Description**	**Reply Message (indicates success)**
Std_MsgAck	the command has been received but not yet executed	Not applicable
Std_MsgArmRTC	arms a component instance for an RTC action	*Std_MsgArmRTCCmplt*
Std_MsgArmxRTCs	arms a component instance for multiple RTC actions	*Std_MsgArmxRTCsCmplt*
Std_MsgCancelAllEvts	cancels the reporting of all events by the component instance	*Std_MsgCancelAllEvtsCmplt*

Command Message	**Description**	**Reply Message (indicates success)**
Std_MsgCancelEvt	cancels the reporting of a single event by the component instance	*Std_MsgCancelEvtCmplt*
Std_MsgCancelxEvts	disables the reporting of multiple events by the component instance	*Std_MsgCancelxEvtsCmplt*
Std_MsgComtest	establishes a communications path with a component or instance of the component	*Std_MsgComtestCmplt*
Std_MsgDetectEvt	configures a component instance to detect a specific event	*Std_MsgDetectEvtCmplt*
Std_MsgDetectxEvts	configures a component instance to detect multiple events	*Std_MsgDetectxEvtsCmplt*
Std_MsgDisarmRTC	disables an RTC action so that a component instance can no longer enact it	*Std_MsgDisarmRTCCmplt*

Command Message	**Description**	**Reply Message (indicates success)**
Std_MsgDisarmxRTCs	disables multiple RTC actions so that a component instance can no longer enact them	*Std_MsgDisarmxRTCsCmplt*
Std_MsgDisarmAllRTCs	disables all RTC actions so that a component instance will not perform any RTC actions	*Std_MsgDisarmAllRTCsCmplt*
Std_MsgError	indicates an error in executing a command message	Not applicable
Std_MsgEvtDetected	indicates an event has occurred	Not applicable
Std_MsgExecute	executes the function contained in the message	*Std_MsgExecuteCmplt*
Std_MsgExit	shuts down a component instance	*Std_MsgExitCmplt*
Std_MsgGetParm	requests a parameter value of a component instance	*Std_MsgGetParmCmplt*
Std_MsgGetxParms	requests multiple parameter values of component instance	*Std_MsgGetxParmsCmplt*
Std_MsgInit	initializes a component instance	*Std_MsgInitCmplt*

Command Message	**Description**	**Reply Message (indicates success)**
Std_MsgSetAllParmsDef	sets all parameters to their default values	*Std_MsgSetAllParmsDefCmplt*
Std_MsgSetParm	changes the value of a specific parameter	*Std_MsgSetParmCmplt*
Std_MsgSetParmDef	sets a specific parameter to its default value	*Std_MsgSetParmDefCmplt*
Std_MsgSetxParms	changes the value of multiple parameters	*Std_MsgSetxParmsCmplt*
Std_MsgSetxParmsDef	sets multiple parameters to their default values	*Std_MsgSetxParmsDefCmplt*

Std_MsgAck

Definition

The *Std_MsgAck* message is a reply message for any DM3 command message. It indicates that the command has been received but not yet executed.

Message Sender and Recipient

The component or component instance sends this message to the host.

Additional Information

This message may be used in addition to or in place of a command-specific reply message.

Message Contents

The body of this message contains no data fields.

Cautions

The *Std_MsgAck* message verifies that the designated component or component instance has received the command message; it does not indicate that a command has been started or successfully completed. The following response messages can be returned to the host after returning a *Std_MsgAck* message:

- the appropriate reply message for the command message indicated by *Std_MsgAck*. The reply message indicates the command message was successfully executed. The reply message usually ends in "*Cmplt*".
- a *Std_MsgError* message for the command message indicated by *Std_MsgAck*. The *Std_MsgError* message indicates a failure in the execution of the command message.

- *Std_MsgEvtDetected* messages for various events, which will announce successful and unsuccessful events triggered by the command message indicated by *Std_MsgAck*.

Errors

None.

Other Related Messages

All command messages that use this optional message as a response.

Std_MsgArmRTC

Definition

The *Std_MsgArmRTC* message is a command message that arms a component instance for an RTC action.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message arms a single RTC event that has been set up using a particular transaction ID from a particular source address. The source address and transaction ID in the header of this message are used by the receiver to uniquely identify this request. They are also used when cancelling the request. The transaction ID and **Label** combination must be unique in the system to avoid collision; at a minimum, the transaction ID should not equal 0.

Message Contents

The body of the *Std_MsgArmRTC* message contains two data fields in packed-byte format. The Std_MsgArmRTC_put macro inserts the fields into a message, from a data structure of type Std_MsgArmRTC_t, which contains the following elements:

Field	Data Type	Description
Label	UInt32	event label to expect in the *Std_MsgEvtDetected* message
Action	UInt32	action to take once this event has been received

Cautions

The sender must ensure that the transaction ID in the header of this message is the same as that which is used in the *Std_MsgDetectEvt* message when the condition to be detected is set up.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrLabelTrans	A Label and Transaction ID pair that is already in use was specified in an Arm RTC command message.
Std_ErrRTCAction	Invalid Run Time Control action was specified in an event handling command message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTCCmplt*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*

- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgArmRTCCmplt

Definition

The *Std_MsgArmRTCCmplt* message is a reply message for the *Std_MsgArmRTC* command. It confirms the component instance is armed for an RTC action.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgArmRTC*

Std_MsgArmxRTCs

Definition

The *Std_MsgArmxRTCs* message is a command message that arms a component instance for multiple RTC actions.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

The source address and transaction ID of this message are used by the receiver to uniquely identify this request. They are also used when cancelling the request. The transaction Id and Label combination must be unique in the system to avoid collision. At a minimum the transaction ID should not equal 0.

Message Contents

The body of the *Std_MsgArmxRTCs* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgArmxRTCs_put macro inserts the fields into a message, from a data structure of type Std_MsgArmxRTCs_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	the number of events to expect. This fixed data field specifies the number of **Label**/**Action** pairs in the variable part of the message body.

Each **Label/Action** pair can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for each of the two fields. For the initial call, the value of the offset variable must be Std_MsgArmxRTCs_varstart, which is the value to which the

Std_MsgArmxRTCs_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Label/Action** pair.

Field	**Data Type**	**Description**
Label	UInt32	the label of an event to expect in a *Std_MsgEvtDetected* message. Each **Label** is paired with a corresponding **Action**.
Action	UInt32	the action to take once the corresponding event has been received. Each **Action** has a specific corresponding **Label**.

Cautions

A sender of this message must ensure that the Transaction ID of this message is the same as that which is used in the *Std_MsgDetectEvt* message when each of the conditions to be detected is set up.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrLabelTrans	A Label and Transaction ID pair that is already in use was specified in an Arm RTC command message.
Std_ErrListEmpty	The list contains no elements.

ErrorCode Field Values	Description
Std_ErrRTCAction	Invalid Run Time Control action was specified in an event handling command message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCsCmplt*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgArmxRTCsCmplt

Definition

The *Std_MsgArmxRTCsCmplt* message is a reply message for the *Std_MsgArmxRTCs* command. It confirms the component instance is armed for multiple RTC actions.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgArmxRTCs*

Std_MsgCancelAllEvts

Definition

The *Std_MsgCancelAllEvts* message is a command message that cancels the reporting of all events by the component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message cancels the reporting of all events that have been set up with a particular transaction ID from a particular source address.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrNotEnabled	An event-cancelling message was received for an event that is not currently enabled.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvtsCmplt*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgCancelAllEvtsCmplt

Definition

The *Std_MsgCancelAllEvtsCmplt* message is a reply message for the *Std_MsgCancelAllEvts* command. It confirms the specified events will not be reported by the component instance.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgCancelAllEvts*

Std_MsgCancelEvt

Definition

The *Std_MsgCancelEvt* message is a command message that cancels the reporting of a single event by the component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message cancels the reporting of a single event that had been set up using a particular transaction ID from a particular source address.

Message Contents

The body of the *Std_MsgCancelEvt* message contains one data field in packed-byte format. The Std_MsgCancelEvt_put macro inserts the fields into a message, from a data structure of type Std_MsgCancelEvt_t, which contains the following elements:

Field	**Data Type**	**Description**
Type	UInt32	the event type to cancel.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtType	Invalid event type was specified in an event handling command message.
Std_ErrNotEnabled	An event-cancelling message was received for an event that is not currently enabled.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_DisarmxRTC*
- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvtCmplt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgEvtDetected*

Std_MsgCancelEvtCmplt

Definition

The *Std_MsgCancelEvtCmplt* message is a reply message for the *Std_MsgCancelEvt* command. It verifies the event will not be reported by the component instance.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgCancelEvt*

Std_MsgCancelxEvts

Definition

The *Std_MsgCancelxEvts* message is a command message that disables the reporting of multiple events by the component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

The events must have been previously enabled using the same transaction ID and source address as used for this command. If the Component Instance is able to cancel the events, it replies to the sender with a *Std_MsgCancelxEvtsCmplt* message.

Message Contents

The body of the *Std_MsgCancelxEvts* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgCancelxEvts_put macro inserts the fields into a message, from a data structure of type Std_MsgCancelxEvts_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	the number of events to disable. This single fixed data field identifies the number of **Type** fields contained in the variable part of the message body.

Each **Type** can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for the field. For the initial call, the value of the offset variable must be

Std_MsgCancelxEvts_varstart, which is the value to which the Std_MsgCancelxEvts_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Type**.

Field	Data Type	Description
Type	UInt32	the event type to be disable. There are **Count** of these **Type** fields in a variable data structure.

Cautions

If one of the event types in the list is invalid, then none of the events will be canceled.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtType	Invalid event type was specified in an event handling command message. The invalid type will be returned.
Std_ErrListEmpty	The list contains no elements.
Std_ErrNotEnabled	An event-cancelling message was received for an event that is not currently enabled.
Std_ErrSystem	System level error occurred while executing a command.

ErrorCode Field Values	**Description**
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvtsCmplt*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgCancelxEvtsCmplt

Definition

The *Std_MsgCancelxEvtsCmplt* message is a reply message for the *Std_MsgCancelxEvts* command. It verifies the events will not be reported by the component instance.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgCancelxEvts*

Std_MsgComtest

Definition

The *Std_MsgComtest* message is a command message that establishes a communications path with a component or instance of the component.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

Std_MsgComtest may be used as a primitive debug feature.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrSystem	System level error occurred while executing a command.

ErrorCode Field Values	Description
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgComtestCmplt*

Std_MsgComtestCmplt

Definition

The *Std_MsgComtestCmplt* message is a reply message for the *Std_MsgComtest* command. It verifies a communications path with a component or instance of the component.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgComtest*

Std_MsgDetectEvt

Definition

The *Std_MsgDetectEvt* message is a command message that configures a component instance to detect a specific event.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This may be used for simply expressing interest in asynchronous events or for Run Time Control. The source address and transaction ID of this message are used by the receiver to uniquely identify this request. They are also used when cancelling the request. A transaction ID of 0 is used by convention to simply enable asynchronous events that are not used for Run Time Control.

If the receiver of this message is able to report the specified event, it replies to the sender with a *Std_MsgDetectEvtCmplt* message. Note that this message does not indicate that the event has been detected, but rather that the next occurrence of the event will be reported as requested.

Message Contents

The body of the *Std_MsgDetectEvt* message contains three data fields in packed-byte format. The Std_MsgDetectEvt_put macro inserts the fields into a message, from a data structure of type Std_MsgEvtDetected_t, which contains the following elements:

Field	**Data Type**	**Description**
RetAddr	QcompDesc	the address to send the *Std_MsgEvtDetected* message to.

Type	UInt32	the event type to be enabled.
Label	UInt32	an event label to be included in the *Std_MsgEvtDetected* message that will be understood by the one receiving that message.

Cautions

A receiver of this message must always use the transaction ID of this message as the transaction ID of the *Std_MsgEvtDetected* message.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtType	Invalid event type was specified in an event handling command message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*

- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvtCmplt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgDetectEvtCmplt

Definition

The *Std_MsgDetectEvtCmplt* message is a reply message for the *Std_MsgDetectEvt* command. It verifies the component instance can detect a specific event.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgDetectEvt*

Std_MsgDetectxEvts

Definition

The *Std_MsgDetectxEvts* message is a command message that configures a component instance to detect multiple events.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

Event reporting may be used for simply expressing interest in asynchronous events or for Run Time Control. The source address and transaction ID of this message are used by the receiver to uniquely identify this request. They are also used when cancelling the request. The events are represented as tuples of **Type** and **Label**. A transaction ID of 0 is used by convention to simply enable asynchronous events that are not used for Run Time Control. If the receiver of this message is able to enable the specified event types, it will reply to the sender with a *Std_MsgDetectxEvtsCmplt* message.

Message Contents

The body of the *Std_MsgDetectxEvts* message contains a variable-size payload that includes two fixed data fields, followed by a variable-length list of data items.

The Std_MsgDetectxEvts_put macro inserts the fields into a message, from a data structure of type Std_MsgDetectxEvts_t, which contains the following elements:

Field	**Data Type**	**Description**
RetAddr	QCompDesc	the address to send the *Std_MsgEvtDetected* messages to.

Field	Data Type	Description
Count	UInt32	the number of events to report. This fixed data field specifies the number of **Type**/**Label** field pairs that are contained in the variable struture.

Each **Type**/**Label** pair can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for each of the two fields. For the initial call, the value of the offset variable must be Std_MsgDetectxEvts_varstart, which is the value to which the Std_MsgDetectxEvts_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Type**/**Label** pair.

Field	Data Type	Description
Type	UInt32	an event type to be enabled. Each **Type** has an associated **Label**.
Label	UInt32	the event label to be returned in the *Std_MsgEvtDetected* message corresponding to each event **Type**. The label must be understood by the one receiving that message.

Cautions

The receiver of this message must always return the transaction ID in the header of this message as the transaction ID of all specified *Std_MsgEvtDetected* messages.

If one of the event types in the list is invalid, then none of the events will be enabled.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message.

The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtType	Invalid event type was specified in an event handling command message. The invalid type will be returned.
Std_ErrListEmpty	The list contains no elements.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvtsCmplt*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgDetectxEvtsCmplt

Definition

The *Std_MsgDetectxEvtsCmplt* message is a reply message for the *Std_MsgDetectxEvts* command. It verifies the component instance can detect multiple events.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgDetectxEvts*

Std_MsgDisarmAllRTCs

Definition

The *Std_MsgDisarmAllRTCs* message is a command message that disables all RTC actions so that a component instance will not perform any RTC actions.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message disarms all RTC actions that have been set up using a particular transaction ID from a particular source address.

Message Contents

The body of this message contains no data fields.

Cautions

A sender of this message must ensure that the Transaction ID of this message is the same as that which was used in the *Std_MsgDetectEvt* message when the conditions to be detected were set up.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvents*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCsCmplt*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgDisarmAllRTCsCmplt

Definition

The *Std_MsgDisarmAllRTCsCmplt* message is a reply message for the *Std_MsgDisarmAllRTCs* command. It confirms the component instance will not perform any RTC actions.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgDisarmAllRTCs*

Std_MsgDisarmRTC

Definition

The *Std_MsgDisarmRTC* message is a command message that disables an RTC action so that a component instance can no longer enact it.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message disarms a single RTC event that has been set up using a particular transaction ID from a particular source address

Message Contents

The body of the *Std_MsgDisarmRTC* message contains one data field in packed-byte format. The Std_MsgDisarmRTC_put macro inserts the fields into a message, from a data structure of type Std_MsgDisarmRTC_t, which contains the following elements:

Field	**Data Type**	**Description**
Label	UInt32	label of event to disarm.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtLabel	Invalid event label was specified in an event handling command message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTCCmplt*
- *Std_MsgDisarmxRTCs*
- *Std_MsgEvtDetected*

Std_MsgDisarmRTCCmplt

Definition

The *Std_MsgDisarmRTCCmplt* message is a reply message for the *Std_MsgDisarmRTC* command. It confirms the component instance will not perform an RTC action.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgDisarmRTC*

Std_MsgDisarmxRTCs

Definition

The *Std_MsgDisarmxRTCs* message is a command message that disables multiple RTC actions so that a component instance can no longer enact them.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message disarms multiple RTC actions that have been set up using a particular transaction ID from a particular source address.

Message Contents

The body of the *Std_MsgDisarmxRTCs* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgDisarmxRTCs_put macro inserts the fields into a message, from a data structure of type Std_MsgDisarmxRTCs_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	the number of RTC actions to disarm. This fixed data field specifies the number of **Label** fields that are contained in the variable portion of the message body.

Each **Label** can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for the field. For the initial call, the value of the offset variable must be Std_MsgDisarmxRTCs_varstart, which is the value to which the

Std_MsgDisarmxRTCs_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Label**.

Field	Data Type	Description
Label	UInt32	the label of an event disarm. The message will contain **Count** of these labels.

Cautions

If one of the Labels in the list is invalid then none of the RTC events will be disarmed.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrEvtLabel	Invalid event label was specified in an event handling command message.
Std_ErrListEmpty	The list contains no elements.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCsCmplt*
- *Std_MsgEvtDetected*

Std_MsgDisarmxRTCsCmplt

Definition

The *Std_MsgDisarmxRTCsCmplt* message is a reply message the *Std_MsgDisarmxRTCs* command. It confirms the component instance will not perform the RTC actions.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgDisarmxRTCs*

Std_MsgError

Definition

The *Std_MsgError* message is an error message that indicates an error in executing a command message.

Message Sender and Recipient

The component or component instance sends this message to the host.

Additional Information

This message indicates failure of either an incoming command or command execution. Components and component instances should also be able to receive this message as a valid reply to any command it may send to other components or component instances.

Message Contents

The body of the *Std_MsgError* message contains three data fields in packed-byte format. The Std_MsgError_get macro extracts the fields from a message, into a data structure of type Std_MsgError_t, which contains the following elements:

Field	**Data Type**	**Description**
ErrorMsg	UInt32	message type where error was detected. If the error was an asynchronous error not due to any specific command message, this field should be set to 0.
ErrorCode	UInt32	reason for error. Standard error codes are listed in an appendix to this document. Component-specific error codes are defined in the individual component interface specifications.

Field	Data Type	Description
Data[4]	UInt32	any extra data to report as qualification of the error. The use of these extra data fields is specific to particular error codes and the receiver of the message must be aware of the usage.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- All command messages that can result in an error.

Std_MsgEvtDetected

Definition

The *Std_MsgEvtDetected* message is an event-reporting message that indicates an event has occurred.

Message Sender and Recipient

The component instance sends this message to the host.

Additional Information

This event reporting was configured by either the *Std_MsgDetectEvt* message or the *Std_MsgDetectxEvts* message.

Message Contents

The body of the *Std_MsgEvtDetected* message contains two data fields in packed-byte format. The Std_MsgDetectEvt_get macro extracts the fields from a message, into a data structure of type Std_MsgDetectEvt_t, which contains the following elements:

Field	**Data Type**	**Description**
Label	UInt32	the event label that was specified in the *Std_MsgDetectEvt* command that enabled the event.
Data[5]	UInt32	any extra data associated with this event. The receiver of this data must be able to interpret it based on the type of event.

Cautions

A sender of this message must ensure that the transaction ID in the header of this message is the same as that which was specified in the *Std_MsgDetectEvt* message.

Events with extra data should NOT be used for Run Time Control.

A receiver must match both the **Label** AND the transaction ID against its list of events to expect. If either is invalid then this condition should be flagged somehow and the message should be ignored.

Errors

None.

Other Related Messages

- *Std_MsgArmRTC*
- *Std_MsgArmxRTCs*
- *Std_MsgCancelAllEvts*
- *Std_MsgCancelEvt*
- *Std_MsgCancelxEvts*
- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgDisarmAllRTCs*
- *Std_MsgDisarmRTC*
- *Std_MsgDisarmxRTCs*

Std_MsgExecute

Definition

The *Std_MsgExecute* message is a command message that executes the function contained in the message.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

How this message is interpreted is completely proprietary to the component writer using it. If the Component or Instance is able to execute the function, it replies to the sender with a *Std_MsgExecuteCmplt* message.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgExecuteCmplt*

Std_MsgExecuteCmplt

Definition

The *Std_MsgExecuteCmplt* message is a reply message for the *Std_MsgExecute* message. It indicates the component instance has executed the encapsulated function.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgExecute*

Std_MsgExit

Definition

The *Std_MsgExit* message is a command message that shuts down a component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This command should be accepted at all times.

The component instance will shut down in whatever proprietary way is appropriate. This command is usually sent at shutdown time when a DM3 Load Module is about to be unloaded. The module exit function will usually send this message to all of the Components in that module. The Components may then forward this to the Instances supported by that component.

Message Contents

The body of this message contains no data fields.

Cautions

When receiving this message the Component or Instance should gracefully shut itself down in its own proprietary way without impacting the rest of the system.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrInstInUse	Instance is in use.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnexpectedMsg	Unexpected message was received. This unexpected message indicates one of the following conditions: • the message is not part of the standard component message set or the resource-specific message set • the message is part of the standard component message set, but the receiving component or instance is not configured to recognize it • the receiving component or instance can recognize the message but is not in the proper state to execute it.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgExitCmplt*
- *Std_MsgInit*

Std_MsgExitCmplt

Definition

The *Std_MsgExitCmplt* message is a reply message for the *Std_MsgExit* command. It verifies the component instance has shut down.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgExit*

Std_MsgGetParm

Definition

The *Std_MsgGetParm* message is a command message that requests a parameter value of a component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of the *Std_MsgGetParm* message contains one data field in packed-byte format. The Std_MsgGetParm_put macro inserts the fields into a message, from a data structure of type Std_MsgGetParm_t, which contains the following elements:

Field	**Data Type**	**Description**
Num	Qparm	parameter number of parameter to read.

Cautions

The maximum size for parameter values using this command is 32 bits.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages.
Std_ErrParmWriteOnly	A parameter-writing message was received for one or more write-only parameters.
Std_ErrSystem	System level error occurred while executing a command.

Other Related Messages

- *Std_MsgGetParmCmplt*
- *Std_MsgGetxParms*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParm*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParm*
- *Std_MsgSetxParmsDef*

Std_MsgGetParmCmplt

Definition

The *Std_MsgGetParmCmplt* message is a reply message for the *Std_MsgGetParm* command. It contains a requested parameter value.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of the *Std_MsgGetParmCmplt* message contains a variable-size payload that includes three fixed data fields, followed by a variable-length list of data items.

The Std_MsgGetParmCmplt_get macro extracts the fixed fields from a message, into a data structure of type Std_MsgGetParmCmplt_t, which contains the following elements:

Field	Data Type	Description
Num	Qparm	parameter number of parameter that was read
Val	UInt32	parameter value read
TextSize	UInt32	size of optional text string

Each **Text** value can be retrieved from the message body with a single call to the **qMsgVarFieldGet( )** function, specifying the field definition and destination variable for the field. For the initial call, the value of the offset variable must be Std_MsgGetParmCmplt_varstart, which is the value to which the Std_MsgGetParmCmplt_get macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Text**.

Field	Data Type	Description
Text	Char	optional text string (ASCII null-delimited) associated with the version parameter. If there is no text, **TextSize** will be zero.

Cautions

The maximum size for parameter values in this message is 32 bits.

Errors

None.

Other Related Messages

- *Std_MsgGetParm*

Std_MsgGetxParms

The *Std_MsgGetxParms* message is a command message that requests multiple parameter values of component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of the *Std_MsgGetxParms* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgGetxParms_put macro inserts the fields into a message, from a data structure of type Std_MsgGetxParms_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	number of parameters to read. This fixed data filed specifies the number of **Num/Val** pairs that are contained in the variable part of the message body.

Each **Num/Val** pair can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for each of the two fields. For the initial call, the value of the offset variable must be Std_MsgGetxParms_varstart, which is the value to which the Std_MsgGetxParms_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Num/Val** pair.

Field	**Data Type**	**Description**
Num	Qparm	the parameter number of a parameter to read. The message contains **Count** parameter number fields, each with a dummy **Val** field.

Field	Data Type	Description
Val	UInt32	dummy placeholder for parameter value when read. There is one **Val** for each of the **Count Num** fields.

Cautions

The maximum size for parameter values using this command is 32 bits.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrListEmpty	The list contains no elements.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages.
Std_ErrParmWriteOnly	A parameter-writing message was received for one or more write-only parameters.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgGetParm*
- *Std_MsgGetxParmsCmplt*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParm*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParms*
- *Std_MsgSetxParmsDef*

Std_MsgGetxParmsCmplt

Definition

The *Std_MsgGetxParmsCmplt* message is a reply message for the *Std_MsgGetxParms* command. It contains requested parameter values.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of the *Std_MsgGetxParmsCmplt* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgGetxParmsCmplt_get macro extracts the fixed fields from a message, into a data structure of type Std_MsgGetxParmsCmplt_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	number of parameters read. This fixed data field specifies the number of **Num**/**Val** pairs that are contained in the variable part of the message body.

Each **Num**/**Val** pair can be retrieved from the message body with a single call to the **qMsgVarFieldGet( )** function, specifying the field definition and destination variable for each of the two fields. For the initial call, the value of the offset variable must be Std_MsgGetxParmsCmplt_varstart, which is the value to which the Std_MsgGetxParmsCmplt_get macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Num**/**Val** pair.

Field	**Data Type**	**Description**
Num	QParm	parameter number of the parameter that was read. For each **Num**, there is a corresponding **Val**.
Val	UInt32	the parameter value for the parameter specified by **Num**. There is a **Val** for each **Num** in the list.

Cautions

The maximum size for parameter values using this command is 32 bits.

Errors

None.

Other Related Messages

- *Std_MsgGetxParms*

Std_MsgInit

Definition

The *Std_MsgInit* message is a command message that initializes a component instance.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

This message initializes component instance in whatever proprietary way is appropriate.

For Components, this command is sent by the host download utility at startup time. The Component will usually create its Instances when it receives this message. This command message may optionally be sent to an Instance, usually by its parent Component, to initialize it.

Message Contents

The body of this message contains no data fields.

Cautions

This command should only be sent once, at initialization time.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message.

The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrInstCreate	Error creating instances.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnexpectedMsg	Unexpected message was received. This unexpected message indicates one of the following conditions: • the message is not part of the standard component message set or the resource-specific message set • the message is part of the standard component message set, but the receiving component or instance is not configured to recognize it • the receiving component or instance can recognize the message but is not in the proper state to execute it.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgExit*
- *Std_MsgInitCmplt*

Std_MsgInitCmplt

Definition

The *Std_MsgInitCmplt* message is a reply message for the *Std_MsgInit* command. It verifies the component instance is initialized.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgInit*

Std_MsgSetAllParmsDef

Definition

The *Std_MsgSetAllParmsDef* message is a command message that sets all parameters to their default values.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of this message contains no data fields.

Cautions

Setting parameters at run time can have adverse affects if a Component or Instance is actively using these parameters.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgGetParm*
- *Std_MsgGetxParms*
- *Std_MsgSetAllParmsDefCmplt*
- *Std_MsgSetParm*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParms*
- *Std_MsgSetxParmsDef*

Std_MsgSetAllParmsDefCmplt

Definition

The *Std_MsgSetAllParmsDefCmplt* message is a reply message for the *Std_MsgSetAllParmsDef* command. It verifies all parameters are set to their default value.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgSetAllParmsDef*

Std_MsgSetParm

Definition

The *Std_MsgSetParm* message is a command message that changes the value of a specific parameter.

Message Sender and Recipient

The host sends this message to the component instance.

Additional Information

The parameter is defined as the tuple of Num and Val. If the Component or Instance successfully sets the specified parameter, it replies to the sender of the command with a *Std_MsgSetPArmCmplt* message.

Message Contents

The body of the *Std_MsgSetParm* message contains two data fields in packed-byte format. The Std_MsgSetParm_put macro inserts the fields into a message, from a data structure of type Std_MsgSetParm_t, which contains the following elements:

Field	**Data Type**	**Description**
Num	Qparm	parameter number of the parameter to set.
Val	UInt32	parameter value to set for parameter.

Cautions

Setting parameters at run time can have adverse affects if a Component or Instance is actively using these parameters. The maximum size for parameter values using this command is 32 bits.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages.
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrParmValue	Invalid parameter value was specified in a parameter service command message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgGetParm*
- *Std_MsgGetxParms*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParmCmplt*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParms*
- *Std_MsgSetxParmsDef*

Std_MsgSetParmCmplt

Definition

The *Std_MsgSetParmCmplt* message is a reply message for the *Std_MsgSetParm* command. It verifies the parameter is set.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgSetParm*

Std_MsgSetParmDef

Definition

The *Std_MsgSetParmDef* message is a command message that sets a specific parameter to its default value.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of the *Std_MsgSetParmDef* message contains one data field in packed-byte format. The Std_MsgSetParmDef_put macro inserts the fields into a message, from a data structure of type Std_MsgSetParmDef_t, which contains the following elements:

Field	Data Type	Description
Num	Qparm	parameter number of the parameter to reset.

Cautions

Setting parameters at run time can have adverse affects if a Component or Instance is actively using these parameters.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	**Description**
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages.
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_GetxParms*
- *Std_MsgGetParm*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParm*
- *Std_MsgSetParmDefCmplt*
- *Std_MsgSetxParms*
- *Std_MsgSetxParmsDef*

Std_MsgSetParmDefCmplt

Definition

The *Std_MsgSetParmDefCmplt* message is a reply message for the *Std_MsgSetParmDef* command. It verifies the parameter is set to its default value.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgSetParmDef*

Std_MsgSetxParms

Definition

The *Std_MsgSetxParms* message is a command message that changes the value of multiple parameters.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of the *Std_MsgSetxParms* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgSetxParms_put macro inserts the fields into a message, from a data structure of type Std_MsgSetxParms_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	the number of parameters to set to defaults. This fixed data field specifies the number of **Num/Val** pairs in the variable part of the message body.

Each **Num/Val** pair can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for each of the two fields. For the initial call, the value of the offset variable must be Std_MsgSetxParms_varstart, which is the value to which the Std_MsgSetxParms_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Num**/**Val** pair.

Field	Data Type	Description
Num	QParm	the parameter number of a parameter to set. There are **Count** parameters specified in the message, each with a corresponding **Val**.
Val	UInt32	the parameter value to set for parameter specified in **Num**. Each **Num** has its own **Val.**

Cautions

Setting parameters at run time can have adverse affects if a Component or Instance is actively using these parameters.

The maximum size for parameter values using this command is 32 bits.

If multiple parameters are set using this command and there is an error in any of the parameter specifications then NO parameters should be updated.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrListEmpty	The list contains no elements.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages. The invalid parameter number will be returned in the *Std_MsgError* message.

ErrorCode Field Values	**Description**
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrParmValue	Invalid parameter value was specified in a parameter service command message. The invalid parameter number and value will be returned in the *Std_MsgError* message.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgGetParm*
- *Std_MsgGetxParms*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParm*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParmsCmplt*
- *Std_MsgSetxParmsDef*

Std_MsgSetxParmsCmplt

Definition

The *Std_MsgSetxParmsCmplt* message is a reply message for the *Std_MsgSetxParms* command. It verifies the parameters are set.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgSetxParms*

Std_MsgSetxParmsDef

Definition

The *Std_MsgSetxParmsDef* message is a command message that sets multiple parameters to their default values.

Message Sender and Recipient

The host sends this message to the component instance.

Message Contents

The body of the *Std_MsgSetxParmsDef* message contains a variable-size payload that includes one fixed data field, followed by a variable-length list of data items.

The Std_MsgSetxParmsDef_put macro inserts the fields into a message, from a data structure of type Std_MsgSetxParmsDef_t, which contains the following elements:

Field	**Data Type**	**Description**
Count	UInt32	the number of parameters to set to defaults. This fixed data field specifies the number of **Num** fields in the variable part of the message body.

Each **Num** can be filled into the message body with a single call to the **qMsgVarFieldPut( )** function, specifying the field definition and source variable for the field. For the initial call, the value of the offset variable must be Std_MsgSetxParmsDef_varstart, which is the value to which the Std_MsgSetxParmsDef_put macro automatically updated its offset variable. The offset variable is automatically updated with each successive function call, to reflect the start of a new **Num**.

Field	Data Type	Description
Num	QParm	the parameter number of a parameter to reset. The message contains **Count** parameter numbers.

Cautions

Setting parameters at run time can have adverse affects if a Component or Instance is actively using these parameters.

Errors

If an error occurs in the execution of the requested command, a *Std_MsgError* message will be returned to the requester instead of the successful reply message. The **ErrorCode** field in the body of this error message may contain one of the following values defined for this command:

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received.
Std_ErrListEmpty	The list contains no elements.
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages.
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrSystem	System level error occurred while executing a command.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Other Related Messages

- *Std_MsgGetParm*

- *Std_MsgGetxParms*
- *Std_MsgSetAllParmsDef*
- *Std_MsgSetParm*
- *Std_MsgSetParmDef*
- *Std_MsgSetxParms*
- *Std_MsgSetxParmsDefCmplt*

Std_MsgSetxParmsDefCmplt

Definition

The *Std_MsgSetxParmsDefCmplt* message is a reply message for the *Std_MsgSetxParmsDef* command. It verifies the parameters are set to their default values.

Message Sender and Recipient

The component instance sends this message to the host.

Message Contents

The body of this message contains no data fields.

Cautions

None.

Errors

None.

Other Related Messages

- *Std_MsgSetxParmsDef*

3. Event Notification

When a DM3 component or component instance detects an event, it uses the *Std_MsgEvtDetected* message to broadcast this event occurrence to certain DM3 entities—specifically, to those clients (a component instance or the host application) registered for notification about this type of event.

DM3 component instances use event notification to keep each other informed of events that have occurred. This is important for the following reasons:

- tracking and analyzing the performance of the different DM3 component instances. Event messages can show the incremental progress (or failure) or a command as it executes.
- to trigger RTC actions when specific events occur. An event message about a particular event can trigger an RTC action by a client registered for notification about the event.

Each DM3 resource uses the following standard messages for the event-notification process:

- *Std_MsgDetectEvt*
- *Std_MsgDetectxEvts*
- *Std_MsgEvtDetected*

3.1. About Event Notification and RTC Actions

After receiving a *Std_MsgEvtDetected* message indicating an event, a client can take an appropriate action. To do so, the client must first be set up to perform the RTC action.

For more information about RTC actions, see the appropriate appendix.

3.2. Event Registration

In order for event reporting to occur, the following component instances must be involved in a registration process for each specific event type:

- the component instance that will experience the event, so that it will notify the appropriate clients (component instances registered for notification about a particular event).
- the component instance that needs event notification about a particular event

For more information about

- each of these messages, see the document *DM3 Standard Component Interface Messages* (part number 05-1040-001)
- event notification and event registration, see the *DM3 Mediastream Architecture Overview* (part number 05-0813-001).

3.3. DM3 Resource Events

Each DM3 resource is configured to send notification to clients registered for this type of event notification. For more information, see the appropriate DM3 resource user's guide.

Appendix A

Parameters

Table 2 lists the parameters supported by all DM3 components and component instances in a DM3 firmware system.

Parameters control the behavior of a component and its instances. Parameter access is classified as:

- read only (R)
- write only (W)
- both read and write (R/W)

The parameter level is classified as:

- component-level (C), which have the Dialogic factory settings as their defaults when started
- instance level (I), which have the component-level parameter values as their default upon allocation
- both component and instance level (C/I)

All components and their instances support the *Std_MsgSetParm* and *Std_MsgGetParm* messages to allow their parameters to be set and read, respectively. Parameters may be set or read while a component or instance is active or idle.

All these standard parameters are of the UInt32 data type.

Table 2. Standard Parameter Definitions

Parameter Name	Access	Level	Description
Std_ParmVersion	R	C	The version number of the Component. For Dialogic Components, the format of the version number is in Hex, e.g. Version 3.60 Alpha 4.15 is represented as 0x23600415.
Std_ParmVendor	R	C	The vendor ID of the Component.
Std_ParmMaxInst	R	C	The maximum number of Instances supported by this Component
Std_ParmInstNum	R/W	C	The number of Instances to create or already created. This parameter can be set at any time, but it only takes effect when the instance is initialized (with Std_MsgInit).
Std_ParmStartInst	R/W	C	The starting Instance number

Appendix B
Error Codes

If an error occurs when a component instance tries to execute a command, a *Std_MsgError* message will be sent to the client (the sender of the command message), alerting it to the error. This *Std_MsgError* message is sent in lieu of any reply message normally returned when the command goes through successfully.

Each component has its own set of valid error types. Errors that may be generated by components and component instances of DM3 resources are discussed in this appendix.

How Error Notification Works

The following process explains how error notification works:

1. A component or component instance sends a command message to another component or component instance.
2. An error occurs in the execution of this command.
3. The entity that failed the command execution sends a *Std_MsgError* message to the client (the source of the command message).

For more information about

- the *Std_MsgError* message, see the *DM3 Standard Component Interface Messages* (part number 05-1040-001)
- error notification, see the *DM3 Mediastream Architecture Overview* (05-0813-001).

Standard Component Error Codes

Table 3 lists the error types potentially generated by all DM3 components and component instances. These codes are returned in the **ErrorCode** field of the *Std_MsgError* message. The use of the additional data portion of the *Std_MsgError* message body (the **Data[4]** field) varies according to the error code, and this usage is noted in the Description column of the table. Note that the message type of the command message that produced the error is always contained in the **ErrorMsg** field of the *Std_MsgError* message.

Table 3. Error Codes of DM3 Components and Component Instances

ErrorCode Field Values	Description
Std_ErrBusy	Component or Instance was busy executing a previous command when another command was received. The additional data field is not used for this error code.
Std_ErrEvtLabel	Invalid event label was specified in an event handling command message. For this error code, the additional data portion of the *Std_MsgError* message body is used to return the invalid event label.
Std_ErrEvtType	Invalid event type was specified in an event handling command message. For this error code, the additional data portion of the *Std_MsgError* message body is used to return the invalid event type.
Std_ErrInstCreate	Error creating instances. The additional data field is not used for this error code.
Std_ErrInstInUse	Instance is in use. The additional data field is not used for this error code.
Std_ErrLabelTrans	A Label and Transaction ID pair that is already in use was specified in an Arm RTC command message. Sender should change one or both and resend message. The additional data field is not used for this error code.
Std_ErrListEmpty	The list contains no elements.
Std_ErrNotEnabled	An event-cancelling message was received for an event that is not currently enabled.

ErrorCode Field Values	**Description**
Std_ErrParmNum	Invalid parameter number was specified in one of the parameter service messages. For this error code, the additional data portion of the *Std_MsgError* message body is used to return the invalid parameter number and the corresponding parameter value.
Std_ErrParmReadOnly	A parameter-setting message was received for one or more read-only parameters.
Std_ErrParmValue	Invalid parameter value was specified in a parameter service command message. For this error code, the additional data portion of the *Std_MsgError* message body is used to return the parameter number of the parameter with the invalid value and the invalid value itself.
Std_ErrParmWriteOnly	A parameter-writing message was received for one or more write-only parameters.
Std_ErrRTCAction	Invalid Run Time Control action was specified in an event handling command message. For this error code, the additional data portion of the *Std_MsgError* message body are used to return the event label associated with the invalid action and the invalid action itself.
Std_ErrSystem	System level error occurred while executing a command. The additional data field is not used for this error code.

ErrorCode Field Values	Description
Std_ErrUnexpectedMsg	Unexpected message was received. This unexpected message indicates one of the following conditions: • the message is not part of the standard component message set or the resource-specific message set • the message is part of the standard component message set, but the receiving component or instance is not configured to recognize it • the receiving component or instance can recognize the message but is not in the proper state to execute it.
Std_ErrUnsupportedMsg	Unsupported message was received. This unsupported message is part of the standard component message set.

Index

A

Architecture, DM3
    definition, 2

C

Component
    definition, 2

Components
    and error notification, 95
    and the Std_MsgError message, 95

Conventions, documentation, 2

D

DM3 architecture
    overview, 1

Documentation conventions, 2

E

Event notification
    and Std_MsgDetectEvt message, 91
    and Std_MsgDetectxEvts message, 91
    and the Std_MsgEvtDetected message, 91
    the process, 91

I

Introduction to DM3 architecture
    definition, 2

M

Message
    definition, 3

R

Resource
    definition, 2

Resource-specific messages
    overview, 1

Run Time Control (RTC)
    and Standard Component Message Set, 6

S

Std_MsgDetectEvt message, 91

Std_MsgDetectxEvts message, 91

Std_MsgEvtDetected message, 91

Std_MsgEvtDetected message, 91

DOCUMENTATION FEEDBACK FORM

Document Title:	DM3 Standard Component Interface Messages	
Publication Date:	March, 1998	Part Number: 05-1040-001-03

1. Please rate this document in the following areas:

	Excellent	Good	Adequate	Fair	Poor	N/A
Accuracy	❑	❑	❑	❑	❑	❑
Clarity	❑	❑	❑	❑	❑	❑
Ease of Use	❑	❑	❑	❑	❑	❑
Relevance to Job	❑	❑	❑	❑	❑	❑
Code Examples	❑	❑	❑	❑	❑	❑
Organization	❑	❑	❑	❑	❑	❑
Completeness	❑	❑	❑	❑	❑	❑
Figures/Illustrations	❑	❑	❑	❑	❑	❑
Appearance	❑	❑	❑	❑	❑	❑
Overall Satisfaction	❑	❑	❑	❑	❑	❑

2. How can we improve this document?

❑ Improve the index	❑ Add more step-by-step procedures and tutorials
❑ Improve the organization	❑ Make it more concise
❑ Improve overviews and introductions	❑ Add more detail
❑ Include more illustrations and figures	❑ Add more/better code examples
❑ Add more/better quick reference aids	❑ Make it less technical
❑ Add more troubleshooting information	❑ Make it more technical

3. Please include any other comments on an additional sheet.

4. FAX this form to **DIALOGIC DOCUMENTATION MANAGER** at **(973) 993-5916.**

NAME: ______________________________ **COMPANY:** __________________________

PHONE: _____________________________ **ADDRESS:** __________________________

NOTES

NOTES

NOTES