

STATE OF COLORADO

Intelligent Transportation Systems
700 Kipling Street, Suite 2500
Lakewood, Colorado 80215
Phone (303) 512-5834
FAX (303) 239-0848



INTEGRATION **Contract Routing No. 04 HAA 00063**

Skyline Detailed Design

Version 1.0

Approved By

Frank Kinder
CDOT ITS Office

Signature: _____

Date: _____

John Williams
CDOT ITS Office

Signature: _____

Date: _____

Prepared By:



CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

Revision History

Date	Version	Description	Author
April 20, 2005	1.0	Initial Version	Ramesh Vellanki

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

Table of Contents

i.	Executive Summary	1
ii.	Summary of Key Technologies	1
1.	Introduction	2
1.1	Purpose	2
1.2	Scope	2
1.3	Audience	2
1.4	Reference	2
2.	Key Design Concepts	2
2.1	Standards	2
2.2	Key Technologies	2
3.	Detailed Design	3
3.1	Cdot.ctms.layer.services.comm.skyline.interaction.SkylineRecordFactory	3
3.2	Skyline Record Factory Class Diagram	3
3.3	Skyline Record Factory List Sequence Diagram	4
3.4	Skyline Record Factory Map Sequence Diagram	4
3.5	Cdot.ctms.layer.services.comm.skyline.connection.SkylineModemConnectionFactory	5
3.6	Cdot.ctms.layer.services.comm.skyline.connection.SkylineSerialConnectionFactory	5
3.7	Cdot.ctms.layer.services.comm.skyline.connection.SkylineSerialConnection	6
3.8	Cdot.ctms.layer.services.comm.skyline.connection.SkylineModemConnection	6
3.9	Skyline Connection Factory Class Diagram	6
3.10	Skyline Modem Connection Factory Sequence Diagram	8
3.11	Skyline Serial Connection Factory Sequence Diagram	8
3.12	Cdot.ctms.layer.services.comm.skyline.interaction.SkylineInteractionFactory	9
3.13	Skyline Interaction Factory Class Diagram	10
3.14	Cdot.ctms.layer.services.comm.skyline.interaction package	10
3.14.1	PollDmsInteraction	10
3.14.2	Poll Dms Interaction Sequence Diagram	11
3.14.3	TestDmsPixelsInteraction	11
3.14.4	ActivateMessageInteraction	12
3.14.5	RFSInteraction	12
3.14.6	ClearSignInteraction	12
3.14.7	AdjustBrightnessInteraction	12
3.15	Cdot.ctms.layer.services.comm.skyline.data package	12

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

Table of Figures

Figure 1 Skyline Record Factory Class Diagram	3
Figure 2 Skyline Record Factory List Sequence Diagram.....	4
Figure 3 Skyline Record Factory Map Sequence Diagram	5
Figure 4 Skyline Connection Factory Class Diagram.....	8
Figure 5 Skyline Modem Connection Factory Sequence Diagram	8
Figure 6 Skyline Serial Connection Factory Sequence Diagram.....	9
Figure 7 Skyline Interaction Factory Class Diagram.....	10
Figure 8 Skyline Poll Dms Interaction Sequence Diagram	11

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

Skyline Detailed Design

i. Executive Summary

This document describes the detailed design of the Skyline extensions to the default Comm Server implementation. It discusses how the core Comm Server components are customized for Skyline specific features.

ii. Summary of Key Technologies

The development Communications server core framework will employ proven technologies, software methodologies, and best practices. Specifically, the Rational Unified Process will be followed in order to facilitate a thorough analysis and design of the complex system. The core technology will be built using Java, EJB (Enterprise JavaBeans), J2CA (Java Connector Architecture), XML, Object-Oriented concepts, and several design patterns including Factory, Delegate, Model-View-Controller, Service Activator, and Service Locator.

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

1. Introduction

This document describes the detailed design of the Skyline extensions to the default Comm Server implementation. It discusses how the core Comm Server components are customized for Skyline specific features.

1.1 Purpose

- ❑ To supplement the technical information provided in the Communications Server Detailed Design.
- ❑ The reader should be able to understand how to add vendor specific implementations to the Comm Server.

1.2 Scope

The scope of this document is the Skyline specific extensions to the core Comm Server.

1.3 Audience

- ❑ Architects
- ❑ Developers

1.4 Reference

The following references were either used in the creation of this document or may be used to supplement the detailed descriptions of the Skyline detailed design.

- ❑ Communications Server Detailed Design.doc

2. Key Design Concepts

The same design concepts that were used in the implementation of the core Comm Server are used here.

2.1 Standards

The same standards that were used in the implementation of the core Comm Server are used here.

← Formatted: Bullets and Numbering

2.2 Key Technologies

The same technologies that were used in the implementation of the core Comm Server are used here.

← Formatted: Bullets and Numbering

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

3. Detailed Design

The following section describes the Skyline extensions to the core Comm Server in detail. It describes the Comm API interfaces that are implemented in order to plug the Skyline specific features into the Comm Server. It follows the section on “Extending the Comm Server” from the Communications Server Detailed Design.

3.1 Cdot.ctms.layer.services.comm.skyline.interaction.SkylineRecordFactory

This class implements the `cdot.ctms.layer.services.comm.cci.RecordFactory` interface. It uses `MibFactory` and `NTCIPFactory` to build an ordered collection of Records or a map of records keyed by the record name.

3.2 Skyline Record Factory Class Diagram

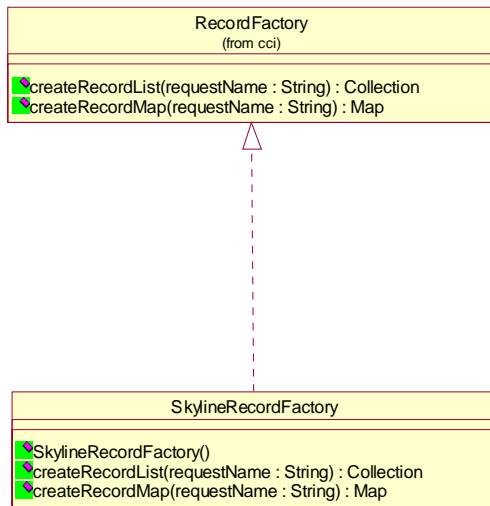


Figure 1 Skyline Record Factory Class Diagram

3.3 Skyline Record Factory List Sequence Diagram

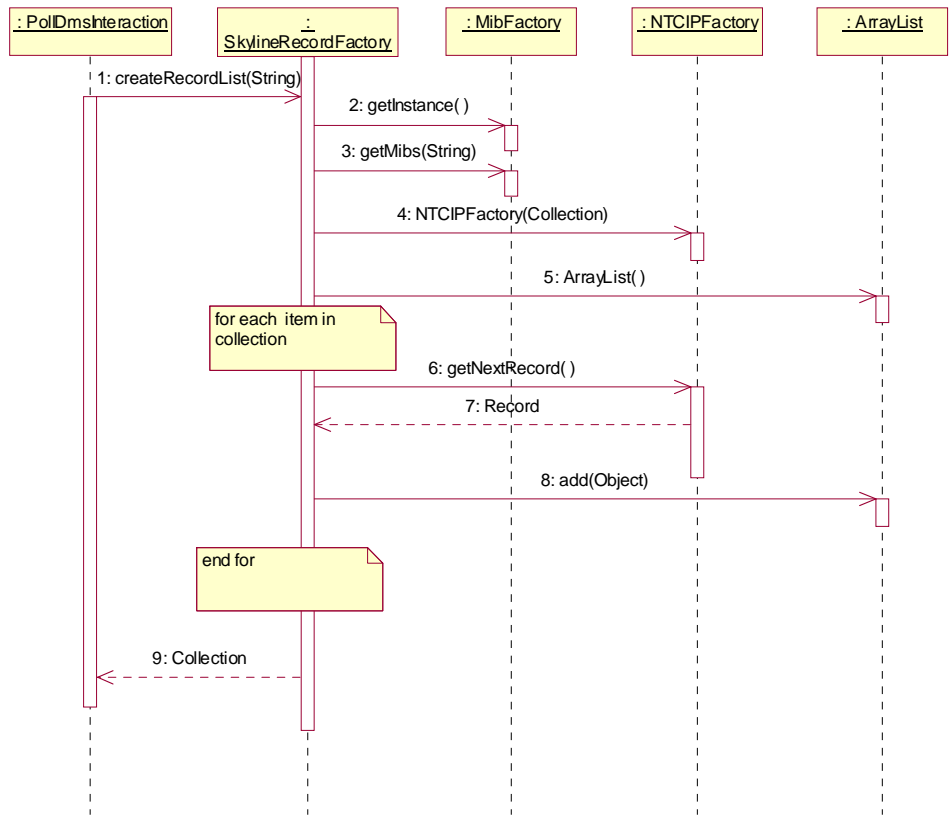


Figure 2 Skyline Record Factory List Sequence Diagram

3.4 Skyline Record Factory Map Sequence Diagram

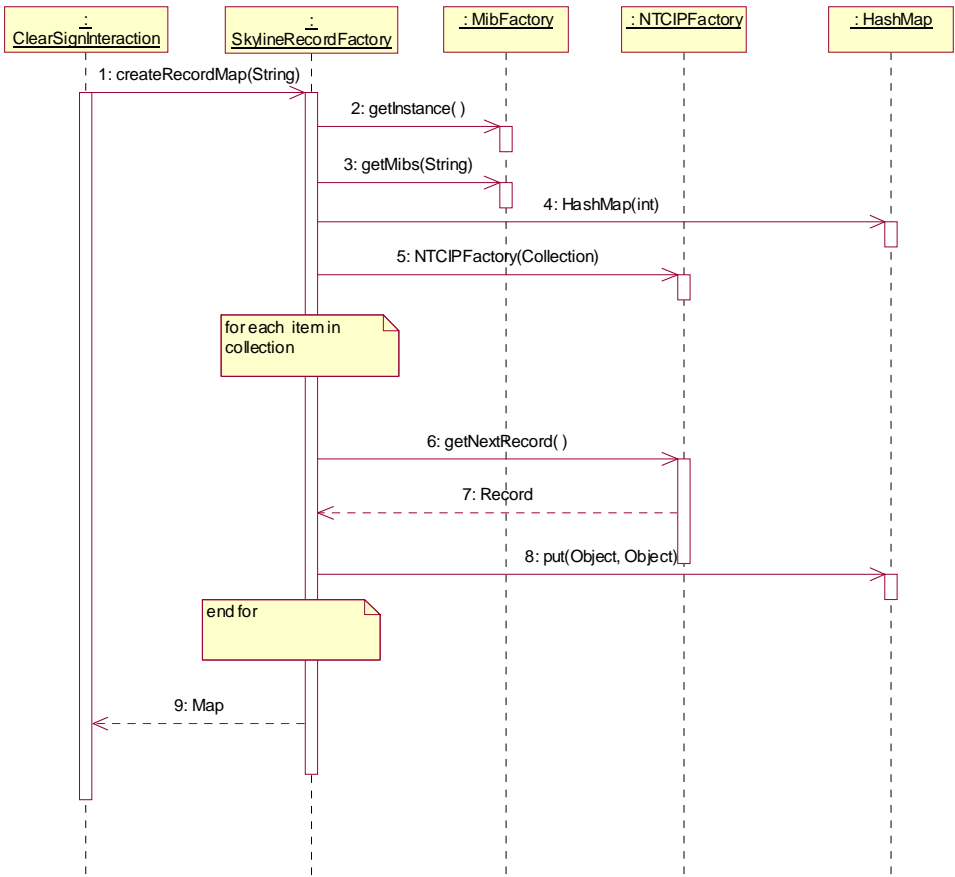


Figure 3 Skyline Record Factory Map Sequence Diagram

3.5 Cdot.ctms.layer.services.comm.skyline.connection.SkylineModemConnectionFactory

This implements the `cdot.ctms.layer.services.comm.spi.ConnectionFactory` interface. It provides a default constructor that allows it to be instantiated by the Container when it is deployed. The `getConnection()` method uses the `ConnectInfo` that is passed to it to build a new `SkylineModemConnection`. It also sets the appropriate `ProtocolAdaptor` in the `SkylineModemConnection` before returning it. The `ProtocolAdaptor` contains all the protocol information needed to execute records.

3.6 Cdot.ctms.layer.services.comm.skyline.connection.SkylineSerialConnectionFactory

This also implements the `cdot.ctms.layer.services.comm.spi.ConnectionFactory` interface. It returns a `SkylineSerialConnection` from its `getConnection()` method.

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

3.7 Cdot.ctms.layer.services.comm.skyline.connection.SkylineSerialConnection

It specializes the CTMSSerialPort connection's execute() to handle SNMP encoding of the Record. It's function is to execute an input record over the connection.

3.8 Cdot.ctms.layer.services.comm.skyline.connection.SkylineModemConnection

It specializes the CTMSModemPort connection's execute() to handle SNMP encoding of the Record. It's function is to execute an input record over the connection.

3.9 Skyline Connection Factory Class Diagram

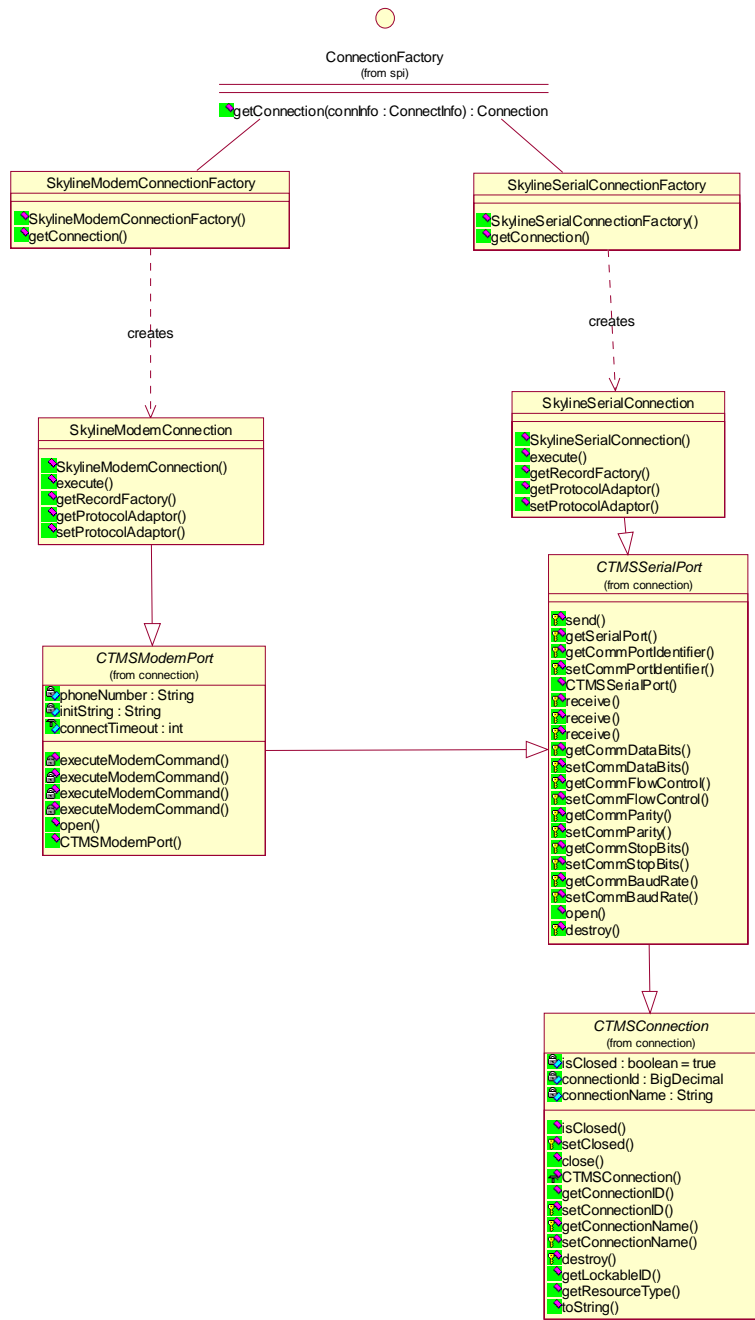


Figure 4 Skyline Connection Factory Class Diagram

3.10 Skyline Modem Connection Factory Sequence Diagram

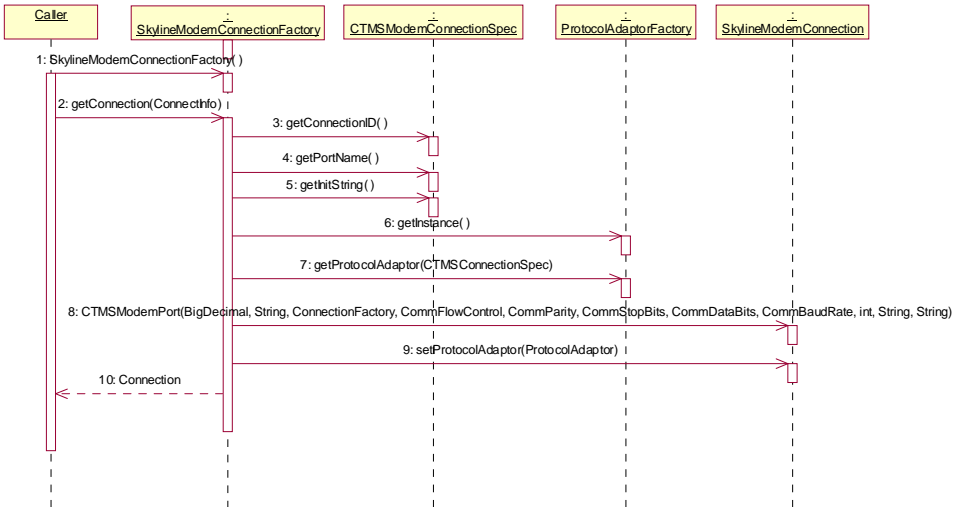


Figure 5 Skyline Modem Connection Factory Sequence Diagram

3.11 Skyline Serial Connection Factory Sequence Diagram

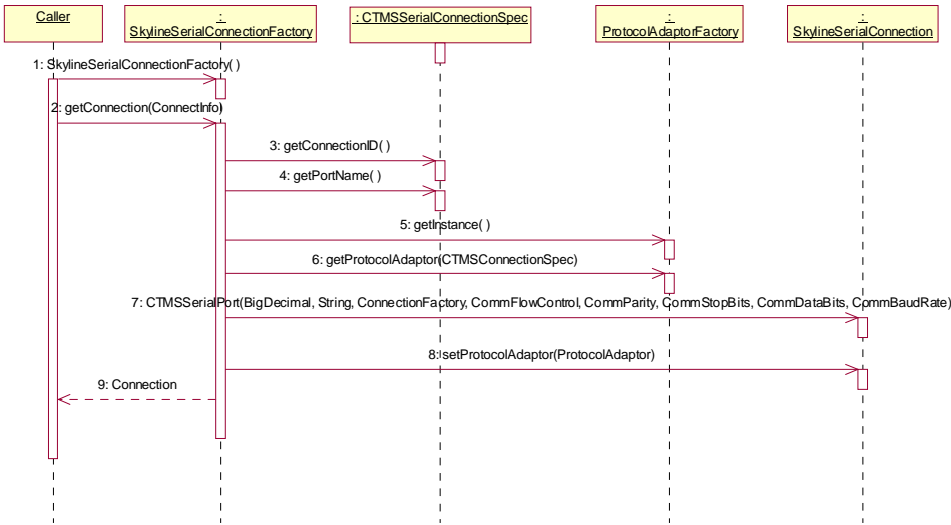


Figure 6 Skyline Serial Connection Factory Sequence Diagram

3.12 Cdot.ctms.layer.services.comm.skyline.interaction.SkylineInteractionFactory

This class implements the `cdot.ctms.layer.services.comm.cci.InteractionFactory` interface. The `CTMSContainer` invokes the `createInteraction()` method passing in the `Connection` and the `InteractionContext`. `SkylineInteractionFactory` then gets the instruction type and creates the appropriate interaction and returns it. The interactions that have been implemented are `PollDmsInteraction`, `RFSInteraction`, `AdjustBrightnessInteraction`, `TestDmsPixelsInteraction` and `ClearSignInteraction`.

3.13 Skyline Interaction Factory Class Diagram

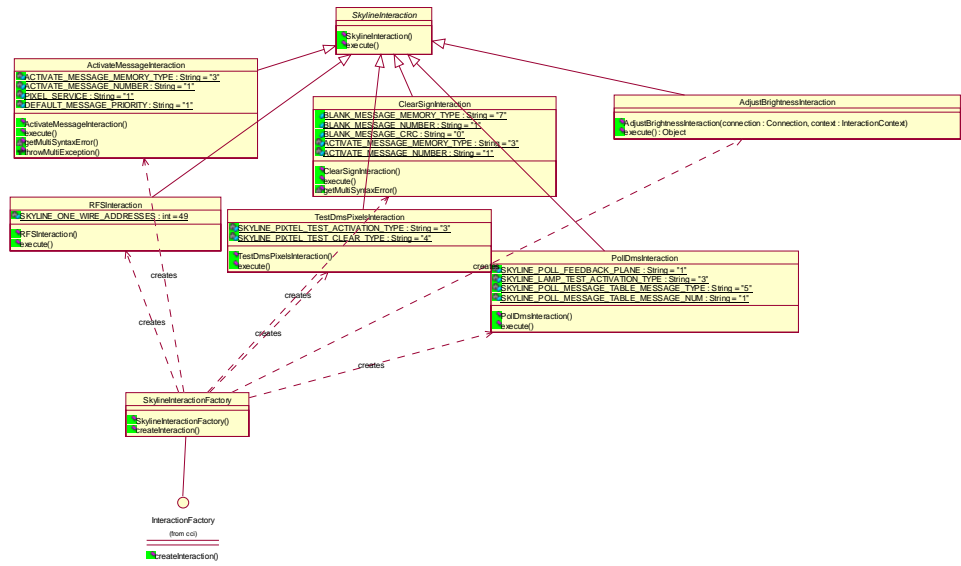


Figure 7 Skyline Interaction Factory Class Diagram

3.14 Cdot.ctms.layer.services.comm.skyline.interaction package

The classes in this package implement the SkylineInteraction interface. All the logic of executing the records is implemented in the execute() method. The information needed to execute the records is available in the InteractionContext. All the interactions follow a repeatable pattern. They execute the records, store the results in a HashMap and finally call the exchanges to build the appropriate DTOs. In addition, they may also execute other interactions. They then populate a HashMap with the DTO(s) and return the HashMap or return a single DTO. If there are any errors during the execution, a CommException is thrown.

3.14.1 PollDmsInteraction

PollDmsInteraction is responsible for executing the records that provide the latest status of various properties on a DMS. Some of these properties include the temperature, sensor statuses, door statuses, true feedback with the current message on the DMS, etc. It reads the technology type of the DMS and creates a list of records depending upon the technology type. For now, the two technology types supported are LED and Lamp. It can also be extended quite easily to support other types as well. This interaction makes use of NTCIPGroupRecord to execute multiple records at once. It keeps count of the number of failures during record execution. Five consecutive failures is considered failure of the poll itself and it throws a CommException. It also executes the TestDmsPixelsInteraction at the end and returns a HashMap with PollModelCDTO and PixelTestModelCDTO. The following sequence diagram illustrates the steps needed to execute the poll.

3.14.2 Poll Dms Interaction Sequence Diagram

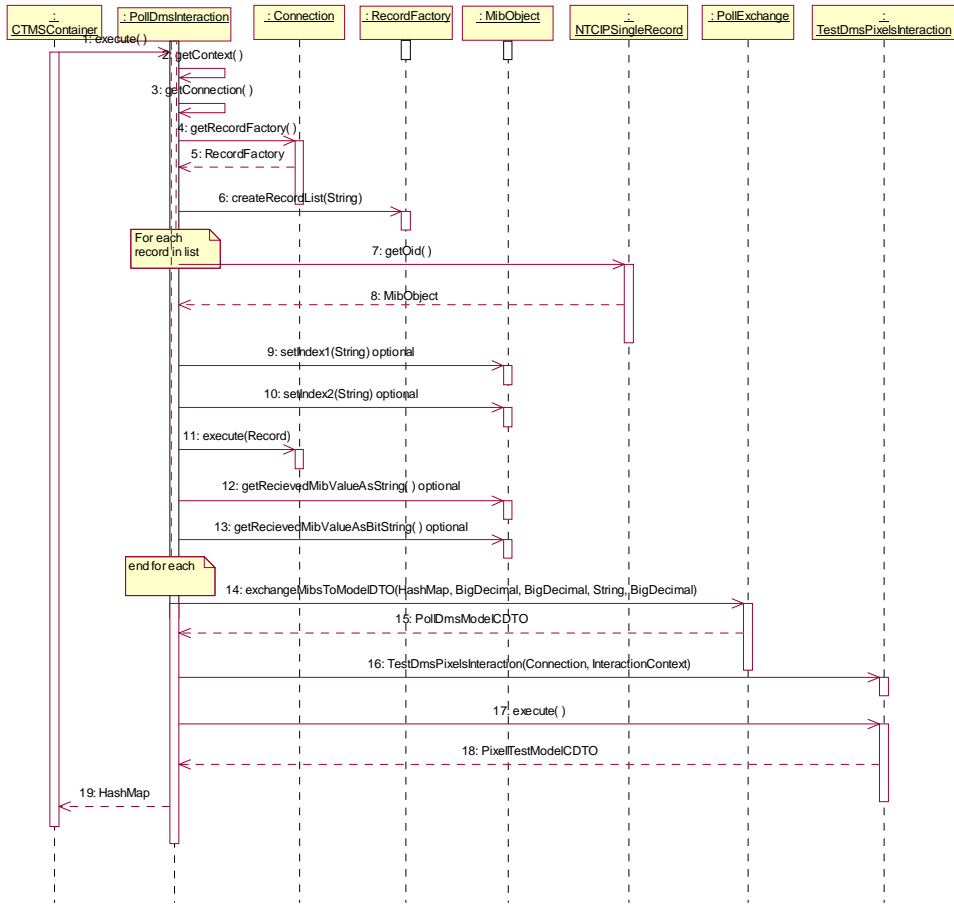


Figure 8 Skyline Poll Dms Interaction Sequence Diagram

3.14.3 TestDmsPixelsInteraction

This interaction is responsible for executing the records that indicates if there are any pixel errors on the DMS. It first executes the pixel test and then reads the number of pixel errors and the details of the errors. The details include the position of the error and the error type. Error types are stuck on, stuck off, half stuck on and half stuck off. It returns a PixelTestModelCDTO.

CTMS/CTIS	Version: 1.0
Communications Server Detailed Design	Date: April 20, 2005

3.14.4 ActivateMessageInteraction

This interaction is responsible for activating a message on a DMS. It first maps the CTMS priority to a number between 1 and 255 on the DMS. If the CTMS priority is low, the priority on the sign is set 1. For medium it is set to 150 and for high, it is set to 225. After the message has been activated on the DMS, it verifies the CRC values to make sure that there were no errors during the activation. It finally executes a poll and returns a HashMap with DmsMessageDTO, PollModelCDTO and PixelTestModelCDTO. The reason for executing a poll after activating a message is to make sure that the CTMS system always has a record of the latest message on the DMS and the status of the DMS as well.

3.14.5 RFSInteraction

The purpose of this interaction is to read the latest configuration data from the DMS. Some of the configuration items include the technology type, number of modules, module size, etc. It is mostly used after adding a new DMS into the CTMS system. Reading the configuration from the DMS itself instead of manually entering the info guarantees that the CTMS system has the correct configuration of the DMS. It returns a DmsConfigModelCDTO.

3.14.6 ClearSignInteraction

This interaction is used to clear the current message on the DMS. It follows the same pattern as ActivateMessageInteraction. It executes a poll after the message has been successfully cleared and returns a HashMap with DmsMessageDTO, PollModelCDTO and PixelTestModelCDTO.

3.14.7 AdjustBrightnessInteraction

This interaction is used to adjust the brightness on a DMS. Reasons for adjusting the brightness may include the time of the day, aging of the display elements on the DMS etc. Brightness mode can be set to auto or manual. If the mode is set to auto, the DMS will use the photo sensors on the DMS and adjust the brightness automatically. If the mode is manual, it sets the brightness value. Finally, it reads the settings back to make sure that the changes took effect. Finally, it uses AdjustBrightnessExchange to build a DmsBrighttessCDO and returns it.

3.15 Cdot.ctms.layer.services.comm.skyline.data package

This package contains all the exchanges required to convert the results of Record execution into a format that can be used to populate the model DTOs. After the interactions execute a record successfully, they place the results into a HashMap that is keyed by the name of the MIB. The exchanges have the intelligence to decode the data and convert it into meaningful values and populate the appropriate model DTOs. The logic needed to decode the result of each MIB is very specific. For example, in the PollExchange, in order to determine the doors that are open, the integer value that is returned from the MIB is converted to a bit string and each bit gives the status of a door(1 is open, 0 is closed).

PollExchange decodes the results of polling a DMS and returns a PollModelCDTO.

RFSExchange decodes the results of a “read from sign” and returns a DmsConfigModelCDTO.

PixelTestExchange decodes the results of a pixel test and returns a DmsPixelTestModelCDTO.

AdjustBrightnessExchange decodes the results of adjusting brightness.