SISO-STD-015-2016

Standard for Distributed Debrief Control Architecture

Version 1.0

11/10/2016

Prepared by
Distributed Debrief Control Architecture
Product Development Group

P.O. Box 781238
Orlando, FL 32878-1238, USA

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA
.

Revision History

Version	Section	Date (MM/DD/YYYY)	Description
1.0	All	08/19/2016	Published Version (11/10/2016)

Participants

At the time this product was submitted to the Standards Activity Committee (SAC) for approval, the Distributed Debrief Control Architecture Product Development Group had the following membership and was assigned the following SAC Technical Area Director:

Product Development Group

Michael France (Chair)
Randy Pitz (Vice-Chair / Secretary)

SAC Representative Jeff Abbott (SAC Technical Activity Director)

Len Granowetter
Lance Marrou
Brett Tainter
Eric Watz

The Product Development Group would like to especially acknowledge those individuals that significantly contributed to the preparation of this product as follows:

PDG Drafting Group

Eric Watz (Editor)

Michael France
John Jinkerson
Randy Pitz
Brett Tainter

The following individuals comprised the ballot group for this product.

Ballot Group

Curt Blais
Larry Boyce
John Brennan
Philip Cofield
Patrice Le Leydour
William Oates
Randy Pitz
David Ronnfeldt
Graham Shanks
Brett Tainter
Steven Weiss
…..

When the Standards Activity Committee approved this product on 16 September 2016, it had the following membership:

Standards Activity Committee

Jeff Abbott (Chair)
Marcy Stutzman (Vice Chair / Secretary)

Curt Blais
Kevin Gupton
Jean-Louis Igarza
Lance Marrou
Thom McLean
William Oates
Simone Youngblood
Robert Lutz
Grant Bailey
Peggy Gravitz
Lana McGlynn

When the Executive Committee approved this product on 11/10/2016, it had the following membership:

Executive Committee

Michael O'Connor (Chair)
Robert Lutz (Vice Chair)
Jeff Abbott (Secretary)

James Coolahan
John Daly
John Diem
Lana McGlynn
Katherine Morse
Roy Scrudder
Robert Siegfried
Eric Whittington

Introduction

Exercise replay is a common element of debriefing systems. Many tool vendors implement exercise replay using a variety of techniques and possess unique capabilities meeting the specific needs of the program. When these systems need to be integrated into a larger distributed debrief spanning multiple sites, problems arise due to a lack of an interoperable standard. The distributed debrief in this case may not leverage the rich capabilities each site has to offer, and only provide the bare minimum capability. Existing debriefing capabilities can be maintained and enhanced while reducing integration and operating costs through the development of an interoperable protocol for distributed debrief.

This standard specifies the control aspects of a distributed debrief. This standard does not include specifications for the actual playback of data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Standard for Distributed Debrief Control Architecture

1 Overview

1.1 Scope

This standard defines the Distributed Debrief Control Architecture (DDCA) in terms of an object model, messages, and state definitions.

1.2 Purpose

DDCA is an object model that is designed to specify the states and behaviors required for multiple discrete debrief systems to interoperate during a distributed debrief (DD) event. Interoperability between different implementations shall be enabled through consistent use of these messages, states and behaviors.

1.3 Objectives

This standard defines an architecture for distributed debriefing that promotes interoperability while supporting efforts aimed at reducing integration and operating costs.

1.4 Intended Audience

The audience for this document is the portion of the Modeling and Simulation (M&S) community who conduct distributed debrief activities, including instructor operator station developers and others who are interested in or are involved in distributed debrief. Other communities of interest, although not the intended primary audience, are encouraged to leverage the standard described here for use in their domains.

2 References (Normative)

2.1 SISO References

#	Document Number	Title	Date
1.	SISO-ADM-002-2011	SISO Policies and Procedures	11 Apr 2011
2.	SISO-ADM-003-2011	SISO Balloted Products Development and Support Process	14 Nov 2011
3.	SISO-ADM-005-2011	Policy for the Style and Format of SISO documents	13 Jun 2011
4.	SISO-REF-028-2011	Final Report for the Distributed Debrief Control Protocol Study Group	16 Feb 2009

2.2 Other References

#	Document Number	Title	Date
1.	n/a	X.667 : Information technology - Procedures for the operation of object identifier registration authorities: Generation of universally unique identifiers and their use in object identifiers https://www.itu.int/rec/T-REC-X.667-201210-I/en	03 Mar 2014

3 Definitions, Acronyms, and Abbreviations

3.1 Definitions

Term	Definition
Data Time	The wall-clock Zulu time at which data were generated. Represents the "origin" time at which any given piece of data or information was generated and subsequently recorded.
DDCA Client	The DDCA Client application "listens" to a DDCA Master application, and executes the Control messages issued from the Master application. Client applications do not control the debrief event.
DDCA Master	The DDCA Master application has "control" of the debrief session, meaning it is the application responsible for issuing Control messages such as Start, Stop, Play, etc. to any joined Client application(s).
DDCA Message	DDCA Messages represent the individual messages exchanged between a Master application and a Client application. Throughout the remainder of this document, DDCA Messages will be represented with the BOLD CamelCase style, where each word begins with a capital letter, e.g., **MessageName**.
DDCA State	DDCA States, hereby referred to as "States" throughout the remainder of this document, are the particular state at which a DDCA Client application exists at any one time during a distributed debrief event. Throughout the remainder of this document, DDCA States will be represented with the BOLD ALL CAPS style, e.g., **STATENAME**.
Debrief	Describes instances where a log file or recorded data is played back for the purpose of evaluating a training mission.
Debrief Time	The wall-clock Zulu time during which a distributed debrief event is occurring. Represents the current wall-clock Zulu time of the debrief.
Distributed Debrief	Describes instances where a debrief event is conducted with multiple participants at discrete locations.
Replay	Describes instances where a log file or recorded data is played back for any technical or engineering-related purpose(s).
Time Synchronization	The act of setting multiple system clocks to a common value.

3.2 Acronyms and Abbreviations

Acronym or Abbreviation	Meaning
DD	Distributed Debrief
DDCA	Distributed Debrief Control Architecture
NTP	Network Time Protocol
UUID	Universally Unique Identifier

4 Distributed Debrief Control Architecture

4.1 DDCA Concepts

DDCA is designed as a system-agnostic state diagram that represents the concept of operations for conducting a distributed debriefing event. DDCA operates on the principle that a single application, known as the DDCA Master application, has "control" of the replay session while one or more Client applications are listening; meaning, the Master application is responsible for sending out Control messages to all Client application(s) and instructing the Client application(s) in such tasks as: the replay time window to load, exactly where in the replay time window to point, and when to play, stop playing, etc.

A single application shall be designated as the DDCA Master application. Any number of Client applications may join into or leave a DD event at any time. Multiple replay time windows, which may be contained in separate physical recording files (a.k.a. log files), may be replayed during a single DD event.

DDCA represents the desired end-states of DDCA Client applications and the Master application during and throughout a distributed debrief event. DDCA is designed as an object model, complete with associated state diagrams, in order to:

- Maximize adoption of the standard
- Allow for adoption of DDCA using any protocol
- Illustrate the expected behavior of DDCA Master and Client applications

The DDCA consists of an object model that defines:

- Messages used in DDCA
- States of Master and Client applications
- Behaviors when transitioning between various states
- Interoperability between DDCA implementations across the M&S architectures is enabled through consistent use of these messages, states and behaviors.

4.2 DDCA Message Types

Within the DDCA there are three types of messages:

- Control messages: sent from a Master application to one or more Client applications.
- Request messages: sent to a DDCA application (Master or Client) when the requester requires information from the application. Master and Client applications may send Request messages.
- Response messages: sent from requestee to requester in response to a Request message.

The DDCA includes two types of applications: Master and Client applications.

4.3 DDCA Master Application

In DDCA, the Master is the “main”, or controlling, application. The Master application is responsible for all control messages sent during a DDCA session. Master applications within DDCA:

- Shall not respond to Control messages.
- Shall send Request messages to Client applications.
- Shall respond to Request messages sent from Client applications.
- Shall send Control messages to Client applications.

4.4 DDCA Client Application

A DDCA Client application is a “listener” for Control messages from the DDCA Master application. Client applications within DDCA:

- Shall not send Control messages.
- Shall respond to Request and/or Control messages from the Master application.
- Shall not respond to Request messages from other Client applications.
- May send request messages to the Master application.

4.5 DDCA Capabilities

The capabilities of DDCA are illustrated in Figure 1. This set of DDCA capabilities are considered to be essential to the core functionality of the Distributed Debrief Control Architecture.

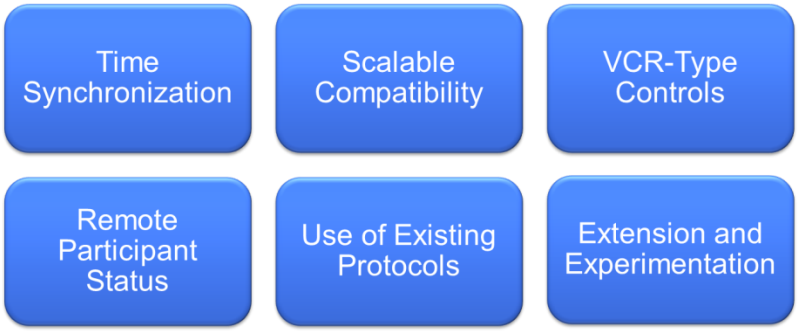

Figure 1 DDCA Core Capabilities

4.5.1 Time Synchronization

The DDCA shall rely on external implementations such as NTP servers to provide a means of synchronizing individual system clocks across all participants in an exercise. Time synchronization is a critical enabling feature for playback during a distributed debrief event. Time synchronization is the ability of multiple discrete systems to know the true, common time during playback. Time synchronization is also the capability of managing a common timeline for playback across all distributed systems. Prior to, and during the execution of, a DD event, all DDCA Client applications and the Master application shall have their individual system clocks synchronized to a single external source.

The ability to precisely synchronize system clocks is paramount to the time synchronization capability of DDCA. All DDCA Client application(s) in a DD event shall be synchronized in their playback, to the time sent out by the Master application in its Sync message.

4.5.2 Scalable Compatibility

Indicates the level of compatibility the Client application may participate in a DD event.

4.5.3 VCR-Type Controls

Replay control is another critical enabling feature for distributed debrief. This is the ability to control the playback of recorded simulation data using one of the available sub-states. DDCA Messages covering this functionality are discussed later in this document.

4.5.4 Remote Participant Status

DDCA shall include the ability for a Master application to query the status of all Client applications. At any time during a DD event, the Master application may issue queries to one or more Client applications that are participating in the DD event; the Client application shall provide its current status to the Master application.

4.5.5 Use of Existing Protocols

The DDCA standard is an architecture, and therefore does not specify a particular mechanism to be used in the sending and receiving of DDCA messages.

DDCA messages may be transported over any supported protocol. The specific method used to transport and receive a DDCA message is independent of the DDCA itself; applications can receive and process DDCA messages received by any means.

4.5.6 Extension and Experimentation

The ability to try new features within a standard is paramount to extending the standard's capabilities. DDCA will include a DDCA-X message, which may be used to extend the current capabilities by adding new message types and/or features within a DD event.

4.6 DDCA States

The DDCA state diagram is illustrated at a high level in Figure 2 and illustrates the possible Client application states during a DDCA event. DDCA Client application states are described in Table 1. DDCA state transitions are individually numbered and are described in Table 2.

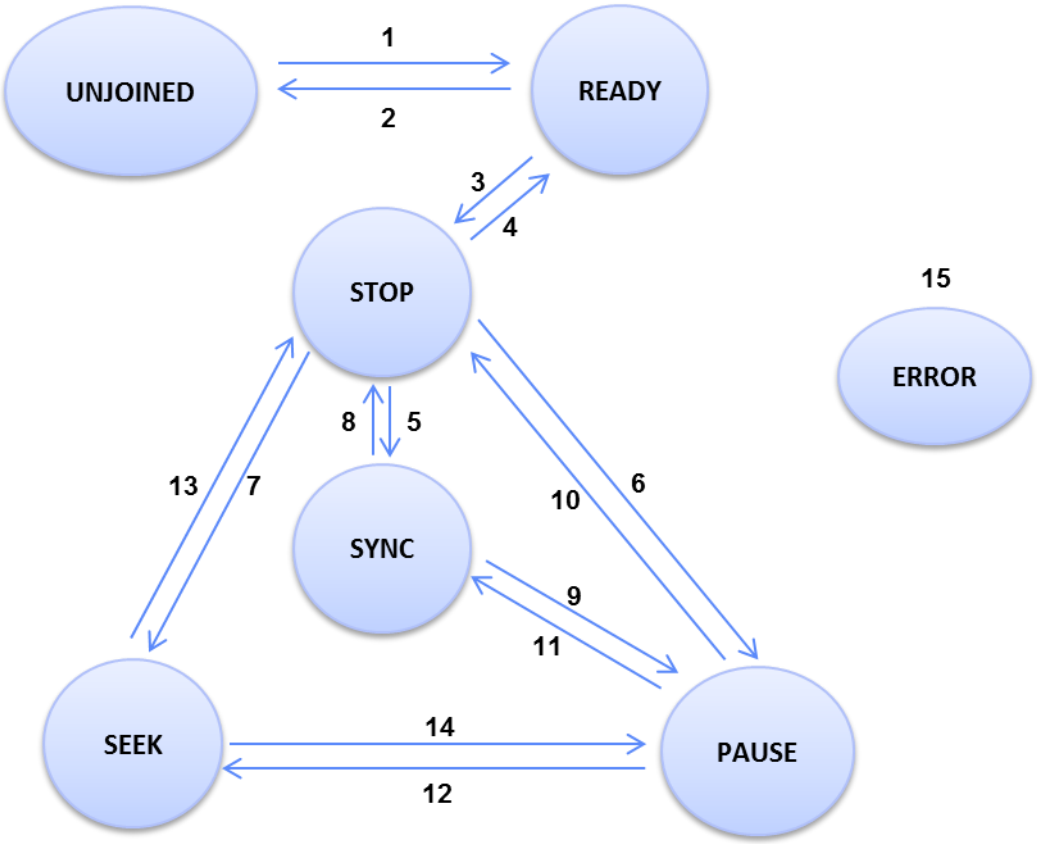

Figure 2 DDCA State Diagram

Table 1 DDCA States

State	Value	Description
UNJOINED	0	In this state, Client application(s) are not participating in a DDCA event, and are free to join any DDCA event.
READY	1	Client application(s) transition to this state after having joined a DDCA event, and await Control message(s) from the DDCA Master application. Once a Client application is in the **READY** state, the Client application shall not respond to Control messages from any DDCA Master application other than the application which is controlling the DDCA event to which the Client application has joined
STOP	2	Sent by: DDCA Master Application All Client application(s) shall cease the current operation and shall await further Control message(s) from the Master application. The **STOP** state indicates that a replay file is loaded but not yet playing. The Client applications' index/offset/file pointer into the replay file should be at position 0.
SEEK	3	Sent by: DDCA Master Application All Client application(s) shall seek to the time specified by the Master application.
SYNC	4	Sent by: DDCA Master Application Contains sub-states: **PLAY**, **FAST FORWARD**, and **REWIND** The **SYNC** state contains the following possible sub-state values: **PLAY**, **FAST FORWARD**, and **REWIND**. The **PLAY** sub-state is a forward playback at a speed of 1x. The **FAST FORWARD** sub-state is a forward playback at an integral speed other than 1x. The **REWIND** sub-state is playback in reverse at a negative integral speed value including, but not limited to, 1x. All Client application(s) shall match their internal Data Time and Speed with the Data Time and Speed values contained in the most recent **Sync** message received from the Master application. If a Client does not support a specific sub-state of Sync, that Client shall remain in their current state until the Sync message indicates a state that the Client supports, at which point the Client shall continue processing Sync messages.
PAUSE	5	All Client application(s) shall pause the current operation. No replay data is sent from Client application(s) in this state.

Table 2 DDCA State Transitions

Transition	Begin State	End State	Trigger(s)
1	**UNJOINED**	**READY**	Either: 1) Master application sends a **Join** message to Client(s), followed by Client application(s) Acknowledge with status of **READY**, OR 2) Client application sends a **JoinRequest** message to Master application, followed by Master application sending a **JoinResponse** message with a Response = Accept Client application (s) transition to **READY**.
2	**READY**	**UNJOINED**	Either: Master application sends a **Leave** message to Client application(s), OR Client application sends a **LeaveRequest** message to Master application, followed by Master application sending **LeaveResponse** message with Response = Accept Client application(s) transition to **UNJOINED**.
3	**READY**	**STOP**	Master application sends a **Load** message to Client application(s), followed by Client application(s) **Acknowledge** with status of **STOP**. Client application(s) transition to **STOP**.
4	**STOP**	**READY**	Master application sends a **Close** message to Client application(s), followed by Client application(s) **Acknowledge** with status of **READY**. Client application(s) transition to **READY**.
5	**STOP**	**SYNC**	Client application(s) currently in **STOP** state, and receive a **Sync** message with a purpose of Play, a DataTime at which to start playing, and the Speed at which to play (may be greater than 1x, equal to 1x, or less than 1x). Client application(s) transition to **SYNC**.
6	**STOP**	**PAUSE**	Client application(s) currently in **STOP** state, and receive a **Sync** message with a purpose of Pause, a Speed of 0, and the DataTime at which the Pause operation occurs. Client application(s) transition to **PAUSE**.
7	**STOP**	**SEEK**	Client application(s) currently in **STOP** state, and receive a **Sync** message with a purpose of Seek, a Speed of 0, and the DataTime to seek to. Client application(s) transition to **SEEK**.
8	**SYNC**	**STOP**	Client application(s) currently in **SYNC** state, and receive a **Stop** message from Master application. Client application(s) transition to **STOP**.

9	**SYNC**	**PAUSE**	Client application(s) currently in **SYNC**-Play state, and receive a **Sync** message with a purpose of Pause, a Speed of 0, and the DataTime at which the Pause operation occurs. Client application(s) transition to **PAUSE**.
10	**PAUSE**	**STOP**	Client application(s) currently in **PAUSE** state, and receive a **Stop** message from Master. Client application(s) transition to **STOP**.
11	**PAUSE**	**SYNC**	Client application(s) currently in **PAUSE** state, and receive a **Sync** message with a purpose of Play, a DataTime at which to start playing, and the Speed at which to play (may be greater than 1x, equal to 1x, or less than 1x). Client application(s) transition to **SYNC**.
12	**PAUSE**	**SEEK**	Client application(s) currently in **PAUSE** state, and receive a **Sync** message with a purpose of Seek, a Speed of 0, and the DataTime to seek to. Client application(s) transition to **SEEK**.
13	**SEEK**	**STOP**	Client application(s) currently in **SEEK** state, and receive a **Stop** message from the Master application. Client application(s) transition to **STOP**.
14	**SEEK**	**PAUSE**	Client application(s) currently in **SYNC** state, and receive a **Sync** message with a purpose of Pause, a Speed of 0, and the DataTime at which the Pause operation occurs. Client application(s) transition to **PAUSE**.
15	**ANY**	**ERROR**	Client application(s) experience(s) an application error. May be entered from any DDCA state. Client application(s) may transition to **ERROR** at any time. From the **ERROR** state, a Client application shall, upon resolving the error condition, enter the **UNJOINED** state. Client application(s) transition to **ERROR**.

5 DDCA Message Header

All DDCA messages shall incorporate a common message header with the following information:

Table 3 DDCA message header

Field Name	Data Type	Description
SessionID	UUID	UUID of the DDCA Session ID. A UUID is an identifier standard used in software construction, standardized by the Open Software Foundation. A UUID is a 16-octet (128-bit) number. In its canonical form, a UUID is represented by 32 hexadecimal digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). For example: 550e8400-e29b-41d4-a716-446655440000 See Section 2.2, Other References, #1 for additional information on UUIDs. The Session ID is used to uniquely identify a DDCA distributed debrief event
SenderID	UUID	UUID of the sending application. DDCA uses Universally Unique Identifier (UUIDs) to uniquely identify the Master and Client applications in a DDCA event. Prior to joining a DDCA event, all Client application(s) shall generate a UUID value. For all communications between Master and Client applications, the Master application shall refer to the Client application using the Client application's UUID.
Receiver ID	UUID	UUID of the receiving application. DDCA Client applications shall use this field to determine if a DDCA Command message is intended for their Client application, intended for another Client application, or intended for all Client applications. If the Receiver ID field contains all zero values, all DDCA Client applications shall process the DDCA Command message. If the Receiver ID field contains non-zero values, the following rules apply to all Client applications: - DDCA Command messages with a Client ID matching that of a DDCA Client application shall be processed by the Client application. - Client application(s) whose ID does not match the Receiver ID shall not process the Command message.
Message Type	Unsigned 8-bit integer	Enumeration of message types (some protocols/interfaces may require this in order to process)
Message Size	Unsigned 16-bit integer	Length of the message (some protocols/interfaces may require this in order to process), in bytes. This value represents the length of the message, including the message header.
Message ID	Unsigned 32-bit integer	Unique (by sender) message id. The sender of the message shall maintain a unique message ID value, and shall increment this value by 1 for each message it sends.
Debrief Time	Unsigned 64-bit integer	Wall clock Zulu time of the DDCA message creation, in milliseconds since January 01, 1970. May be used by Client applications to determine message ordering or to compute network latency.

6 DDCA Master Messages

Table 3 details messages sent from, and received by, a DDCA Master application.

Table 4 DDCA Master messages

Message Name	Description
Load	Instructs Client application(s) to load a recording for replay.
Close	Instructs Client application(s) to close the file(s) being replayed.
Sync	Instructs Client application(s) to perform Play, Stop, Pause, and Seek operations. **Sync** will be sent out at periodic intervals.
Join	Instructs that a particular Client or Client application(s) join a DDCA event.
Leave	Instructs that a particular Client or Client application(s) leave a DDCA event.
List	Instructs Client application(s) to list all available recordings.
Request Status	Requests status of Client application(s) on the network.
DDCACompatibility	Requests Client application(s) to report compatibility level for DDCA interoperability.
LeaveResponse	Process a Client application's request to leave a DDCA event.
JoinResponse	Process a Client application's request to join a DDCA event.
DDCA-X	A generic message structure that supports extensible content.

6.1 Load

The **Load** Control message is used to load recordings for playback. Recordings are identified by Session ID. The **Load** Control message can only be processed from a Client application's **READY** state and enables the Client application's entry into the **STOP** state.

When a Client application receives a Load Control message, it shall immediately send an Acknowledge message with an Acknowledge Status of "WaitingToComply". The Client shall then attempt to load any recording(s) that match the DataTime window in the Load Control message. If no recordings are found that match the DataTime window, the Client remains in the Ready state. Following the attempt to load its recording(s), the Client application shall send a LoadResponse message indicating that it has completed processing of the Load Control message.

DDCA shall use the StartDataTime and EndDataTime fields in the **Load** Control message. These fields represent a window of time that is desired for playback. The window of time to be searched by DDCA Client applications corresponds to the wall-clock Zulu time at which the data were originally generated. DDCA Client applications are responsible for searching their inventory of recordings, and replaying the data that corresponds to the time window.

Table 5 Load Message

Field Name	Data Type	Description
StartDataTime	Unsigned 64-bit integer	DataTime corresponding to the beginning of the playback window. DataTime field represents the date/time at which the data was created (in milliseconds since January 01, 1970).
EndDataTime	Unsigned 64-bit integer	DataTime corresponding to the end of the playback window. DataTime field represents the date/time at which the data was created (in milliseconds since January 01, 1970).

6.2 Close

The **Close** Control message instructs Client application(s) to free or release any resources such as files, videos, etc. in use as a result of a previous **Load** control message. Upon receipt of the **Close** Control message, a Client application shall cease any operations currently in progress, and shall unload any file(s) that are currently loaded and transition into the **READY** state.

No additional fields, outside of the DDCA message header, are defined for the DDCA **Close** message.

6.3 Sync

The **Sync** Control message enables Play, Stop, Pause, and Seek operations by way of an enumerated type variable. The current DataTime of the DD event, and the speed of the sync are also passed with this Control message. The Purpose field within the **Sync** Control message specifies the desired state for all Client applications and its values are described further in Table 6.

The DDCA Master application shall issue periodic **Sync** Control messages. The frequency of issuance for periodic Sync messages shall be such that a Client application(s) and the Master application maintain in synch their respective Debrief Time as well as Data Time values.

Table 6 Sync message

Field Name	Data Type	Description
Purpose	Unsigned 32-bit integer	Purpose of the sync operation: Play, Stop, Pause, or Seek.
CurrentTime	Unsigned 64-bit integer	DataTime corresponding to a specific data time within the playback window. DataTime represents the date/time at which the data was created (in milliseconds since January 01, 1970).
Speed	8-bit integer	The requested playback speed

Table 7 Sync Operation Purpose values

State	Value	Description
Play	0	Replays data from the playback window at the requested playback speed
Stop	1	Stops playback of current recording. Causes Client applications to perform an internal “jumpTo” operation that results in their internal file pointers being reset to the first data in the playback window
Pause	2	Pauses playback of current recording. Does not perform a “jumpTo” operation
Seek	3	Seeks to a specified time within the playback window

6.4 Join

The **Join** Control message enables the Master application to instruct a Client application to join a DDCA event by referencing its ClientID. A Client application that accepts a **Join** Control message transitions from the **UNJOINED** state to the **READY** state.

Table 8 Join message

Field Name	Data Type	Description
ClientID	UUID	Unique identifier of the Client application

6.5 Leave

The **Leave** Control message enables the Master application to instruct a Client application to leave the DD event by referencing its Client ID. Upon receipt of a **Leave** Control message, a Client application transitions to the **UNJOINED** state.

Table 9 Leave message

Field Name	Data Type	Description
ClientID	UUID	Unique identifier of the Client application

6.6 List

The **List** Request message requests all Client applications to list their available recordings. The **List** Control message can occur from the **READY** state and keeps the Client in the **READY** state.

The **List** Request message contains two parameters, StartDataTime and EndDataTime, which together represent the window of time to which the Client application should constrain its search for available recordings. If the Master application desires that its Client application(s) return a list of all available recordings, the StartDataTime and EndDataTime fields within the **List** Control message should be set to 0.

Table 10 List Message

Field Name	Data Type	Description
StartDataTime	Unsigned 64-bit integer	DataTime corresponding to the beginning of the search window. DataTime field represents the date/time at which the data was created (in milliseconds since January 01, 1970).
EndDataTime	Unsigned 64-bit integer	DataTime corresponding to the end of the search window. DataTime field represents the date/time at which the data was created (in milliseconds since January 01, 1970).
SearchType	Unsigned 8-bit integer	Specifies the search type

The SearchType field indicates what recordings are to be searched and returned by the Client application. The field shall be one of the values in Table 11.

Table 11 Search type values

Value	Description
1	All available recordings. Client application shall list all available recordings
2	Starts at, or later than Client application shall list all recordings that start at, or later than, the StartDataTime parameter.
3	Starts at, or earlier than Client application shall list all recordings that start at, or earlier than, the StartDataTime parameter.
4	Ends at, or later than Client application shall list all recordings that end at, or later than, then EndDataTime parameter.
5	Ends at, or earlier than Client application shall list all recordings that end at, or earlier than, the EndDataTime parameter
6	Inclusive Client application shall list all recordings that start at, or later than the StartDataTime parameter AND that end at, or earlier than the EndDataTime parameter

6.7 Request Status

The **Request Status** Request message enables the Master application to request the status of Client application(s). The **Request Status** Request message may occur in any state, and enables no state transitions. Client applications that receive a **Request Status** message should respond to the message sender with their current status.

Table 12 Request status message

Field Name	Data Type	Description
MasterID	UUID	Unique identifier of the Master application

6.8 DDCA Compatibility

The **DDCACompatibility** Request message is issued by the Master application to request a Client application to report its compatibility level for DDCA interoperability. Levels of DDCA compatibility are illustrated in Table 14. This field exists to support growth and expansion of the DDCA standard.

Table 13 DDCA compatibility message

Field Name	Data Type	Description
CompatibilityLevel	Unsigned 8-bit integer	Enumerated value representing DDCA compatibility level

Table 14 DDCA compatibility levels

Compatibility Level	Value	Description
Standard	1	Client application supports all core DDCA functions.

6.9 Leave Response

The **LeaveResponse** Response message is issued by the Master application, in response to the Master application receiving a **LeaveRequest** message from a Client application. The Join/Leave responses are illustrated in Table 17.

Table 15 Leave response message

Field Name	Data Type	Description
MasterID	UUID	Unique identifier of the Master application
Response	Unsigned 8-bit integer	Enumerated leave status response
ClientID	UUID	Unique identifier of the participant

6.10 Join Response

The **JoinResponse** Response message is issued by the Master application, in response to the Master application receiving a **Join** request from a Client application. Join/Leave responses are illustrated in Table 17.

Table 16 Join response message

Field Name	Data Type	Description
MasterID	UUID	Unique identifier of the Master application
Response	Unsigned 8-bit integer	Enumerated join status response
ClientID	UUID	Unique identifier of the participant

Table 17 illustrates the supported values for Join / Leave responses.

Table 17 Join/Leave response

Join/Leave Response	Value	Description
Accept	1	Client application request to join/leave a DDCA event is accepted.
Reject	2	Client application request to join/leave a DDCA event is rejected.
Error	3	An unknown error occurred.

6.11 DDCA-X

The **DDCA-X** message can be sent by either a Master or a Client application. **DDCA-X** messages are designed to provide a variable payload section, similar to that of a DIS Set Data Pdu. The variable payload section may contain any content as specified by the sender of the message. Senders and recipients of **DDCA-X** message(s) must have a shared understanding of the **DDCA-X** payload prior to using the message.

The DataID, DataLength, and Data fields may be repeated as necessary within the **DDCA-X** message. If repeated, all fields {id, length, and data} shall be included. The complete length of the **DDCA-X** message, including all payload contents, shall be reflected in the MessageSize field within the message header.

Table 18 DDCA-X message

Field Name	Data Type	Description
DataID	Unsigned 32-bit integer	Identifier for the data field
DataLength	Unsigned 16-bit integer	Length of the data field, in bytes.
Data	Unsigned 8-bit integer array	Contents of the data field

7 DDCA Client Application Messages

Table 19 details messages sent from, and received by, a DDCA Client application.

Table 19 DDCA Client messages

Message Name	Description
Error	Sent anytime a Client application experiences an error condition
JoinRequest	Request to join a DDCA event
LeaveRequest	Request to leave a DDCA event
Acknowledge	Sent in response to a number of messages from the DDCA Master application
ListResponse	Lists the Client application's available recording sessions
LoadResponse	Sent when a Client application has completed processing of a Load Control message.
StatusResponse	Indicates the Client application's current Status
DDCACompatibility Response	Indicates the Client application's DDCA compatibility level
DDCA-X	A generic message structure that supports extensible content (Reference section 6.11 for a description of DDCA-X messages).

7.1 Error

The Error message is sent anytime a Client application experiences an error condition. Information in this message includes error code, error level, and a description of the error.

Table 20 Error message

Field Name	Data Type	Description
ErrorCode	Unsigned 32-bit integer	Client-defined error code
ErrorLevel	Unsigned 8-bit integer	Level of error. See Table 21 for error descriptions.
DataLength	Unsigned 16-bit integer	Length of error data
ErrorData	Character array	Description of the error condition

DDCA shall support error levels as described in Table 21. Client applications shall determine the appropriate error level for a given message; i.e., avoid using Error level for engineering-level debugging messages.

Table 21 Error levels

Error Level	Value	Description
Verbose	1	May contain any content.
Information	2	Contains useful information when the Client application is functioning normally.
Warning	3	Indicates something is wrong, and the Client application is able to continue functioning in a normal or degraded state.
Error	4	Indicates something is wrong, and the Client application requires technical assistance to resolve the issue. The Client application is unable to continue.

7.2 Join Request

This message is sent when a Client application is requesting to join a DDCA event. If accepted, the Client application shall transition from the **UNJOINED** state to the **READY** state

Table 22 Join request message

Field Name	Data Type	Description
ClientID	UUID	Unique identifier of the Client application
NameLength	Unsigned 16-bit integer	Indicates the length of the Name field
Name	Character array	Name of the Client application

7.3 Leave Request

This message is sent when a Client application is requesting to leave a DDCA event. Upon acknowledgement, the Client application shall transition into the **UNJOINED** state and is free to join other DDCA events

Table 23 Leave request message

Field Name	Data Type	Description
ClientID	UUID	Unique identifier of the Client application

7.4 Acknowledge

Many of the messages sent by the Master application are acknowledged by the Client application. Examples include: **Sync**, **Join**, **Leave**, **Close**, and **Load**. The components of an **Acknowledge** message sent by the Client application include the message ID being responded to (message ID), the Acknowledge State (Ack State), and the Client application's current state (CurrentState).

Table 24 Acknowledge message

Field Name	Data Type	Description
MessageID	Unsigned 32-bit integer	Unique identifier of the message
Ack State	Unsigned 8-bit integer	Status of Acknowledge
Current State	Unsigned 8-bit integer	Current state (see Table 1 for state descriptions)

Table 25 illustrates the supported values for Acknowledge State (Ack State).

Table 25 Acknowledge States

Acknowledge State	Value	Description
AbleToComply	1	Client application is able to comply with the Request or Command message being processed.
WaitingToComplete	2	Client application waiting to complete with the Request or Command message being processed.
UnableToComply	3	Client application is unable to comply with the Request or Command message being processed.

7.5 ListResponse

This message is sent when a Client application has received the **List** Control message from the DDCA Master application. Upon receipt of this message, the Client application shall immediately send an Acknowledge message with the Acknowledge State field set to “WaitingToComplete” and the CurrentState set to **READY**. When the Client application has retrieved its list, it shall send its list of all available sessions to the Master application via the **ListResponse** message.

Table 26 List Response message

Field Name	Data Type	Description
NumSessions	Unsigned 32-bit integer	Number of Session Data records contained in the ListResponse message.
ClientID	UUID	Unique identifier of the Client application
SessionData	variable	The fields in Table 27 are repeated for Number of Sessions

Table 27 Session data structure

Field Name	Data Type	Description
SessionIdentifier	UUID	Unique identifier of the Client session
StartTime	Unsigned 64-bit integer	The starting DataTime for the Session, in seconds since 01-Jan-1970
EndTime	Unsigned 64-bit integer	The ending DataTime for the Session, in seconds since 01-Jan-1970
NameLength	Unsigned 32-bit integer	Length of the SessionName field in bytes
SessionName	Character array	Name of the Client session

7.6 LoadResponse

Upon receipt of the **Load** Control message, the Client application shall immediately send an Acknowledge message with the Acknowledge State field set to WaitingToComplete and the CurrentState set to **READY**. Client application(s) shall then load any recording(s) that match the DataTime window specified in the **Load** Control message.

If a Client application successfully loads recording(s) that match the DataTime window specified in the **Load** Control message, it shall set the NumSessions field equal to the number of Sessions that have been loaded, shall send the **LoadResponse** message, and shall transition to the **STOP** state.

If a Client application fails to load any recording(s) that match the DataTime window specified in the **Load** Control message, it shall set the NumSessions field equal to zero (0), shall send the **LoadResponse** message, and shall remain in the **READY** state.

Table 28 Load Response message

Field Name	Data Type	Description
NumSessions	Unsigned 16-bit integer	Number of Session Data records loaded by the Client application.
ClientID	UUID	Unique identifier of the Client application

7.7 Status Response

This message is sent when a Client application is reporting its status to the DDCA Master application. If in the **UNJOINED** state, the Client application shall respond to any **Status Request** Control message received. If in any other state, the Client application shall respond only to **Status Request** Control messages from the Master application it is joined to. A **StatusResponse** message may be issued in any state, and results in no state transitions.

Table 29 Status response message

Field Name	Data Type	Description
ClientID	UUID	Unique identifier of the Client application
CurrentState	Unsigned 8-bit integer	Current state (see Table 1 for state descriptions)
NameLength	Unsigned 16-bit integer	Indicates the length of the Name field
Name	Character array	Name of the Client application

A DDCA Client application may be in exactly one state at any instance in time.

7.8 DDCA Compatibility Response

The **DDCACompatibilityResponse** message is sent when the Client application is reporting its compatibility level for DDCA interoperability to the Master application. The enumerated values for DDCA compatibility levels are represented in Table 14.

8 DDCA Interaction Examples

The following sub-sections provide example scenarios in which a DDCA Master and Client application(s) interact, and illustrate the DDCA Control messages used and DDCA states entered as a result of these scenarios.

8.1 Joining a DDCA event

This section describes how a DDCA Client application joins a DDCA event. There are two means a Client application may join a DDCA event: Client-initiated or Master-initiated. Both scenarios start with the Client application in an **UNJOINED** state.

In the Client-initiated scenario, the Client application sends a request to join message to the Master application. The Client application's **JoinRequest** message is received and processed by the Master application, which then issues a **JoinResponse** message to the Client application. If the Client application is allowed to join the DDCA event, the Response field contains the JoinAccept value. The Client application transitions to the **READY** state and is now joined to the DDCA event. If the Client application is not allowed to join the DDCA event, the Response field contains the JoinReject value.

In the Master-initiated join scenario, the Master application sends a **RequestStatus** to the Client application(s) it wishes to join the DDCA event. The Master application then selects one or more Client application(s) whose state is **UNJOINED**, and sends the Client application(s) a **Join** request. If the Client application(s) are able to join the DDCA event, the application(s) respond to with an **Acknowledge** message containing the MessageID from the Join message, and the Client application(s) transition to the READY state.

When a DDCA Client application joins a DDCA event, the application transitions from the **UNJOINED** to the **READY** state. From the **READY** state, following successful receipt and processing of the **Load** Control message, the Client application transitions into the **STOP** state. From the **STOP** state, the Client application transitions into one of three possible states: **SYNC**, **SEEK**, or **PAUSE**.

It is possible for a Client application to transition directly from the **STOP** state into a **SEEK** state; for example, if a recording is opened and the first **Sync** Control message is to seek to a point 10 minutes into the recording, the Client application transitions from **STOP** directly into **SEEK**.

It is possible for the Client application to transition directly from the **STOP** state into a **PAUSE** state; for example, if a recording is opened and the first **Sync** Control message is to pause, the Client application transitions from **STOP** directly into **PAUSE**.

It is possible for a Client application to transition directly from the **STOP** state into the **SYNC** state; for example, if a recording is opened and the first **Sync** Control message is to play at a speed of 1x, the Client application would transition into the **SYNC** state and would begin to replay its recorded data at 1x.

A DDCA Master-initiated join sequence, illustrated as a UML sequence diagram, is shown in Figure 3. UML sequence diagrams allow the specification of various runtime scenarios by showing a UML object's lifeline (as a vertical line) and the messages exchanged with other UML object(s). Figure 3 illustrates the sequence of events and interactions that occur when a Master application requests a Client application to join a DDCA event. In this sequence, the Master application requests status information from all Client applications, and chooses one (or more) Client application(s) that are issued a Join request. Upon receipt of the Join request, the Client application, should it wish to join the DDCA event, shall issue a **JoinResponse** message with an Accept status. Should the Client application not wish to join the DDCA event, the Client application shall acknowledge the request with a Reject status in the **JoinResponse** message.

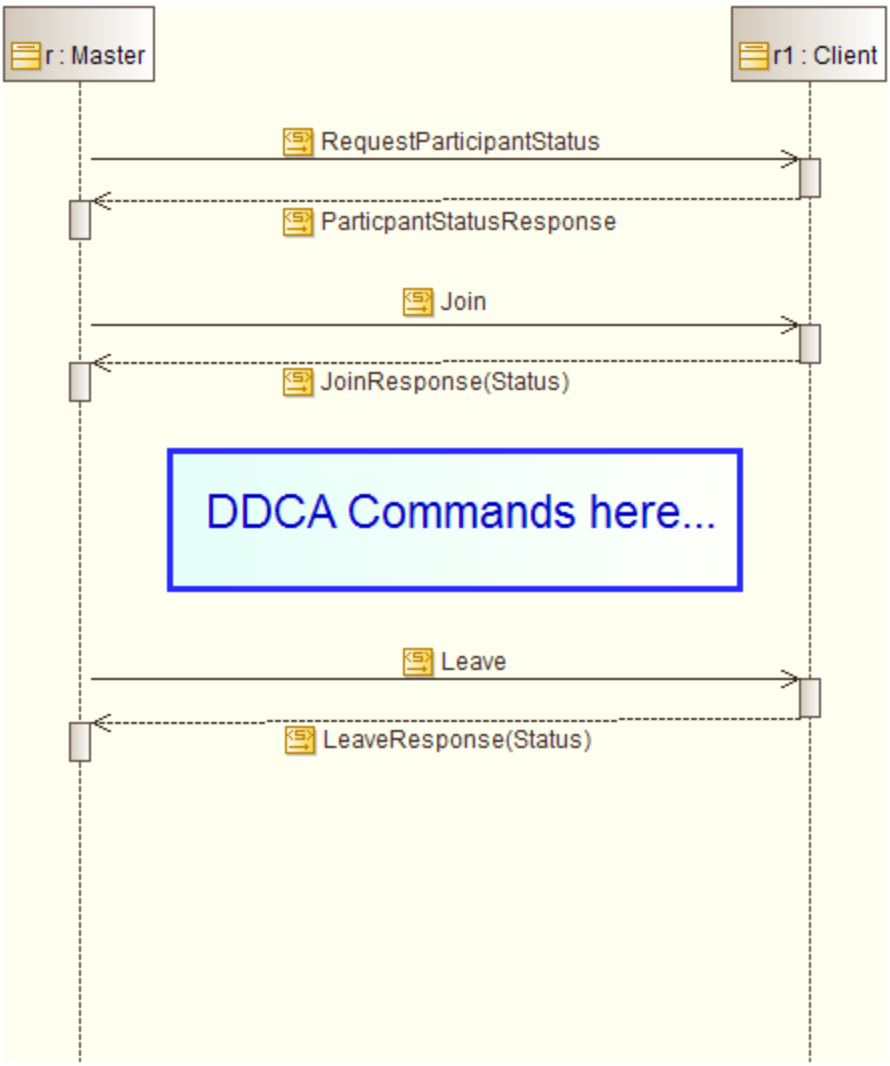

Figure 3 DDCA Master-Initiated Join

Figure 4 illustrates the sequence of interactions that occur when a Client application requests to join a DDCA event. In this sequence, the Client application issues a **JoinRequest** to the Master application. The Master application shall issue a **JoinResponse** message with either an Accept or Reject status; the value of the status field indicates whether the Master application accepts or rejects the Client application's join request.

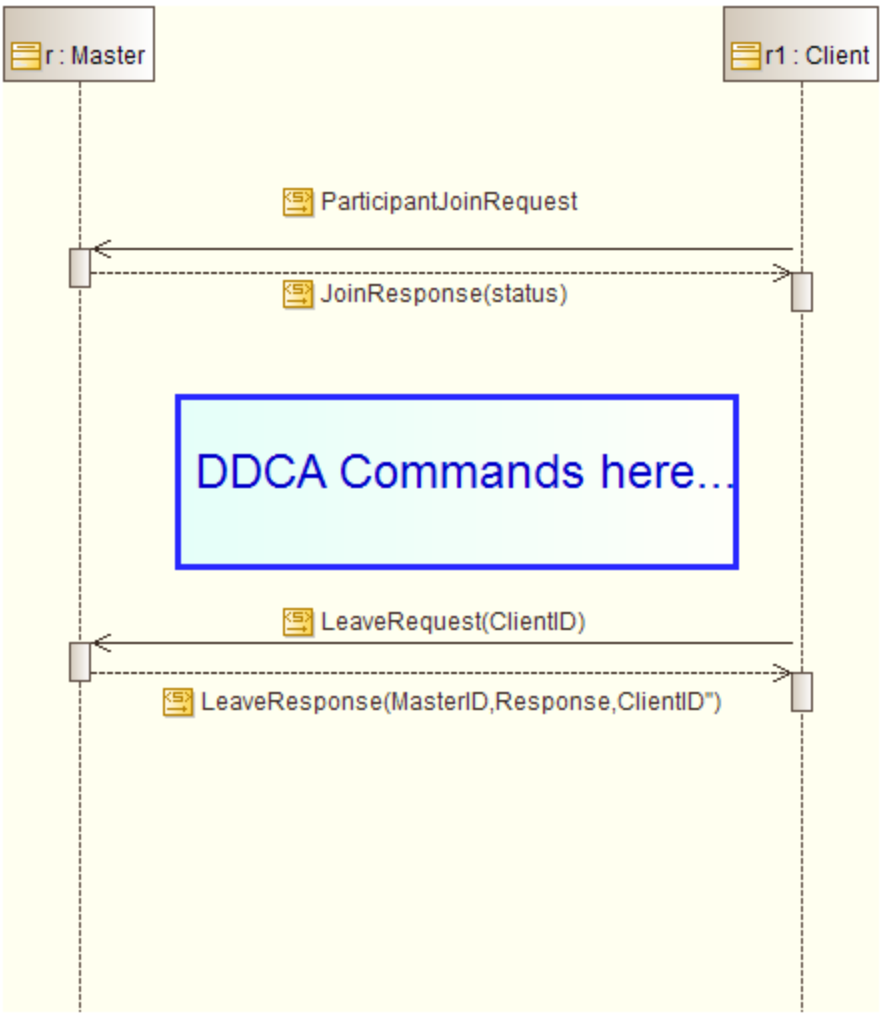

Figure 4 DDCA Client-Initiated Join

8.2 Example Playback Sequence

An example of DDCA playback is shown in Figure 5 In this example, there are two object lifelines: the DDCA Master application (named master in the diagram below) and the DDCA Client application (Client in the sequence diagram). The messages exchanged between the Master application and Client application during a typical playback sequence are illustrated in the order they are expected to occur. Sync messages are sent periodically by the Master application, so there may be 0, 1, or more than 1 Sync message with the same value(s) between any consecutive steps.

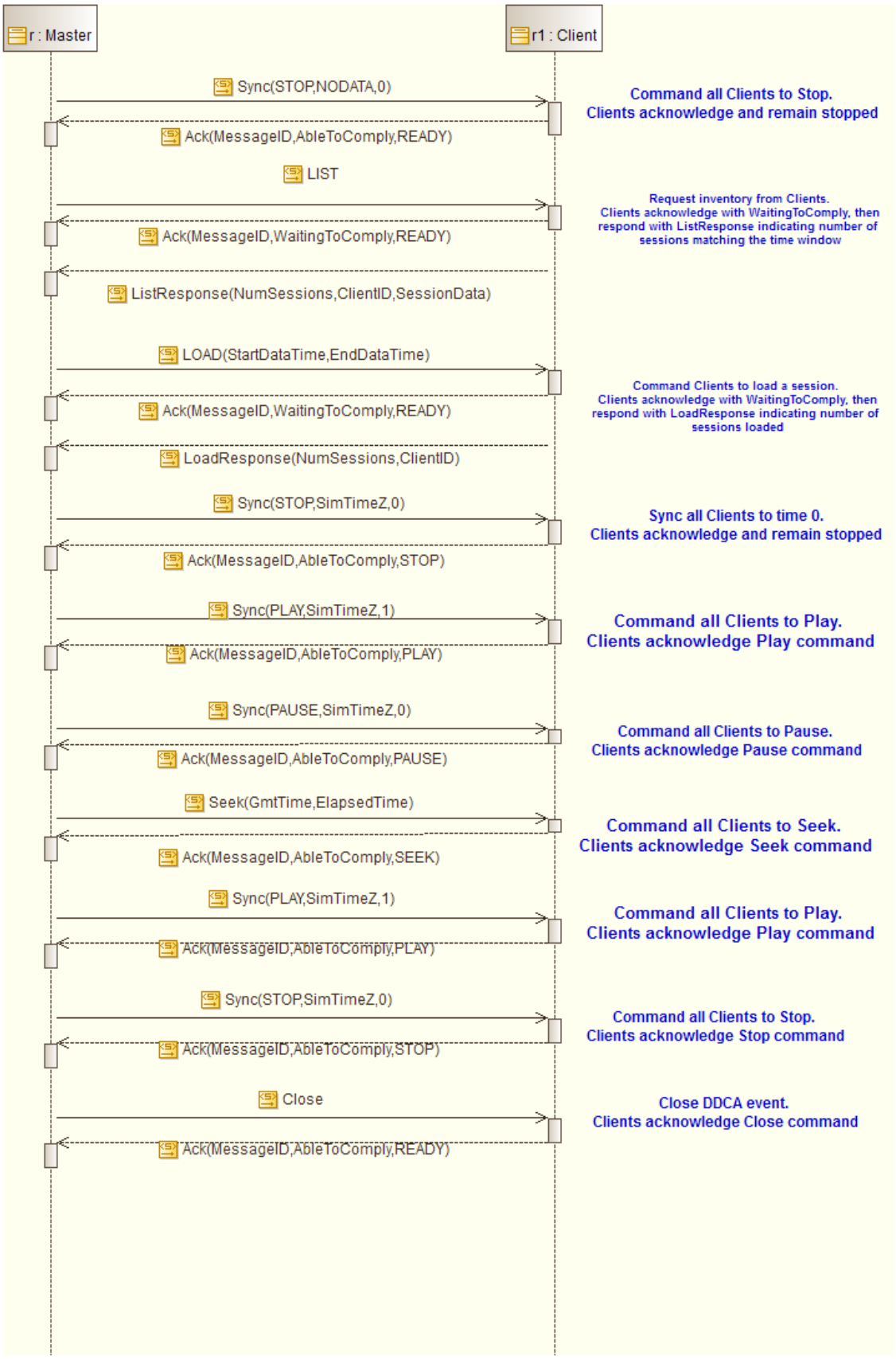

Figure 5 DDCA Playback Sequence

Throughout a DDCA event, the Master application broadcasts **Sync** messages at periodic intervals. **Sync** messages are used to ensure that all Client application(s) of the controller are in the same time and state as the controller and each other. As the contents of the **Sync** message change, they will be noted in the sequence description below. **Sync** messages sent by the Master application are responded to with an Acknowledge message by DDCA Client application(s) that are currently joined to the controller issuing the **Sync**.

Prior to initiating the start of the distributed debrief event, the Master application issues **Sync** messages indicating a **STOP** state and NODATA time. Joined Client application(s) respond to the **Sync** messages with **Acknowledge** messages confirming a **READY** state.

When it is time to begin the distributed debrief, the Master application issues a **List** message, requesting all DDCA Client application(s) to respond with a list of available recordings. Client application(s) would then respond to the **List** message with **ListResponse** messages, showing the session ID and session name of each playback file that is available.

Once the Master application has chosen the session to playback, the Master application issues a **Load** message. This message identifies to the Client application(s) the file to load for playback. **Sync** messages after the load should now indicate the DataTime of the playback file, but otherwise not change. **Acknowledge** responses from the DDCA Client application(s) continue to indicate a **STOP** state.

When the playback is to begin, the Master application's **Sync** messages change to indicate a **SYNC** state, the current DataTime of the playback, and the playback speed of 1 (real-time). **Acknowledge** responses sent from the DDCA Client application(s) confirm the **SYNC** state.

As needed for discussion, the Master application issues **Sync** messages with a speed of 0 (PAUSE), 1 (PLAY), other positive integer value (fast-forward), or negative integer value (rewind). **Acknowledge** messages from the DDCA Client application(s) confirm the PLAY, PAUSE, or FF/REW state as indicated.

The Master application issues periodic **Seek** messages, indicating where exactly in the DataTime window the Client applications are to point. If the previous state was **SYNC**, the Master application issues a **Pause** message prior to issuing the **Seek** message. In a **Seek** operation, the Client application executes the necessary operations to move its internal pointer to a specific point within the time window; i.e. the Client application executes a "jump to" operation. **Acknowledge** messages from the DDCA Client application(s) confirm the **SEEK** state as indicated.

When the playback is finished, the Master application issues **Sync** messages with the state of **STOP**, the current DataTime time at the time of the stop, and a speed of 0. **Acknowledge** responses from the DDCA Client application(s) confirm the **STOP** state.

The Master application then issues a **Close** message, indicating that all joined DDCA Client application(s) may free up any committed resources. The Client applications' **Acknowledge** responses indicate a state of **READY**. The Master application now issues **List** or **Load** Control messages to repeat the playback sequence as necessary.

At this point, the Master application is able to issue a **Leave** message, informing the Client application(s) that the session has ended. Client application(s) do not need to issue an **Acknowledge** response to this message, but instead transition to the **UNJOINED** state.

8.3 Example Failed Playback Sequence

An example of DDCA playback wherein a Client application does not have sessions matching the time window is shown in Figure 6 In this example, there are two object lifelines: the DDCA Master application

(named Master in the diagram below) and the DDCA Client application (Client in the sequence diagram). The messages exchanged between the Master application and Client application during a typical playback sequence are illustrated in the order they are expected to occur. Sync messages are sent periodically by the Master application, so there may be 0, 1, or more than 1 Sync message with the same value(s) between any consecutive steps.

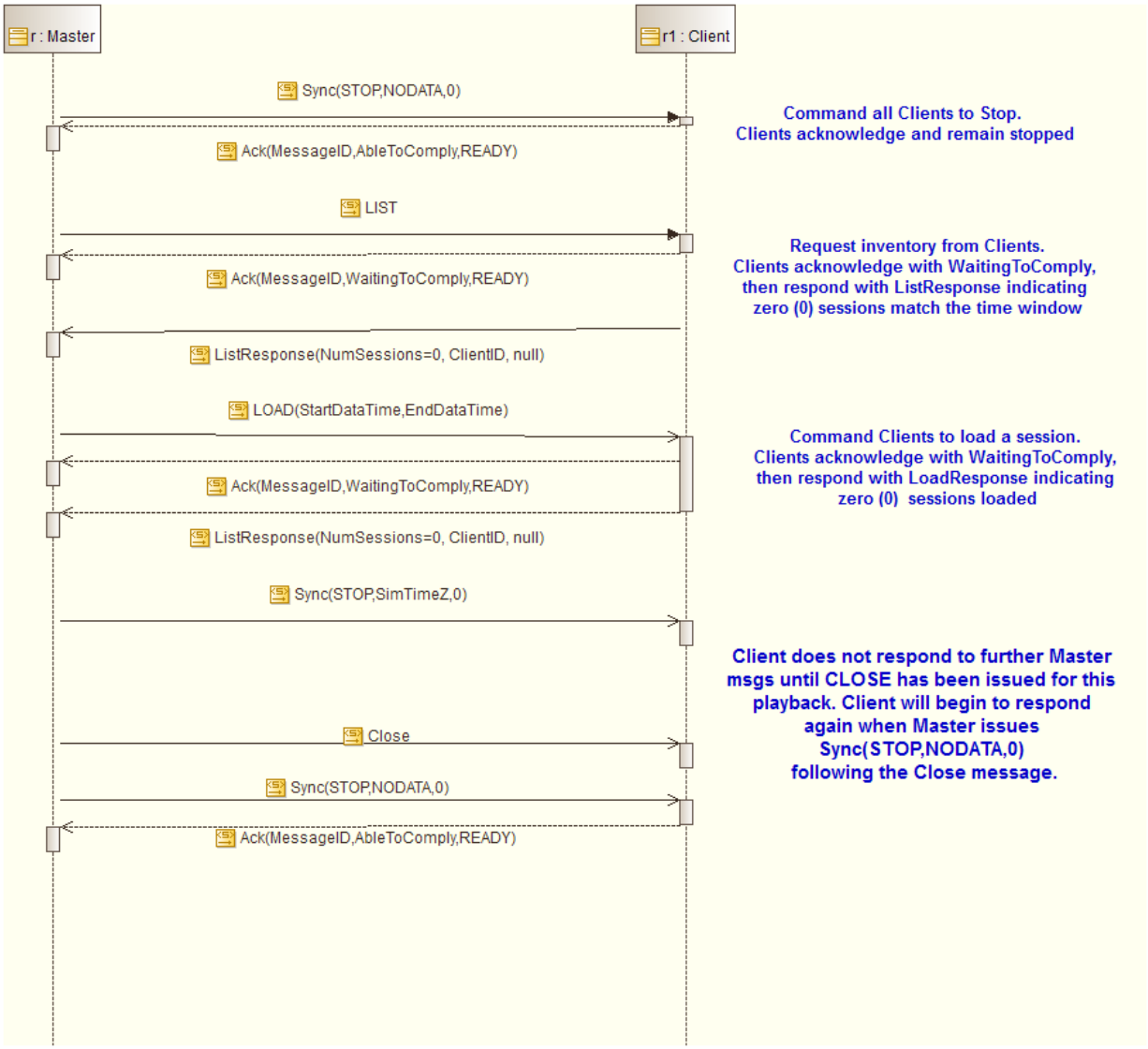

Figure 6 Client has no sessions for Playback

Throughout a DDCA event, the Master application broadcasts **Sync** messages at periodic intervals. **Sync** messages are used to ensure that all Client application(s) of the controller are in the same time and state as the controller and each other. As the contents of the **Sync** message change, they will be noted in the sequence description below. **Sync** messages sent by the Master application are responded to with an Acknowledge message by DDCA Client application(s) that are currently joined to the controller issuing the **Sync**.

Prior to initiating the start of the distributed debrief event, the Master application issues **Sync** messages indicating a **STOP** state and NODATA time. Joined Client application(s) respond to the **Sync** messages with **Acknowledge** messages confirming a **READY** state.

When it is time to begin the distributed debrief, the Master application issues a **List** message, requesting all DDCA Client application(s) to respond with a list of available recordings. Client application(s) which do not have available sessions matching the DataTime window specified in the **Load** message will set the NumSessions field of the **ListResponse** message to zero (0).

Once the Master application has chosen the session to playback, the Master application issues a **Load** message. This message identifies to the Client application(s) the DataTime window to load for playback. Client application(s) which no not have a session matching the DataTime window will set the NumSessions field of the **LoadResponse** message to zero (0). If no matching sessions are found, the Client application(s) **Acknowledge** responses continue to indicate a **READY** state.

Client application(s) that have not identified a session to load continue to respond (with Acknowledges) to further SYNC messages indicating an Acknowledge Status of UnableToComply and a **READY** state, until such time as the Master issues a **Close** message followed by a **Sync** message with a state of **STOP** and a time of NODATA.

At this point, the Master application is able to issue a **Leave** message, informing the Client application(s) that the session has ended. Client application(s) do not need to issue an **Acknowledge** response to this message, but instead transition to the **UNJOINED** state.