Internet Protocol (IP) Telephony Reference Guide

05-1043-001

COPYRIGHT NOTICE

All contents of this document are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation. Every effort is made to ensure the accuracy of this information. However, due to ongoing product improvements and revisions, Dialogic Corporation cannot guarantee the accuracy of this material, nor can it accept responsibility for errors or omissions. No warranties of any nature are extended by the information contained in these copyrighted materials. Use or implementation of any one of the concepts, applications, or ideas described this document or on Web pages maintained by Dialogic-may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not condone or encourage such infringement. Dialogic makes no warranty with respect to such infringement, nor does Dialogic waive any of its own intellectual property rights which may cover systems implementing one or more of the ideas contained herein. Procurement of appropriate intellectual property rights and licenses is solely the responsibility of the system implementer. The software referred to in this document is provided under a Software License Agreement. Refer to the Software License Agreement for complete details governing the use of the software.

Publication Date: April, 1998

Part Number: 05-1043-001

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

List of Tables

1. Introduction

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.

For a detailed description of the DM3 architecture, see the *DM3 Mediastream Architecture Overview*.

2. TSP/NetTSP Resource Messages

Each instance of the TSC and NetTSC 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.

  Table 1 lists the standard messages used by instances of the TSC and NetTSC components.

  - For a description of how this resource uses the *Std_MsgError* message, see *Appendix A*.
  - For a description of how this resource handles event reporting and RTC actions by using the *Std_MsgDetectEvt* and *Std_MsgEvtDetected* messages, see *Appendix B*.

Table 1. Standard Component Messages Used by the TSP and NetTSP Resources

Command Message	**Reply Message (indicates success)**	**Utilized by Which Component(s)**
Std_MsgAck	Not applicable	Generic TSC only
Std_MsgArmRTC	*Std_MsgArmRTCCmplt*	Generic TSC only
Std_MsgArmxRTCs	*Std_MsgArmxRTCCmplt*	Generic TSC only
Std_MsgCancelAllEvts	*Std_MsgCancelAllEvtsCmplt*	NetTSC
Std_MsgCancelEvt	*Std_MsgCancelEvtCmplt*	Generic TSC, NetTSC
Std_MsgCancelxEvts	*Std_MsgCancelxEvtsCmplt*	Generic TSC, NetTSC
Std_MsgComtest	*Std_MsgComtestCmplt*	Generic TSC, NetTSC
Std_MsgDetectEvt	*Std_MsgDetectEvtCmplt*	Generic TSC, NetTSC
Std_MsgDetectxEvts	*Std_MsgDetectxEvtsCmplt*	Generic TSC, NetTSC
Std_MsgDisarmAllRTCs	*Std_MsgDisarmAllRTCsCmplt*	Generic TSC only
Std_MsgDisarmRTC	*Std_MsgDisarmRTCCmplt*	Generic TSC only

Command Message	**Reply Message (indicates success)**	**Utilized by Which Component(s)**
Std_MsgDisarmxRTCs	*Std_MsgDisarmxRTCsCmplt*	Generic TSC only
Std_MsgError	Not applicable	Generic TSC, NetTSC
Std_MsgEvtDetected	Not applicable	Generic TSC, NetTSC
Std_MsgExecute	*Std_MsgExecuteCmplt*	Generic TSC only
Std_MsgExit	*Std_MsgExitCmplt*	Generic TSC, NetTSC
Std_MsgGetParm	*Std_MsgGetParmCmplt*	Generic TSC, NetTSC
Std_MsgGetxParms	*Std_MsgGetxParmsCmplt*	Generic TSC only
Std_MsgInit	*Std_MsgInitCmplt*	Generic TSC, NetTSC
Std_MsgSetAllParmsDef	*Std_MsgSetAllParmsDefCmplt*	NetTSC
Std_MsgSetParm	*Std_MsgSetParmCmplt*	Generic TSC, NetTSC
Std_MsgSetxParms	*Std_MsgSetxParmsCmplt*	Generic TSC only
Std_MsgSetParmDef	*Std_MsgSetParmDefCmplt*	Generic TSC, NetTSC
Std_MsgSetxParmsDef	*Std_MsgSetxParmsDefCmplt*	TBD

Table 2. The TSC Component Message Set

Command Message	Reply Message (indicates success)	Used by Which TSC Component
TSC_MsgAcceptCall	*Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgAnswerCall	*Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgBlindTransferCall	Not applicable	Generic TSC only
TSC_MsgCancelComplete	Not applicable	Generic TSC only
TSC_MsgCancelTransfer	Not applicable	Generic TSC only
TSC_MsgCompleteCall	*TSC_MsgCompleteCallCmplt*	Generic TSC only
TSC_MsgCompleteTransfer	Not applicable	Generic TSC only
TSC_MsgDefineBSet	*TSC_MsgDefineBSetCmplt*	Generic TSC only
TSC_MsgDropCall	*Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgGetCallInfo	*TSC_MsgGetCallInfoCmplt, Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgGetCallState	*TSC_MsgGetCallStateCmplt, Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgGetChanState	*TSC_MsgGetChanStateCmplt, Std_MsgEvtDetected*	Generic TSC only

Command Message	**Reply Message (indicates success)**	**Used by Which TSC Component**
TSC_MsgHoldCall	Not applicable	Not applicable
TSC_MsgInitTransfer	Not applicable	Not applicable
TSC_MsgMakeCall	*TSC_MsgMakeCallCmplt, Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgParkCall	Not applicable	Not applicable
TSC_MsgPickupCall	Not applicable	Not applicable
TSC_MsgReconnectCall	Not applicable	Not applicable
TSC_MsgRedirectCall	Not applicable	Not applicable
TSC_MsgRejectCall	*Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgReleaseCall	*Std_MsgEvtDetected*	Generic TSC, NetTSC
TSC_MsgRetrieveCall	Not applicable	Not applicable
NetTSC_MsgSendNonStandardCmd	*Std_MsgEvtDetected*	NetTSC
NetTSC_MsgSendUserInputIndication	*Std_MsgEvtDetected*	NetTSC
TSC_MsgSetChanState	*TSC_MsgSetChanStateCmplt, Std_MsgEvtDetected*	Generic TSC only
TSC_MsgWaitCall	*Std_MsgEvtDetected*	Generic TSC only

NetTSC_MsgSendNonStandardCmd

Definition

NetTSC_MsgSendNonStandardCmd is a command message that sends non-standard data on the H.245 channel, between two Dialogic gateways.

Used By Which TSC Components

NetTSC only

Message Structure

NetTSC_MsgSendNonStandardCmd contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call.
CmdLen	UInt16	The length of the data in the **CmdVal[100]** field, in bytes.
CmdVal[100]	Char	The non-standard data.

Call States

You may implement this command only in certain call states:

Valid Call State(s) for Command	Resulting Call State
Connected	Not applicable

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

This command message does not have a corresponding reply message.

If the message executes successfully, the following message(s) will be sent to the far gateway:

- *Std_MsgEvtDetected,* if the far gateway is registered for this type of event. In this case the message will indicate a NetTSC_EvtH245Data_Type_NonStdCmd event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
TBD	

Other Related Messages

None.

Cautions

NetTSC_MsgSendNonStandardCommand should be used only when both the caller and the called party are connected via Dialogic gateways.

NetTSC_MsgSendUserInputIndication

Definition

NetTSC_MsgSendUserInputIndication is a command message that puts DTMF input onto the H.245 channel.

Used By Which TSC Components

NetTSC only

Message Structure

NetTSC_MsgSendUserInputIndication contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call.
Length	UInt16	The length of the data in the **Val** field, in bytes.
Val	Char	The DTMF data.

Call States

You may implement this command only in certain call states:

Valid Call State(s) for Command	Resulting Call State
Connected	Not applicable

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

This command message does not have a corresponding reply message.

If the message executes successfully, the following message(s) will be sent to the far gateway:

- *Std_MsgEvtDetected,* if the far gateway is registered for this type of event. In this case the message will indicate a NetTSC_EvtH245Data_Type_UsrInputIndication event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
TBD	

Other Related Messages

None.

Cautions

None.

TSC_MsgAcceptCall

Definition

TSC_MsgAcceptCall is a command message that accepts an incoming call.

Used By Which TSC Components

Generic TSC, NetTSC

Additional Information

The *TSC_MsgAcceptCall* message handles any call **acceptance,** which involves sending an acknowledgement to the sender that the call has been received but no audio channel has been opened yet; this keeps the system from timing out; the *TSC_MsgAnswerCall* message handles the actual call **answering.**

The *TSC_MsgAcceptCall* message is used under these conditions:

- the offered call is intended to be answered either by
  - the accepting client, or
  - a client that is the target of a hand-off
- the called party is detected to be free but answer is not certain

The *TSC_MsgAcceptCall* message may initiate alerting to the calling and/or called party, as required by the telephony network environment.

Message Structure

TSC_MsgAcceptCall contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call to accept.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Offered	Accepted

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgAnswerCall*
- *TSC_MsgWaitCall*

Cautions

None

TSC_MsgAnswerCall

Definition

TSC_MsgAnswerCall is a command message that answers a call.

Used By Which TSC Components

Generic TSC, NetTSC

Additional Information

The *TSC_MsgAnswerCall* message handles the actual call **answering,** which opens the audio path between the two end points.

Message Structure

TSC_MsgAnswerCall contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to answer.
NumRings	UInt32	The number of rings to wait prior to answering the call. If number of rings has already been received at the time the command is received, then the call will be answered immediately.
KVSet	UInt32	An optional Key-Value set specifying protocol-specific parameters for the operation. It is a placeholder for a KVSet structure. If no KVSet is appended to the *TSC_MsgAnswerCall* message, leave this field out of the message. For more information about KVSets, see *Appendix F*. **NOTE:** You may use multiple KVSet fields.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Offered, Accepted	• Connected (if the call is successfully answered) • Idle (if the remote party drops the call prior to completion of answer operation)

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgWaitCall*

Cautions

None

TSC_MsgBlindTransferCall

Definition

TSC_MsgBlindTransferCall is a command message that performs a blind transfer of a call.

Used By Which TSC Components

Generic TSC only

Additional Information

A blind transfer typically involves the following steps:

1. The call to be transferred (CallId) is placed on hold (HoldPendXfer).
2. A consultation call is made to the specified destination address (DestAddr).
3. One of the following:
    - The network may reconnect the call to be transferred (CallId), to let the client inform the remote party associated with the call (CallId) that the transfer attempt is still in progress. In this case, the call state transitions to ConnectedPendXfer.
    - The transfer is completed automatically prior to the consultation call reaching the Connected state.
4. The initial call (which was placed on hold) transitions to the Idle state.
5. The client sends a *TSC_MsgReleaseCall* command to the TSC instance, thereby freeing the call identifier (CallId).

Message Structure

TSC_MsgBlindTransferCall contains these fields:

Field	Type	Description
CallId	Uint32	The identifier of the call to transfer
DestAddr[128]	Char	The destination for the transferred call.
OrigAddr[128]	Char	The address of the transfer originator. **NOTE:** This may be an optional parameter; refer to the Protocol Implementation notes.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Intermediate Call State(s)	Resulting Call State
Connected	HoldPendXfer, ConnectedPendXfer	• Idle • Connected (if the call is not successfully transferred)

For more information about call states, see *Appendix G* on page 149.

Returned Message: Acknowledgement

The TSC instance may send a *Std_MsgAck* message to the host upon receiving the *TSC_MsgBlindTransferCall* command.

NOTE: The *Std_MsgAck* message does not indicate successful execution of the command. It indicates only the successful **receipt** of the command.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgBlindTransferCmplt*
- *TSC_MsgTransferCall*

Cautions

The outcome of the consultation call for a blind transfer is not checked.

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* message once the transfer completes (once the call state transitions to Idle).

TSC_MsgCancelTransfer

Definition

TSC_MsgCancelTransfer is a command message that cancels a supervised transfer being performed

Used By Which TSC Components

Generic TSC only

Additional Information

When a supervised transfer is canceled, the consultation call and the call to be transferred will change their states, from <state>PendXfer to <state>, where <state> is one of the following:

- Connected (typically the consultation call)
- Hold (typically the call to be transferred)

The original state of each of the calls is dependent on what operations the client has issued on the calls. If the client has issued a *TSC_MsgRetrieveCall* command for the held call (for example, to alternate between the party being transferred and the party being transferred to), then the Connected and Hold states will be reversed for the two calls.

Message Structure

TSC_MsgCancelTransfer contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the consultation call associated with the supervised transfer.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Call	Valid Call State(s) for Command	Resulting Call State
Consultation	ConnectedPendXfer: indicates the other call is in the HoldPendXfer state	Connected
	HoldPendXfer: indicates the other call is in the Connected PendXfer state	Hold
Call to be transferred	ConnectedPendXfer: indicates the other call is in the HoldPendXfer state	Connected
	HoldPendXfer: indicates the other call is in the Connected PendXfer state	Hold

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	**Description**
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgCompleteTransfer*
- *TSC_MsgTransferCall*

Cautions

The client must issue either a *TSC_MsgDropCall* message, *TSC_MsgReleaseCall* message, or both messages on the consultation call to drop the call and free the call identifier.

TSC_MsgCancelComplete

Definition

TSC_MsgCancelComplete is a command message that cancels a previously initiated complete-call operation.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_MsgCancelComplete contains this field:

Field	Type	Description
CmpltId	UInt32	The identifier of the completed call.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.

Other Related Messages

- *TSC_MsgCompleteCallCmplt*

Cautions

None.

TSC_MsgCompleteCall

Definition

TSC_MsgCompleteCall is a command message that requests a call-completion service on a failed call.

Used By Which TSC Components

Generic TSC only

Additional Information

The *TSC_MsgCompleteCall* command requests that the switch or network, on behalf of the requester, to continue attempts to complete a call in the Failed state.

Example: A *TSC_MsgMakeCall* operation may have terminated with a call-state transition to the Failed state because the called party was busy. The client can then issue the *TSC_MsgCompleteCall* command to request that the switch continue to attempt to complete the call according to one of a number of supported completion modes. For the specified completion mode, the call is queued by the switch or network, which will continue to attempt the call completion.

To cancel the call completion request, the client must issue a *TSC_MsgCancelComplete* command.

When the switch or network does eventually complete the call, the TSC will generate a TSC_EvtCallState event with the call state Offered. The client may then acquire information pertaining to the offered call, such as the reason for the call, which in this case will be call completion along with the completion ID.

Message Structure

TSC_MsgCompleteCall contains these fields:

Field	Type	Description	
CallId	UInt32	The identifier of the call to be completed	
Mode	UInt32	The completion mode to use for the call-completion service. Values:	
		Value	**Description**
		CampOn	TBD
		CallBack	The called station or telephony interface device is directed to return the call when the station becomes idle.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Failed	Idle

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.
ErrMaxRequest	The maximum number of active call-completion requests has been exceeded.

Other Related Messages

- TSC_MsgCompleteCallCmplt
- *TSC_MsgMakeCall*

Cautions

Once the call state transitions to Idle, the client must issue a *TSC_MsgReleaseCall* message to free the call identifier.

TSC_MsgCompleteCallCmplt

Definition

TSC_MsgCompleteCallCmplt is a reply message for the *TSC_MsgCompleteCall* command. It indicates a call-completion operation has been successfully initiated.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_MsgCompleteCallCmplt contains this field:

Field	Type	Description
CmpltId	UInt32	The identifier of the completed call

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- TSC_MsgCompleteCall

Cautions

None.

TSC_MsgCompleteTransfer

Definition

TSC_MsgCompleteTransfer is a command message that completes a call transfer initiated by a *TSC_MsgTransferCall* command.

Used By Which TSC Components

Generic TSC only

Additional Information

The *TSC_MsgCompleteTransfer* command can also be used to transfer a call that is on hold to the specified consultation call, without the transfer first being set up via the *TSC_MsgTransferCall* command, if the protocol supports this operation.

Message Structure

TSC_MsgCompleteTransfer contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to be transferred
ConCallId	UInt32	The identifier of the consultation call associated with the supervised transfer.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Call	Valid Call State(s) for Command	Resulting Call State
Consultation	ConnectedPendXfer: indicates the other call is in the HoldPendXfer state	Idle
	HoldPendXfer: indicates the other call is in the Connected PendXfer state	
Call to be transferred	ConnectedPendXfer: indicates the other call is in the HoldPendXfer state	
	HoldPendXfer: indicates the other call is in the Connected PendXfer state	

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgCancelTransfer*
- *TSC_MsgTransferCall*

Cautions

Once the call state for each call transitions to Idle, the client must issue a *TSC_MsgReleaseCall* message to free the call identifiers.

TSC_MsgDefineBSet

Definition

TSC_MsgDefineBSet is a command message that defines a bearer-channel set.

Used By Which TSC Components

Generic TSC only

Additional Information

The variable section of the message is used to define the mappings between channels and physical network timeslots. This section is optional and need only be specified if the default mapping has to be overridden.

Message Structure

TSC_MsgDefineBSet contains these fields:

Field	Type	Description
SetId	Uint8	The ID of the bearer-channel set being defined.
LineId	Uint8	The ID of the line or trunk on which this set resides. Values: 1 through 4.
StartChannelId	Uint8	The ID of the first bearer channel in the set, relative to the trunk. Values: 1 through 24 or 1 through 30.
NumChannels	UInt8	TBD
BaseProtocol	UInt32	The base protocol type. Values: T1-CAS, ISDN.

Field	Type	Description
InboundVariantId	UInt8	The ID of the protocol variant to run for inbound bearer channels.
OutboundVariantId	UInt8	The ID of the protocol variant to run for outbound bearer channels.
DChannelDesc	Compdesc	For CCS protocols only. The address of the D-Channel instance responsible for the bearer channel set. Note: The DChannelDesc.component field should be set to the CCS component ID rather than the actual component address, since this will be used to locate the instance via the **qCompFind( )** function.
AdminGroup	UInt32	An arbitrary administration group identifier given to the bearer-channel set, to be used by applications when requesting bearer channels. Examples: This could be the line identifier AT&T, MCI, Low-Cost direct to Gammalink.
Width	UInt8	The number of timeslots used by each bearer channel. Values: Typically 1.
The following four fields form an array of 4-tuples, terminated by a **BChanId** of 0. If the default mapping is used, then omit these four fields from the message.		
BChanId	UInt8	The ID of the bearer-channel from which to begin timeslot mapping.
SlotId	UInt8	The ID of the timeslot that the bearer channel maps onto.

Field	Type	Description
Direction	UInt8	The call direction on the bearer channel. Values: • outbound • inbound • both
Count	UInt8	The number of channels to map to, incrementing BChanId and SlotId each time.

Example 1:T1-CAS Bearer Channel Set for Trunk 2

```
SetId              = 20   /* Arbitrary set id selected by admin
                          client */
LineId             = 2    /* Trunk 2 */
StartChannelId     = 1    /* Start at the first bearer channel
                          */
NumChannels        = 24   /* Put all 24 channels in this set */
BaseProtocol       = TSC_ParmBaseProtocol_T1_CAS
InboundVariantId   = 2    /* Previously defined T1 E&M variant
                          */
OutboundVariantId  = 2    /* Previously defined T1 E&M variant
                          */
DChannelDesc              /* Not used */
AdminGroup         = MY_TELESALES_TRUNK_ID
Width              = 1    /* one timeslot per channel */
BChanId            = 0    /* Use default 1 to 1 mapping */
```

Example 2: E1 ISDN Bearer Channel Set for Trunk 1, Skipping Physical Timeslot 16

```
SetId              = 10   /* Arbitrary set id selected by admin
                          client */
LineId             = 1    /* Trunk 1 */
StartChannelId     = 1    /* Start at the first bearer channel
                          */
NumChannels        = 30   /* Put all 30 channels in this set */
BaseProtocol       = TSC_ParmBaseProtocol_ISDN
```

```
InboundVariantId   = 1    /* Previously defined 4ESS variant */
OutboundVariantId  = 1    /* Previously defined 4ESS variant */
DChannelDesc.node  = 0    /* Not currently used */
DChannelDesc.board  = MY_BOARD_NUMBER
DChannelDesc.proc  = 0    /* Not currently used */
essor
DChannelDesc.comp  = CCS_Std_ComponentType
onent
DChannelDesc.inst  = 1    /* Use D Channel 1 for trunk 1 */
ance
AdminGroup         = MY_LEASED_TRUNK_ID
Width              = 1    /* one timeslot per channel */
BChanId            = 1    /* First map timeslots 1-15 to
                          channels 1-15 */
SlotId             = 1
Direction          = TSC_MsgDefineBSet_Direction_Both
Count              = 15
BChanId            = 16   /* Next map timeslots 17-31 to
                          channels 16-30 */
SlotId             = 17
Direction          = TSC_MsgDefineBSet_Direction_Both
Count              = 15
BChanId            = 0    /* Terminate list */
```

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgDefineBSetCmplt*
- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrChannelRange	The range specified in the bearer-channel set exceeds the trunk boundary (the line cannot support this many channels).
ErrIdentifier	An invalid, message-specific identifier was specified. In this case the set ID is invalid (must be in the range 1 to 255).
ErrIdentifierInUse	The specified identifier is already in use. In this case the set ID is already being used for another set.

Other Related Messages

- *TSC_MsgDefineBSetCmplt*

Cautions

A bearer-channel set cannot span two trunks. The channel range specified by **StartChannelId** and **NumChannels** must not exceed the limit for one trunk:

- 24 for T-1 CAS

- 30 for E-1 CAS
- 23 for T1 ISDN
- 30 for E1 ISDN.

TSC_MsgDefineBSetCmplt

Definition

TSC_MsgDefineBSetCmplt is a reply message for the *TSC_MsgDefineBSet* command. It indicates that the bearer-channel set has been successfully defined.

Used By Which TSC Components

Generic TSC only

Message Structure

Not applicable: This message has no fields.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgDefineBSet*
- *TSC_MsgGetBSet*

Cautions

None.

TSC_MsgDropCall

Definition

TSC_MsgDropCall is a command message that instructs the TSC instance to drop or disconnect a call.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgDropCall contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call to drop.
Reason	UInt32	The reason for dropping the call. Values: See *Table 9* on page 148.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Accepted, Connected, Delivered, Dialing, DialReady, Disconnected, Failed, Hold, Initiated, Originated, etc.	Idle

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	**Description**
ErrBadParm	Invalid value in the **Reason** field.
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgReleaseCall*

Cautions

1. The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command after performing a *TSC_MsgDropCall* command. For more information about the *TSC_MsgReleaseCall* command, see *TSC_MsgReleaseCall* on page 88.

   Prior to releasing the call, the client may make inquires using the call identifier to obtain call-related information such as time and charges.

2. For ISDN protocols the **Reason** field is translated into the Cause IE issued to the network.

TSC_MsgGetCallInfo

Definition

TSC_MsgGetCallInfo is a command message that requests stored information related to a call.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgGetCallInfo contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call on which to obtain information.
Count	UInt32	The number of **InfoId** structures in the variable body of this message.
InfoId	UInt32	Identifies the format and type of data to return.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgGetCallInfoCmplt*
- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallInfo event.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrParmValue	Invalid parameter value for **InfoId.**
ErrUnavailable	Information is not available.

Other Related Messages

- *TSC_MsgGetCallInfoCmplt*

Cautions

Note that this command may not provide as complete call information as that provided by the *Std_MsgEvtDetected* message about a TSC_EvtCallInfo event. The information available through polling (for example, with a *TSC_MsgGetCallInfo* message) is yet to be determined and will in part be dependent upon the type of protocol that the TSC implements.

TSC_MsgGetCallInfoCmplt

Definition

TSC_MsgGetCallInfoCmplt is a reply message for the *TSC_MsgGetCallInfo* command. It contains the requested call information asked for by the *TSC_MsgGetCallInfo* command.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgGetCallInfoCmplt contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call that this information relates to.
Count	UInt8	The number of **CallInfo** structures in the variable body of this message.
CallInfo	UInt32	A placeholder for the CallInfo structure. The CallInfo structure is a variable length array of the Call Information elements requested in the original *TSC_MsgGetCallInfo*.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgGetCallInfo*

Cautions

None

TSC_MsgGetCallState

Definition

TSC_MsgGetCallState is a command message that requests the current state of a call.

Used By Which TSC Components

Generic TSC, NetTSC (not yet supported)

Additional Information

Using a **CallId** of 0 requests the state of the current call on the channel. If no call is present on the channel, then the *TSC_MsgGetCallStateCmplt* message will indicate a call state of Null.

Message Structure

TSC_MsgGetCallState contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call to obtain the state of. Value: 0.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgGetCallStateCmplt*

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.

Other Related Messages

- *TSC_MsgGetCallStateCmplt*

Cautions

None

TSC_MsgGetCallStateCmplt

Definition

TSC_MsgGetCallStateCmplt is a reply message for the *TSC_MsgGetCallState* command. It returns the current state of a call.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgGetCallStateCmplt contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call that the call state belongs to.
CallState	UInt32	The current call state. If no call is present on the channel, then the *TSC_MsgGetCallStateCmplt* message will indicate a call state of Null. For more information, see *Table 10* on page 149.
Reason	UInt32	The reason for the transition to the current call state, if any. For more information, see *Table 11* on page 152.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

None

Cautions

None

TSC_MsgGetChanState

Definition

TSC_MsgGetChanState is a command message that requests the call state of a channel.

Used By Which TSC Components

Generic TSC only

Message Structure

Not applicable: This message has no fields.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgGetChanStateCmplt*
- *Std_MsgEvtDetected* for TSC_EvtChanState if the host application is registered for this type of event

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgGetChanStateCmplt*
- *TSC_MsgSetChanState*

Cautions

None

TSC_MsgGetChanStateCmplt

Definition

TSC_MsgGetChanStateCmplt is a reply message for the *TSC_MsgGetChanState* command. It returns the call state of a channel.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_MsgGetChanStateCmplt contains this field:

Field	Type	Description
ChanState	UInt32	The current channel state.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgGetChanState*

Cautions

None

TSC_MsgHoldCall

Definition

TSC_MsgHoldCall is a command message that places a call on hold.

Used By Which TSC Components

Generic TSC only

Additional Information

A call that is on hold can be retrieved from hold by using the commands *TSC_MsgRetrieveCall* and *TSC_MsgReconnectCall*.

Message Structure

TSC_MsgHoldCall contains this field:

CallId	UInt32	The identifier of the call to put on hold.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Accepted, Connected, Delivered, Dialing, Initiated	Hold

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* for TSC_EvtCallState if the host application is registered for this type of event

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgReconnectCall*
- *TSC_MsgRetrieveCall*

Cautions

The client should ensure that the call is in the Active state prior to issuing this command (or in a protocol-defined state from which a transition to Hold is valid).

TSC_MsgInitTransfer

Definition

TSC_MsgInitTransfer is a command message that initiates a supervised transfer of a call.

Used By Which TSC Components

Generic TSC only

Additional Information

A successful *TSC_MsgInitTransfer* command initiates the following steps:

1. The call to be transferred is put on hold (or HoldPendXfer).
2. The originating party connects to the destination address to make a consultation call.

After the transfer originator receives a *TSC_MsgInitTransferCmplt* message, which indicates success and provides the call ID, the originating party chooses one of the following actions:

- **complete** the transfer via the *TSC_MsgCompleteTransfer* message
- **cancel** the transfer operation via the *TSC_MsgCancelTransfer* message.

Message Structure

TSC_MsgInitTransfer contains these fields:

CallId	UInt32	The identifier of the call to transfer.
DestAddr[20]	Char	The destination for the transferred call.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Call	Valid Call State(s) for Command	Resulting Call State
Consultation	Null	Dialing
Call to be transferred	Connected	HoldPendXfer

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgInitTransferCmplt*
- *Std_MsgEvtDetected* for TSC_EvtCallState if the host application is registered for this type of event

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgBlindTransferCall*
- *TSC_MsgCancelTransfer*
- *TSC_MsgCompleteTransfer*
- *TSC_MsgInitTransferCmplt*

Cautions

None

TSC_MsgInitTransferCmplt

Definition

TSC_MsgInitTransferCmplt is a reply message for the *TSC_MsgInitTransfer* command. It confirms a call transfer has been successfully initiated.

Used By Which TSC Components

Generic TSC only

Additional Information

Once the *TSC_MsgInitTransferCmplt* message is sent, the originating party chooses one of the following actions:

- **complete** the transfer via the *TSC_MsgCompleteTransfer* message
- **cancel** the transfer operation via the *TSC_MsgCancelTransfer* message.

Message Structure

TSC_MsgInitTransferCmplt contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call to transfer.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgCancelTransfer*
- *TSC_MsgCompleteTransfer*
- *TSC_MsgInitTransfer*

Cautions

The client should monitor the call state of the consultation call in the same way it would for a standard call initiated with a *TSC_MsgMakeCall* command.

TSC_MsgMakeCall

Definition

TSC_MsgMakeCall is a command message that initiates a call to a specified destination address.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgMakeCall contains these fields:

Field	Type	Description
DestAddr[128]	char	The destination address for the call. • **For generic TSC:** typically the phone number to dial. • **For NetTSC:** the IP address.
OrigAddr[128]	char	The address of the originator of the call.
CallProgress	UInt8	Determines whether call progress is monitored prior to the call being completed (in the Connected state, such as for Busy, Ringback, and SIT tones) and post-connection call analysis is run once the call is connected (as for PVD and PAMD). Values: • Disabled=0 (the default) • Enabled=1

KVSet	UInt32	An optional Key-Value set specifying protocol-specific parameters for the operation. It is a placeholder for a KVSet structure. If no KVSet is appended to the *TSC_MsgAnswerCall* message, leave this field out of the message. For more information about KVSets, see *Appendix F*. **NOTE:** You may use multiple KVSet fields.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Null	Initiated

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgMakeCallCmplt*
- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrUnsupportedOp	Operation not supported by channel.
ErrChanState	Invalid channel state (there is an active call on the channel).
ErrKVSetNotSupported	The key value set specified in the **KVSet** field is not supported by this command. For more information about KVSets, see *Appendix F*.

Other Related Messages

- *TSC_MsgMakeCallCmplt*
- *TSC_MsgGetCallState*

Cautions

Note that the response to this message does not indicate that the call has been successfully made. The call state must be tracked by enabling the generation of asynchronous messages. *Std_MsgEvtDetected*

TSC_MsgMakeCallCmplt

Definition

TSC_MsgMakeCallCmplt is a reply message for the *TSC_MsgMakeCall* command. It confirms that a call has been initiated.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgMakeCallCmplt contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the initiated call.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgMakeCall*
- *TSC_MsgGetCallState*

Cautions

The *TSC_MsgMakeCallCmplt* message does not indicate a successful call; it only verifies that the call has been initiated. The client must track the call state of the call to determine if the call is successful or not, by enabling the generation of asynchronous *Std_MsgEvtDetected* messages for TSC_EvtCallState events. For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

TSC_MsgParkCall

Definition

TSC_MsgParkCall is a command message that parks a call.

Used By Which TSC Components

Generic TSC only

Additional Information

The call identifier associated with the call must be released by the client. When the call is picked up from the park address (via the *TSC_MsgPickupCall* message), a new call identifier will be returned.

Message Structure

TSC_MsgParkCall contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to be parked.
ParkAddr[20]	char	The address of where to park the call.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Connected, Hold	Idle

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgParkCallCmplt*
- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.
ErrParmValue	Invalid parameter value for **CallId.**

Other Related Messages

- *TSC_MsgParkCallCmplt*

Cautions

None

TSC_MsgParkCallCmplt

Definition

TSC_MsgParkCallCmplt is a reply message for the *TSC_MsgParkCall* command. It confirms that a call has been successfully parked.

Used By Which TSC Components

Generic TSC only

Additional Information

The *TSC_MsgPickupCall* picks up a parked call.

Message Structure

TSC_MsgParkCallCmplt contains these fields:

Field	Type	Description
ParkId	UInt32	The identifier of the parked call. Note: This is not a call identifier; the park identifier identifies which park the call is in at the park address.
ParkAddr	Char	The address at which the call is parked.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgParkCall*
- *TSC_MsgPickupCall*

Cautions

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command.

TSC_MsgPickupCall

Definition

TSC_MsgPickupCall is a command message that picks up a parked call.

Used By Which TSC Components

Generic TSC only

Additional Information

The call was parked with the *TSC_MsgParkCall* message.

The park identifier (**ParkId**) is automatically released when the call is picked up. Park identifiers are used solely in call park and pickup operations and are not related to call identifiers.

Message Structure

TSC_MsgPickupCall contains these fields:

Field	Type	Description
ParkId	UInt32	The identifier of the parked call. Note: This is not a call identifier; the park identifier identifies which park the call is in at the park address.
ParkAddr[20]	Char	The address at which the call is parked.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgPickupCallCmplt*
- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	**Description**
ErrCallIdentifier	Invalid call identifier.
ErrParmValue	Invalid parameter value for **ParkId.**

Other Related Messages

- *TSC_MsgParkCall*
- *TSC_MsgPickupCallCmplt*

Cautions

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command.

TSC_MsgPickupCallCmplt

Definition

TSC_MsgPickupCallCmplt is a command message that indicates a call has been successfully picked up.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_MsgPickupCallCmplt contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the picked-up call.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Not applicable	Connected

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *TSC_MsgParkCall*
- *TSC_MsgPickupCall*

Cautions

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command.

The *TSC_MsgPickupCallCmplt* message does not indicate a successful call pickup; it only verifies that the call pickup has been initiated. The client must track the call state of the call to determine if the call is successful or not, by enabling the generation of asynchronous Std_MsgEvtDetected messages for TSC_EvtCallState events. For more information about the Std_MsgEvtDetected event message and enabling events, see Appendix B and the DM3 Standard Messages and Run Time Control guide.

TSC_MsgReconnectCall

Definition

TSC_MsgReconnectCall is a command message that retrieves a call on hold, making it the active call.

Used By Which TSC Components

Generic TSC only

Additional Information

- If another call is currently active, then the currently active call will be dropped.
- The *TSC_MsgReconnectCall* command is the equivalent of issuing a *TSC_MsgDropCall* command on the active call, and then a *TSC_MsgRetrieveCall* command on the call being held.

Message Structure

TSC_MsgReconnectCall contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call on hold.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgHoldCall*

Cautions

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command.

TSC_MsgRedirectCall

Definition

TSC_MsgRedirectCall is a command message that redirects a call.

Used By Which TSC Components

Generic TSC only

Additional Information

The *TSC_MsgRedirectCall* command lets the client redirect an incoming call after receiving call-related information such as ANI or DNIS.

Message Structure

TSC_MsgRedirectCall contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to be redirected.
DestAddr[20]	Char	The destination for the redirected call.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Offered	Idle

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	**Description**
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

None

Cautions

The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command.

TSC_MsgReleaseCall

Definition

TSC_MsgReleaseCall is a command message that releases a call.

Used By Which TSC Components

Generic TSC, NetTSC

Additional Information

- Whenever the application no longer requires a call identifier (for example, when the call has been terminated and the information associated with the call has been retrieved or is no longer needed), the call identifier must be released back to the TSC or NetTSC component so the identifier and its associated resources can be reused for another call.
- The client should free the call identifier by issuing a *TSC_MsgReleaseCall* command after performing a *TSC_MsgDropCall* command. For more information about the *TSC_MsgDropCall* command, see *TSC_MsgDropCall* on page 46.

  Prior to releasing the call, the client may make inquires using the call identifier to obtain call-related information such as time and charges.
- If the call has not yet been dropped when the *TSC_MsgReleaseCall* is issued, the *TSC_MsgReleaseCall* command will first perform a drop-call operation on the call before releasing it.

Message Structure

TSC_MsgReleaseCall contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to be released.
Reason	UInt32	The reason for releasing the call. This field is used if the release operation results in the call being dropped and then released. Values: See *Table 9* on page 148. Note: This field is used only if the call is in a non-Idle state (meaning the *TSC_MsgReleaseCall* command results in the call being Dropped and then Released).

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
For generic TSC: Accepted, Connected, Delivered, Dialing, DialReady, Disconnected, Failed, Idle, Initiated, Originated **For NetTSC:** Accepted, Connected, Disconnected, Failed, Idle, Initiated, Originated	Null

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	**Description**
ErrBadParm	Invalid value in the **Reason** field.
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgDropCall*

Cautions

1. The call should be terminated by the client before its call identifier is released. If a call is active when the *TSC_MsgReleaseCall* command is issued, then the TSC instance will typically drop the call before releasing the identifier. Releasing an active call will result in a call-state transition to Idle, followed by the expected transition to Null.

2. For ISDN protocols, the **Reason** field is translated into the Cause IE issued to the network.

TSC_MsgRejectCall

Definition

TSC_MsgRejectCall is a command message that rejects an incoming, unanswered call.

Used By Which TSC Components

Generic TSC, NetTSC

Message Structure

TSC_MsgRejectCall contains these fields:

Field	Type	Description
CallId	UInt32	The identifier of the call to be rejected.
Reason	UInt32	The reason for the rejection. Values: • Busy • Congestion • NumUnavailable • Operator • NumVacant

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Accepted, Offered	Idle

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrBadParm	Invalid value in the **Reason** field.
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- *TSC_MsgRedirectCall*

Cautions

- The call identifier should be released by the client following the call-reject operation.
- For ISDN protocols, the **Reason** field values are the same as those specified for the DropCall and ReleaseCall messages.

TSC_MsgRetrieveCall

Definition

TSC_MsgRetrieveCall is a command message that retrieves a call on hold, making it the active call.

Used By Which TSC Components

Generic TSC only

Additional Information

- If another call is currently active, then the currently active call will be put on hold.
- The *TSC_MsgRetrieveCall* command can be used to alternate between a call on hold and an active call.

Message Structure

TSC_MsgRetrieveCall contains this field:

Field	Type	Description
CallId	UInt32	The identifier of the call to be retrieved.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Hold	Connected

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrCallIdentifier	Invalid call identifier.
ErrCallState	Operation not supported in the current call state.

Other Related Messages

- TSC_MsgHoldCall

Cautions

The call identifier should be released by the client following the call-reject operation.

TSC_MsgSetChanState

Definition

TSC_MsgSetChanState is a command message that sets the channel state.

Used By Which TSC Components

Generic TSC only

Additional Information

It is typically used to take a channel into or out of service.

Message Structure

TSC_MsgSetChanState contains this field:

Field	Type	Description
state	UInt32	The state to set the channel to. • Idle • LocalOutOfService

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *TSC_MsgSetChanStateCmplt*

- *Std_MsgEvtDetected* if the host application is registered for notification about this type of event. In this case the message will indicate a TSC_EvtChanState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrParmValue	Invalid parameter value for **state.**

Other Related Messages

- *Std_MsgEvtDetected* for TSC_EvtChanState if the host application is registered for this type of event

Cautions

- If the channel state is currently in one of the out-of-service states (Alarm, LocalMaintenence, Maintenance, or OutOfService), then the new state change will not take effect until the cause of the out-of-service state is removed.
- Taking a channel out of service may result in the termination of all calls on the channel.
- While a channel is out of service, no incoming calls will be offered to the client.

TSC_MsgSetChanStateCmplt

Definition

TSC_MsgSetChanStateCmplt is a reply message for the *TSC_MsgSetChanState* command. It confirms the start of the channel-state setting.

Used By Which TSC Components

Generic TSC only

Message Structure

Not applicable: This message has no fields.

Call States

Not applicable: This message does not depend on call states.

Returned Message: Success

Not applicable: This message **is** success-indicating reply message, for the command message that triggered it.

Returned Message: Error

Not applicable: Because this message is a reply message for a command message, it does not send the host application a return message that indicates command failure.

Other Related Messages

- *Std_MsgEvtDetected* for TSC_EvtChanState if the host application is registered for this type of event
- *TSC_MsgSetChanState*

Cautions

If the channel state is currently in one of the out-of-service states (Alarm, LocalMaintenence, Maintenance, or OutOfService), then the new state change will not take effect until the cause of the out-of-service state is removed.

TSC_MsgWaitCall

Definition

TSC_MsgWaitCall is a command message that asks the protocol to wait for an incoming call.

Used By Which TSC Components

Generic TSC only

Additional Information

- The *TSC_MsgWaitCall* command enables the generation of a *Std_MsgEvtDetected* message about a TSC_EvtCallState event to the client, for the next call that is Offered, and then disables it once the event has been generated.
- The *TSC_MsgWaitCall* message controls the reporting of incoming calls to the client. When using this command, the client disables detection of Offered-call events, and issues the *TSC_MsgWaitCall* message when it is ready to receive a new call.
- If an incoming call is offered to the TSC instance, and the *TSC_MsgWaitCall* message has not yet been received, then the TSC instance will hold off reporting events for that call to the client. If the client does not issue the *TSC_MsgWaitCall* message before the call is lost (for example, before the remote party disconnects), then the TSC instance will release the call identifier for the call.
- When the client issues a *TSC_MsgWaitCall* message, a *Std_MsgEvtDetected* message about a Offered TSC_EvtCallState event will be generated to the client if there is an offered call pending, or else when a new call is offered. Once the event is sent to the client, normal call-state transition events are reported for the duration of call.

- The client must issue a *TSC_MsgWaitCall* message each time it wishes to receive a new call. The *TSC_MsgWaitCall* message provides a "one-shot" event detection mechanism for the Offered call-state transition.

Message Structure

Not applicable: This message has no fields.

Call States

You may implement this command only in certain call states; doing otherwise returns an error message. When this command executes successfully, it performs a transition to a new call state.

Valid Call State(s) for Command	Resulting Call State
Not applicable	Offered

For more information about call states, see *Appendix G* on page 149.

Returned Message: Success

If the message executes successfully, the following message(s) will be returned to the host application:

- *Std_MsgEvtDetected* if the host application is registered for this type of event. In this case the message will indicate a TSC_EvtCallState event.

  For more information about the *Std_MsgEvtDetected* event message and enabling events, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.

Returned Message: Error

If an error occurs in the execution of the command, a *Std_MsgError* message is sent to the client, with the **ErrorCode** field indicating the reason for the command failure.

NOTE: If this attempted command has a success-indicating reply message (a message ending in "Cmplt"), the *Std_MsgError* message will be sent to the client instead of this reply message.

For more information about the *Std_MsgError* message, see the *DM3 Standard Messages and Run Time Control* guide.

The following errors are valid for this particular command message. The appropriate error from this table will be reported in the **ErrorCode** field of the *Std_MsgError* message:

Error	Description
ErrChanState	Invalid channel state (there is an active call on the channel).
ErrUnsupportedOp	Operation not supported by channel.

Other Related Messages

None

Cautions

None

Appendix A
Error Types of the TSC Component

How Errors Are Reported

If an error occurs when a TSC component or component instance tries to execute a command, a *Std_MsgError* message will be sent to the client (the source of the command message), alerting it to the error.

Many command messages trigger a corresponding reply message when the command goes through successfully—for example, after a TSP component or instance receives a *TSP_MsgMakeCall* command message, it sends a *TSP_MsgMakeCallCmplt* to the client (the source of the command message) when it successfully initiates a new call. When an error occurs, the corresponding reply message is preempted by the *Std_MsgError* message.

The *Std_MsgError* message is the only message that conveys error information. To accommodate a variety of errors, the *Std_MsgError* message has a variable body; the type of error that has occurred determines the type and number of fields inside the message.

Each component has its own set of valid error types. TSC errors are discussed in this appendix.

How Error Notification Works

The following process explains how error notification works:

1. The host sends a command message to a primary component or component instance.
2. An error occurs in the execution of this command.
3. The primary component or component instance sends a *Std_MsgError* message to the client (the source of the command message).

Error Types

The following errors are used by the TSP and NetTSP resources.

Error Name	Description	Used by Which Message(s)
ErrNone	No error.	TBD
ErrUnknownMsg	Unknown message type not supported by TSC.	TBD
ErrCallIdentifier	Invalid call identifier. No call matching the call identifier exists on the TSC instance that received the command.	*TSC_MsgAcceptCall*, *TSC_MsgAnswerCall*, *TSC_MsgBlindTransferCall*, *TSC_MsgCancelTransfer*, *TSC_MsgCancelComplete*, *TSC_MsgCompleteCall*, *TSC_MsgCompleteTransfer*, *TSC_MsgDropCall*, *TSC_MsgGetCallInfo*, *TSC_MsgGetCallState*, *TSC_MsgHoldCall*, *TSC_MsgInitTransfer*, *TSC_MsgParkCall*, *TSC_MsgPickupCall*, *TSC_MsgReconnectCall*, *TSC_MsgRedirectCall*, *TSC_MsgReleaseCall*, *TSC_MsgRejectCall*, *TSC_MsgRetrieveCall*

Error Name	Description	Used by Which Message(s)
ErrCallState	Operation not supported in the current call state.	*TSC_MsgAcceptCall*, *TSC_MsgAnswerCall*, *TSC_MsgBlindTransferCall*, *TSC_MsgCancelTransfer*, *TSC_MsgCompleteCall*, *TSC_MsgCompleteTransfer*, *TSC_MsgDropCall*, *TSC_MsgHoldCall*, *TSC_MsgInitTransfer*, *TSC_MsgParkCall*, *TSC_MsgReconnectCall*, *TSC_MsgRedirectCall*, *TSC_MsgReleaseCall*, *TSC_MsgRejectCall*, *TSC_MsgRetrieveCall*
ErrChanState	Invalid channel state (there is an active call on the channel).	*TSC_MsgMakeCall*, *TSC_MsgWaitCall*
ErrExhausted	Too few resources to complete the command.	TBD
ErrIdentifierInUse	The specified identifier is already in use.	*TSC_MsgDefineBSet*
ErrParmValue	Invalid parameter value.	*TSC_MsgGetCallInfo*, *TSC_MsgParkCall*, *TSC_MsgPickupCall*, *TSC_MsgSetChannelState*
ErrNotEnabled	RTC events not enabled.	TBD
ErrSystem	A system error has occurred.	TBD

Error Name	Description	Used by Which Message(s)
ErrUnavailable	Information is not available.	*TSC_MsgGetCallInfo*
ErrUnsupportedOp	Operation not supported by channel.	*TSC_MsgMakeCall, TSC_MsgWaitCall*
ErrVariantId	Invalid or undefined protocol variant identifier.	TBD
ErrNotInitState	The TSC instance received a *Std_MsgInit* message while in a state other than the initialization state (it had already executed a *Std_MsgInit* command).	TBD
ErrIdentifier	An invalid, message-specific identifier was specified.	*TSC_MsgDefineBSet*
ErrChannelRange	The range specified in the bearer-channel set exceeds the trunk boundary (the line cannot support this many channels).	*TSC_MsgDefineBSet*
ErrBadParm	Invalid value in the **Reason** field.	*TSC_MsgDropCall, TSC_MsgReleaseCall, TSC_MsgRejectCall*

Error Name	Description	Used by Which Message(s)
ErrReadOnly	A client attempted to write a read-only parameter (by using a *Std_MsgSetParm* message).	TBD
ErrWriteOnly	A client attempted to read a write-only parameter (by using a *Std_MsgGetParm* message).	TBD
ErrKVSetNotSupported	The key value set specified in the **KVSet** field is not supported by this command.	*TSC_MsgMakeCall*

Appendix B

Event Types of the TSC and NetTSC Component

Overview

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. Event registration is explained later in this appendix.

The *Std_MsgEvtDetected* message alerts the client that one of the following types of events has occurred:

- an RTC event, which will later trigger a corresponding RTC action at the component instance level. Example: An instance of the signal detector has detected a DTMF tone for the numeral "3".
- a component instance of interest to the client is performing an action triggered by a command message, which was sent by the host to the component instance. Example: A TSC or NetTSC instance has dropped or disconnected a call after receiving a *TSC_MsgDropCall* message.

When a client receives a *Std_MsgEvtDetected* message, it should perform some RTC action appropriate for the type of event conveyed by the message. Before the client can do this, it must already be **configured** to perform the RTC action. Example: If an instance of the signal detector sends a "DTMF tone detected" *Std_MsgEvtDetected* message to a player instance (which is considered a client), the player instance must already be configured with an RTC response that corresponds to this particular DTMF tone—such as stopping the playback. The registration and event-reporting processes are covered later in this appendix.

Messages Used for Event Notification

Table 3 shows the standard messages that each DM3 resource uses for the event-notification process. For detailed descriptions of each of these messages, see the *DM3 Standard Messages and Run Time Control* guide.

Table 3. Standard Messages Used for Event Notification

Standard Message	Description
Std_MsgArmRTC	Configures a component or component instance to take an appropriate RTC action after receiving a *Std_MsgEvtDetected* message for a particular event. In order to receive a *Std_MsgEvtDetected* message about a particular event, the component or component instance must be registered for event notification.
Std_MsgArmxRTCs	Performs the same function that the *Std_MsgArmRTC* message does, except that one *Std_MsgArmxRTCs* message eliminates the need for multiple *Std_MsgArmRTC* messages to the same component or component instance.
Std_MsgDetectEvt	Performs the following two functions: • Registers a component or component instance to receive notification of an event immediately after it occurs. This registered entity is now called a client. • Enables the component instance experiencing an event to generate a *Std_MsgEvtDetected* message to the registered entity when the specified event does occur. Note that the client may or may not be the same entity that sent the *Std_MsgDetectEvt* message.

Standard Message	Description
Std_MsgDetectxEvts	Performs the same function that the *Std_MsgDetectEvt* message does, except that one *Std_MsgDetectxEvts* message eliminates the need for multiple *Std_MsgDetectEvt* messages to the same component or component instance.
Std_MsgEvtDetected	Sent to a registered client as notification that the event specified in the *Std_MsgDetectEvt* or *Std_MsgDetectxEvts* message has occurred.

About the *Std_MsgEvtDetected* Message

The *Std_MsgEvtDetected* message handles all event notification. To accommodate the wide variety of event types, the *Std_MsgEvtDetected* message has a variable body; the type of event that has occurred determines the type and number of fields inside the message.

Each component has its own set of valid event types. TSC and NetTSC component event types are covered later in this appendix.

About the Event-Reporting Process

In order for event reporting to occur, the following steps must be completed for each DM3 component or component instance:

- **Registering component instances as clients for event notification.** Some DM3 entities need to know when other instances have detected a particular event. Each component instance that will detect some type of event must receive a *Std_MsgDetectEvt* message with the return address of a client interested in notification of this type of event.

  Note that the client specified in this *Std_MsgDetectEvt* message may or may not be the entity that sent the message. In other words, one entity may register another entity for event notification. The client whose return address was specified in the *Std_MsgDetectEvt* message is now considered a registered client; it will receive a *Std_MsgEvtDetected* message from the component

instance that detected the event specified in the *Std_MsgDetectEvt* message, immediately after the event takes place.

- **Configuring component instances to perform RTC actions.** Any component instance that should perform a specific RTC action after receiving a *Std_MsgEvtDetected* message must first be configured to do so. The *Std_MsgArmRTC* message (for a single RTC action) and the *Std_MsgArmxRTCs* message (for multiple RTC actions) handle this type of configuration.

  The component instance that received either the *Std_MsgArmRTC* message or the *Std_MsgArmxRTCs* message is now configured to take some appropriate RTC action when it receives a *Std_MsgEvtDetected* message.

Procedure: Registering Clients for Event Notification

Follow these steps to register a component instance (a client) for event notification:

1. Send either a *Std_MsgDetectEvt* or a *Std_MsgDetectxEvts* message to the component instance (client) that should receive notification about an event. For more information about these messages, see the *DM3 Standard Messages and Run Time Control* guide.
2. Repeat step 1 for each additional client you wish to register for event notification.

Procedure: Deregistering Clients to Cease Receipt of Event Notification

Follow these steps to deregister a TSC or NetTSC instance so it no longer receives event notification about a particular event:

1. Send a *Std_MsgCancelEvt* message to the component instance that is currently configured to report the event to a TSC or NetTSC instance (the registered client).

   For more information about the *Std_MsgCancelEvt* message, see the *DM3 Standard Messages and Run Time Control* guide.

2. Repeat step 1 for each additional component instance configured to report the event to the TSC or NetTSC instance (the registered client).

Procedure: Configuring Clients to Perform RTC Actions

After a client is registered for specific types of event notification, it must also be configured to perform a corresponding RTC action for each *Std_MsgEvtDetected* message (event-notification message) it may receive.

Follow these steps to configure a client to perform an RTC action after receiving a *Std_MsgEvtDetected* message:

1. Send one of the following messages to a client:
   - a *Std_MsgArmRTC* message so the client will perform one RTC action
   - a *Std_MsgArmxRTCs* message so the client will perform multiple event actions
2. Repeat step 1 for each additional client you wish to configure for RTC action.

Message-Triggered Events Specific to the TSC and NetTSC Components

Table 4 lists the events specific to the TSC and NetTSC components. Depending on the event that has occurred, the body of the *Std_MsgEvtDetected* message will contain specific fields related to that event. See the listed page for more information about a particular event type.

Table 4. TSC and NetTSC Component Message-Triggered Event Types

Event	Body of the *Std_MsgEvtDetected* Message	Used by Which Component	See Page
NetTSC_EvtH245Data	notifies the application that non-standard data has been received on the H.245 channel	NetTSC only	120
NetTSC_EvtSystemFailed	indicates an H.323 failure (such as an NIC disconnect).	NetTSC only	121
NetTSC_EvtThresholdAlarm	indicates the value of the feature defined in the **Type** field has exceeded the threshold value.	NetTSC only	122
TSC_EvtCallInfo	delivers call-related information to the client as the information becomes available	Generic TSC, NetTSC	124
TSC_EvtCallInfoSet	delivers a set of call-related information to the client when this set becomes available	Generic TSC only	126
TSC_EvtCallState	indicates a change in call state	Generic TSC, NetTSC	128
TSC_EvtChanState	indicates a change in the channel state	Generic TSC only	130

Event	Body of the *Std_MsgEvtDetected* Message	Used by Which Component	See Page
TSC_EvtTrace	reports on a tracing condition	Generic TSC only	131

RTC Actions Specific to the Generic TSC Component

Table 5 lists the RTC action(s) specific to the generic TSC component and its instances. An RTC action is triggered at the instance level by receipt of a *Std_MsgEvtDetected* message—specifically, by the event type specified in the body of the *Std_MsgEvtDetected* message.

RTC actions are enabled using the *Std_MsgArmRTC* or *Std_MsgArmxRTCs* messages. All RTC actions are ignored if the TSC instance is in an invalid state; for example, TSC_RTC_Action_DropCall is ignored if the generic TSC instance has already dropped the call. For more information on Run Time Control, see the *DM3 Standard Messages and Run Time Control* guide.

Table 5. TSC Run Time Control Actions

RTC Action	Description
TSC_RTC_Action_DropCall	Drops the active call on the channel (if any).
TSC_RTC_Action_ChanOOSImmediate	Drops the active call on the channel (if any) and places the channel out of service.
TSC_RTC_Action_ChanOOSFinishCalls	Places the channel out of service once the present active call (if any) is finished.

These actions can be send with the QCompUse command to define what action should occur upon abnormal exit of a using application.

NetTSC_EvtH245Data

Definition

NetTSC_EvtH245Data is an asynchronous event that notifies the application that non-standard data has been received on the H.245 channel.

Used By Which TSC Components

NetTSC only

Message Structure

NetTSC_EvtH245Data contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. Values: • NonStdCmd • UsrInputIndication
CallId	UInt32	The identifier of the call.
Length	UInt16	The length of the data in the **Value[100]** field, in bytes.
Value[100]	Char	The non-standard data.

NetTSC_EvtSystemFailed

Definition

NetTSC_EvtSystemFailed is an asynchronous event that indicates an H.323 failure (such as an NIC disconnect).

Used By Which TSC Components

NetTSC only

Message Structure

NetTSC_EvtSystemFailed contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. Value: • H323

NetTSC_EvtThresholdAlarm

Definition

NetTSC_EvtThresholdAlarm is an asynchronous event that indicates the value of the feature defined in the **Type** field has exceeded the threshold value.

Used By Which TSC Components

NetTSC only (not supported in this release)

Message Structure

NetTSC_EvtThresholdAlarm contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. Values: • LostPackets • PacketsOutOfOrder • H245RoundTripDelay
CallId	UInt32	The identifier of the call.

Field	**Data Type**	**Description**	
TimeInterval	UInt32	The checking time interval. Values:	
		Value in the Label Field	**TimeInterval Field Value**
		H245RoundTripDelay	The time between two successive checkings, in seconds.
		Any other value	Given in 4-ms units.
Value	UInt32	One of the following values:	
		Value in the Label Field	**Value in the Value Field**
		LostPackets	The number of RTP packets that were lost during **TimeInterval.**
		PacketsOutOfOrder	The number of RTP packets that were received out of order during **TimeInterval.**
		H245RoundTripDelay	The round-trip delay time, in seconds.
ThreshVal	UInt32	The threshold value.	
Direction	UInt16	One of the following values:	
		Value	**Description**
		Rx	Packets were received from the remote site.
		Tx	Packets were sent from the host site to the remote site.

TSC_EvtCallInfo

Definition

TSC_EvtCallInfo is an asynchronous event that delivers call-related information to the client as the information becomes available.

Used By Which TSC Components

Generic TSC, NetTSC (not yet supported)

Message Structure

TSC_EvtCallInfo contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event.
Timestamp	UInt32	The board-level time at which the event was generated, in 4-millisecond units.
CallId	UInt32	The identifier of the call that the information is associated with.
InfoId	UInt32	Identifies the format and type of data that has been returned.
Value	UInt32	A 32-bit unsigned integer value. This is a variable field.

Field	Data Type	Description
ValueStr[20]	char	A character string used to return digit string information such as ANI, DTMF, and DNIS. This is a variable field.

TSC_EvtCallInfoSet

Definition

TSC_EvtCallInfoSet is an asynchronous event that delivers a set of call-related information to the client when this set becomes available.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_EvtCallInfoSet contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. For more information, see *Table 10* on page 149.
Timestamp	UInt32	The board-level time at which the event was generated, in 1-millisecond increments.
CallId	UInt32	The identifier of the call with which the information is associated.
Count	UInt32	The number of **CallInfo** structures in the variable body of this message. This is the number of Call Information elements that have been sent with the event.

Field	Data Type	Description
CallInfo	UInt32	A placeholder field that is replaced by the first CallInfo structure (in other words, by the CallInfo.Id field).

TSC_EvtCallState

Definition

TSC_EvtCallState is an asynchronous event that indicates a change in call state. It provides the basis for all call state management.

Used By Which TSC Components

Generic TSC, NetTSC

Additional Information

The TSC_EvtCallState event is typically used for the following things:

- to determine the outcome of a call-control operation on a call
- to provide the notification to the client of the arrival of a new call

Message Structure

TSC_EvtCallState contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. For more information, see *Table 10* on page 149.
Timestamp	UInt32	The board-level time at which the event was generated, in 4-millisecond increments.

Field	Data Type	Description
CallId	UInt32	The identifier of the call on which the call-state transition occurred.
CallState	UInt32	The new call state. For more information, see *Table 10* on page 149.
Reason	UInt32	The reason for the transition to the current call state, if any. For more information, see *Table 11* on page 152.

TSC_EvtChanState

Definition

TSC_EvtChanState is an asynchronous event that indicates a change in the channel state.

Used By Which TSC Components

Generic TSC only

Message Structure

TSC_EvtChanState contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC channel -state event.
Timestamp	UInt32	The board-level time at which the event was generated, in 4-millisecond increments.
ChanState	UInt32	The current channel state.

TSC_EvtTrace

Definition

TSC_EvtTrace is an asynchronous event that reports on a tracing condition.

Used By Which TSC Components

Generic TSC only

Additional Information

This event provides a mechanism to trace the call-control commands received from clients. When enabled, a TSC_EvtTrace event is generated each time a call-control command is received by the TSC instance.

Message Structure

TSC_EvtTrace contains these fields:

Field	Data Type	Description
Label	UInt32	The RTC label specified when this event type was enabled for detection with the *Std_MsgDetectEvt* message. For more information about the *Std_MsgDetectEvt* message, see *Appendix B* and the *DM3 Standard Messages and Run Time Control* guide.
Type	UInt32	The type of RTC call-state event. Values: • CallControl
Timestamp	UInt32	The board-level time at which the event was generated, in 4-millisecond increments.

Field	Data Type	Description
MsgType	UInt32	The message type that resulted in the event.
Body	UInt8	The message that generated the trace event. This is a variable field.

Appendix C

Call-Information Identifiers of the TSC and NetTSC Component

Table 6 lists the call-information identifiers used by the DM3 TSC and NetTSC components.

Table 6. TSC and NetTSC Component Call Identifiers

Information ID	Data Type	Description
CalledId	String	The ID of the original called party.
CallerId	String	The ID of the caller party.
RxPackets	UInt32	The total number of RTP packets received from the remote site during the phone call. This information is received from the RTCP protocol.
RxLostPackets	UInt32	The number of RTP packets that were lost during the phone call. This information is received from the RTCP protocol.
RxPacketsOutOfOrder	UInt32	The number of RTP packets received out of order from the remote site.
TxPackets	UInt32	The total number of RTP packets sent to the remote site during the call. This information is received from the RTCP protocol.

Information ID	Data Type	Description
TxLostPackets	UInt32	The number of transmitted RTP packets that were lost during the phone call. This information is received from the RTCP protocol.
RxJitter	UInt32	The difference in packet spacing at the remote site (the receiver), compared to the host (the sender), for a pair of packets that were sent form the sender to the receiver. This information is received from the RTCP protocol.
TxJitter	UInt32	The difference in packet spacing at the remote site (the receiver) compared to the host (the sender) for a pair of packets that were sent form the sender to the receiver. This information is received from the RTCP protocol.
RxCoder	UInt32	The coder type used to send the RTP packets.
TxCoder	UInt32	The coder type used to receive the RTP packets.
CallDurationTime	UInt32	The duration of the call, in seconds. Note: This duration time is computed according to the PrmCallDurationComput parameter, as described in *Table 7* on page 138.
TxAvgBitRate	TBD	The average bit rate of the RTP packets sent from the host to the remote site.
RxAvgBitRate	TBD	The average bit rate of the RTP packets received from the remote site.

Information ID	Data Type	Description
Origin	UInt32	The origin of the call. Values: • OUTBOUND=0 • INBOUND=1
Display	String	The display information received on this call. This information is received from the Q931 protocol.

Appendix D
Parameters of the TSC and NetTSC Component

Table 7 lists the parameters used by the DM3 TSC and NetTSC component.

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.

Table 7. TSC and NetTSC Component Parameters

Parameter Name	Data Type	Access	Level	Which Component	Description
Parm Bearer ChanId	UInt32	R/W	I	Generic TSC	The ID of the bearer channel with which the TSC instance is associated.
Parm Bearer ChanSet	UInt32	R/W	I	Generic TSC	Identifies the bearer-channel set in which the bearer channel is a member.
Parm LineId	UInt32	R/W	I	Generic TSC	The ID of the line device that the TSC instance is associated with, and to which the bearer channel belongs.
Parm Protocol Base	UInt32	R/W	I	Generic TSC, NetTSC	Defines the protocol family that the TSC instance implements. Values: • T1_CAS=1 • ISDN • H323

Parameter Name	Data Type	Access	Level	Which Component	Description
Parm Inbound VariantId	UInt32	R/W	I	Generic TSC	The ID of the protocol family that a TSC instance uses for inbound calls. This can be set to 0 to disable inbound calls.
Parm Outbound VariantId	UInt32	R/W	I	Generic TSC	The ID of the protocol family that a TSC instance uses for inbound calls. This can be set to 0 to disable outbound calls.
Parm Admin Group	UInt32	R/W	I	Generic TSC	The host configuration identifier assigned by the System Administrator. This provides a way of grouping TSC instances into arbitrary groupings.
ParmSlotId	UInt32	R/W	I	Generic TSC	The logical timeslot that the channel is using.

Parameter Name	Data Type	Access	Level	Which Component	Description
Parm Bearer ChanId	UInt32	R/W	C	Generic TSC	The level of debugging enables for a TSC instance. Default=0 (none).
Parm Encoding	UInt32	R/W	I	Generic TSC	The channel encoding scheme. Values: • Mulaw • Alaw

Appendix E

Key Value Sets (KVSets)

A Key Value Set (KVSet) is a variable-size field utilized by call-control messages. A KVSet field allows the client to append the coder type to be used in the call-control operation.

Coder

Definition

Coder is a KVSet structure.

Used By Which TSC Components

NetTSC only

Message Structure

Coder contains these fields:

Parameter Name	Data Type	Description
CoderType	UInt16	The type of coder. Values: • G7231=0 • GSM=1 • G711ML=2 • DONT_CARE=3
FrameSize	UInt16	The size of the frame, in milliseconds. Values: • 10 • 20 • 30
FramesPerPkt	UInt16	The number of frames per packet. Values: • 1 • 2 • 3 • 4

Parameter Name	Data Type	Description
CoderType	UInt16	The type of coder. Values: • G7231=0 • GSM=1 • G711ML=2 • DONT_CARE=3
Rate	UInt16	The frame rate. Values: • R6_3=0 • R5_3=1
VadEnable	UInt16	Values: • VAD_DISABLE=0 • VAD_ENABLE=1

RTCPInfo

Definition

RTCPInfo is a KVSet structure.

Used By Which TSC Components

NetTSC only

Message Structure

RTCPInfo contains these fields:

Parameter Name	Data Type	Description
CallId	UInt32	The ID of the call.
LocalSR_TimeStamp	UInt32	The time stamp of the RTCP packet transmission from the local sender.
LocalSR_TxPackets	UInt32	The number of packets sent by the local sender.
LocalSR_TxOctets	UInt32	The number of bytes sent by the local sender.
LocalSR_SendIndication	UInt32	The local sender report has changed since the last transmission. Values: • FALSE=0 • TRUE=1

Parameter Name	Data Type	Description
CallId	UInt32	The ID of the call.
LocalRR_FractionLost	UInt32	The percentage of packets lost, as computed by the local receiver.
LocalRR_CumulativeLost	UInt32	The number of packets lost, as computed by the local receiver.
LocalRR_SeqNumber	UInt32	The last sequence number received.
LocalRR_ValidInfo	UInt32	The RFU. Values: • FALSE=0 • TRUE=1
RemoteSR_TimeStamp	UInt32	The time stamp of the RTCP packet transmission from the remote sender.
RemoteSR_TxPackets	UInt32	The number of packets sent by the remote sender.
RemoteSR_TxOctets	UInt32	The number of bytes sent by the Remote Sender.
RemoteSR_SendIndication	UInt32	The remote sender report has changed since the last transmission. Values: • FALSE=0 • TRUE=1

Parameter Name	Data Type	Description
CallId	UInt32	The ID of the call.
RemoteRR_FractionLost	UInt32	The percentage of packets lost, as computed by the remote receiver.
RemoteRR_CumulativeLost	UInt32	The number of packets lost, as computed by the remote receiver.
RemoteRR_SeqNumber	UInt32	The last sequence number received.
RemoteRR_ValidInfo	UInt32	The RFU. Values: • FALSE=0 • TRUE=1

Appendix F

Attributes of the TSC and NetTSC Component, Clusters

Table 8 lists the attributes specific to the DM3 TSC and NetTSC component.

Table 8. TSC and NetTSC Component Standard Attributes

Attribute Name	Description	Supported by Which TSC
Std_ComponentId	Identifier that uniquely identifies the component type (e.g. the vendor that provides the component). Value: • TSC_Std_ComponentId=1 (for Dialogic)	Generic TSC, NetTSC
Std_ComponentType	The standard features of the TSP resource, which identify it as a DM3 resource. Value: • TSC_Std_ComponentType=0x12 (for generic TSC) • TSC_Std_ComponentType=0x1E (for NetTSC)	Generic TSC, NetTSC

Table 9. TSC and NetTSC Component Cluster Attributes

Attribute Name	Description	Supported by Which Component
AttrLineId	Indicates the TSP cluster's associated network line. Values: 1 through 4.	Generic TSC only
AttrChanId	Indicates the TSP cluster's associated bearer channel. The bearer channel is relative to the line ID. Values: 1 through 31.	Generic TSC only
AttrAdminGroup	The Administration group assigned to the channel during configuration. This ID is set via the AdminGroup field of the Bearer Channel Set definition message, and also via SetParm.	Generic TSC only
AttrProtocolBase	The base protocol the cluster is executing. Values: • TSC_ParmProtocolBase_T1_CAS • TSC_ParmProtocolBase_ISDN • TSC_ParmProtocolBase_H323	Generic TSC, NetTSC

Appendix G

TSC and NetTSC Call States and Changes in Call States

Table 10 lists the possible states for a given call.

The call state is stored in the CallState field, which is present in two messages:

- a *Std_MsgEvtDetected* message for a TSC_EvtCallState event
- *TSC_MsgGetCallStateCmplt*

Table 10. TSC and NetTSC Call States

Call State	Description	Used by Which TSC Component
Accepted	An offered call has been accepted by a client. The client has accepted responsibility in answering the call. For certain services, such as ISDN, this also indicates that ringback has been initiated to the calling party.	Generic TSC, NetTSC
Alerting	An alert is being sent to the network (such as generating a ringback tone, or sent alerting IE).	Generic TSC, NetTSC
Connected	The calling and called parties are connected, and the call is active on the related call channel. Information may be exchanged. In the case of an outbound call, this state indicates that the remote party has answered. In the case of an inbound call, this state indicates that the local party has answered the call.	Generic TSC, NetTSC

Call State	**Description**	**Used by Which TSC Component**
ConnectedPendXfer	The call is connected to the local party, but is pending on the completion of the transfer operation.	Generic TSC only
Delivered	The outbound call has been delivered to the remote party (e.g., ringback has been detected).	Generic TSC only
Dialing	The outbound call setup information relative to the call channel is being transferred via the network.	Generic TSC only
Disconnected	The remote party has disconnected from the call.	Generic TSC, NetTSC
DialReady	The network is ready to receive outbound call setup information relative to the call channel. (e.g., the call channel is off-hook and dial-tone has been received).	Generic TSC only
Failed	The outbound call attempt was unsuccessful. The call attempt will transition to this state if it is determined to be unsuccessful due to conditions such as detection of a busy signal, SIT, network protocol error, etc.	Generic TSC, NetTSC
Idle	The local party to the call has disconnected or has been disconnected.	Generic TSC, NetTSC
Initiated	The outbound call attempt has been initiated (*TSC_MsgMakeCall*).	Generic TSC, NetTSC

Call State	Description	Used by Which TSC Component
Offered	The inbound call is newly arrived and is being offered to the client. Call information is available to the client at this time in order for the client to determine the appropriate action to take with regards to the call.	Generic TSC, NetTSC
Hold	The local party has placed the call on hold.	Generic TSC only
HoldPendXfer	The local party has placed the call on hold pending the completion of a transfer to another destination.	Generic TSC only
Null	The call has been released. Call information for the call is no longer available. The call identifier is now invalid.	Generic TSC, NetTSC
Originated	The exchange of outbound call setup information via the network relative to the call channel has been completed. At this point, call progress can be monitored.	Generic TSC only
Proceeding	Part of the ISDN call setup.	Generic TSC, NetTSC
Unknown	The state of the call is not known. This may be due to limitations in the protocol implementation, and/or in the call progress implementation.	Generic TSC, NetTSC

Table 11 lists the reasons for a change in call state. These reasons are stored in the **Reasons** field, which is present in two messages:

- a *Std_MsgEvtDetected* message for a TSC_EvtCallState event
- *TSC_MsgGetCallStateCmplt*

Table 11. Reasons for Call-State Changes

Name of Call State	Reason for Change in Call State	Typical Call States
Busy	Call State change induced by Busy condition.	Failed, Idle
CallCompletion	Call State change occurred due to a call completion request.	Offered, Idle
Canceled	Call State change caused by cancellation.	Failed, Idle (prior to an expected Connected state)
Congestion	Call State induced due to congestion of the network or switch.	Failed, Idle
DestBusy	Indicates that the Call State changed due to a busy condition at the destination address.	Failed, Idle
DestAddrBad	Call State caused by an invalid destination address.	Failed, Idle
DestOutOfOrder	Call State changed due to the destination address being out of order.	Failed, Idle
DestUnobtainable	Call State changed due to the destination address being unobtainable.	Failed, Idle
Forward	Call State change occurred due to the call being forwarded to another address.	Offered, Idle
Incompatible	Change occurred due to an incompatibility condition.	Failed, Idle

Name of Call State	Reason for Change in Call State	Typical Call States
IncomingCall	Change occurred due to an incoming call (i.e. a glare condition).	Failed
NewCall	Indicates that the Call State change represents a new call.	Offered
NoAnswer	Call State change is the result of a no-answer time-out condition.	Failed, Idle
Normal	Indicates that the state transition occurred due to normal conditions.	All states
NetworkAlarm	Call State changed due to a network alarm condition.	Disconnected, Failed, Idle
PickUp	Call State change occurred due to the call being picked up either by a remote address, or by the local address.	Connected, Failed, Idle
ProtocolError	Call State change occurred due to a protocol error.	Disconnected, Failed, Idle
Redirection	Call State change occurred due to the call being redirected.	Offered, Disconnected, Idle
RemoteTermination	Call State change occurred due to remote call termination (i.e. the remote party dropped the call).	Offered, Disconnected, Idle
Rejection	Call state change occurred due to the call being rejected for unknown reasons.	Failed, Idle
SIT	Call State changed due to the receipt of a special information tone (SIT) of unknown detail.	Failed, Idle

Name of Call State	Reason for Change in Call State	Typical Call States
SITCustIrreg	Call State changed due to the receipt of a special information tone (SIT) of the customer irregularity type. This precedes a vacant number, AIS, Centrex number change, non-working station, access code not dialed or dialed in error, or manual intercept operator message.	Failed, Idle
SITNoCircuit	Call State changed due to the receipt of a special information tone (SIT) of the no-circuit type. This typically indicates a trunk blockage irregularity and is followed by a no-circuit or emergency announcement.	Failed, Idle
SITReorder	Call State changed due to the receipt of a special information tone (SIT) of the reorder type. This typically indicates an equipment irregularity and is typically followed by a reorder announcement.	Failed, Idle
Transfer	Call State change occurred due to the call being transferred.	Offered, Disconnected, Idle
Unavailable	No further information available.	Any state
Unknown	No further information known.	Any state
UnallocatedNumber	Destination is an unallocated or unassigned number.	Failed
NoRoute	No route to destination.	Failed

Name of Call State	**Reason for Change in Call State**	**Typical Call States**
NumberChanged	Destination number has changed.	Failed
OutOfOrder	Destination is out of order.	Failed
InvalidFormat	Invalid number format (not enough digits).	Failed
ChanUnacceptable	Requested channel is unacceptable.	Failed
ChanNotImplemented	Channel type not implemented.	Failed
ChanUnavailable	Requested channel is unavailable	Failed
NoChan	No circuits or channels available.	Failed
NoResponse	No user is responding.	Failed
FacilityNotSubscribed	Requested facility is not subscribed.	Failed
FacilityNotImplemented	Requested facility is not implemented.	Failed
ServiceNotImplemented	Bearer service is not implemented.	Failed
BarredInbound	Inbound calls are barred.	Failed
BarredOutbound	Outbound calls are barred.	Failed
DestIncompatible	Destination is incompatible.	Failed
BearerCapUnavailable	Bearer capability is not presently available.	Failed

Index

NOTES

NOTES