DOWNTOWN LINE

Contract C955

InteGRATED SUPERVISORY

CONTROL SYSTEM (iscS)

distributed heterogeneous database propagation design spec

**AMENDMENT RECORDS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Rev*** | ***Date*** | ***Descriptions*** | ***Writer*** | ***Function*** |
| *1* | *05/07/2012* | First draft | Ouyang Zhilin | Module design note |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewed by** |  | **Approval by** |  |
| **Entity / Function** |  | **Entity / Function** |  |
| **Date & Sign.** |  | **Date & Sign.** |  |
| **Reserved to the Engineer** |  |  |  |

**Table of Contents**

[1 introduction 5](#_Toc331665235)

[1.1 General 5](#_Toc331665236)

[1.1.1 Purpose 5](#_Toc331665237)

[1.1.2 Version Control 5](#_Toc331665238)

[1.1.3 Definitions,Acronyms and Abbreviations 5](#_Toc331665239)

[1.1.4 References 5](#_Toc331665240)

[2 Motivation 5](#_Toc331665241)

[2.1 Background 5](#_Toc331665242)

[2.2 Requirements 6](#_Toc331665243)

[2.2.1 Formal Requirements 6](#_Toc331665244)

[2.2.2 Derived Requirements 6](#_Toc331665245)

[3 constraints 6](#_Toc331665246)

[3.1 Environment Constraints 6](#_Toc331665247)

[3.2 System Requirement Constraints 7](#_Toc331665248)

[4 system overview 7](#_Toc331665249)

[5 system architecture 7](#_Toc331665250)

[5.1 System Architecture Design 7](#_Toc331665251)

[5.2 Module Decomposition Design 8](#_Toc331665252)

[5.3 Design Rational 9](#_Toc331665253)

[5.4 Related cots 9](#_Toc331665254)

[5.5 Detail Design 9](#_Toc331665255)

[5.5.1 Sequence Diagram 9](#_Toc331665256)

[5.5.2 Threading and Concurrence 10](#_Toc331665257)

[5.5.3 Message Protocol 11](#_Toc331665258)

[5.5.4 Class Diagram 11](#_Toc331665259)

[5.5.5 Componment Diagram 12](#_Toc331665260)

[5.5.6 Dependencies 12](#_Toc331665261)

[5.5.7 Error Handling 12](#_Toc331665262)

[5.5.8 Propagation Clustering 12](#_Toc331665263)

[5.5.9 Configuration Information 12](#_Toc331665264)

[5.6 Testability 12](#_Toc331665265)

[5.7 Maintainability 12](#_Toc331665266)

[5.8 Impact Analysis 12](#_Toc331665267)

[5.9 Deployment Considerations 12](#_Toc331665268)

[5.10 Flow-on Effects 12](#_Toc331665269)

[5.11 Extensibility and Reuse 12](#_Toc331665270)

[6 gui-component design information 12](#_Toc331665271)

[Appendix A: Software module design review 14](#_Toc331665272)

# introduction

## General

### Purpose

This document has been written as an overall architecture of the library for the distributed heterogeneous databases propagation. It provides detail design and architecture of the message propagation and enables heterogeneous databases communicate with each other to propagate messages like Oracle AQ propagation feature.

### Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes / Reason for Issue |
| 0.01 | 05 July 2012 | Ouyang.Zhilin | First Draft |
| 1.00 |  |  | Issued for Release |
|  |  |  |  |

### Definitions,Acronyms and Abbreviations

|  |  |
| --- | --- |
| Name | Description |
| AQ | Advance Queue |
| MAQ | MySQL Advance Queue |
|  |  |
|  |  |

### References

|  |  |  |
| --- | --- | --- |
|  | Reference | Title |
|  | [9999-P01-01-0001](I:\\STE\\Doc\\Project Documents\\C955_docs\\Database\\DbSynchDesign\\P-Product\\P01-Common\\P0101-General\\9999P01010001-System_Glossary.doc) | System Glossary |
|  |  | Software requirement |
|  |  | Software ICD |
|  |  | Software Architecture Specification |
|  |  | IEC62279 Phases |

# Motivation

This section describes the motivation of the message propagation including the background and major requirements of the sub-system.

## Background

Due to the differences of the Database Deployment between project C955 and C830, the data replication mechanism need to revise based on the new requirements.

In the existing project C830 both the local and central databases are Oracle, and therefore the data replication mechnism used Oracle built-in AQ funcationalities. The new project C955, however, the type of database servers will be defined as below:

* Central Database – residing at a central location accessible across a WAN. For C955 this is the OCCDB using Sun Oracle Enterprise database server.
* Local Database – associated with any physical location accessible from other local servers and central server across a WAN. For C955 this database will use MySQL database and will be considered as the Local Database for agents running on the location for the purpose of this design.

Based on the above design, message propagation cannot feature at station server as heterogeneous databases applied in the system. Therefore message propagation mechanism is needed to enable heterogeneous databases (such as MySQL & Oracle etc.) can interact with each other like the Oracle to Oracle built-in AQ propagation.

## Requirements

This section is to outline the requirements for the message propagation that can be use as a basis for developing possible solution using appropriate technologies.

### Formal Requirements

The requirements of message propagation mechanism for heterogeneous databases should follow the same capabilities of Oracle AQ propagation such as:

1. Queue to queue propagation in which case delivers messages from the source queue to a specific destination queue identified by the address.
2. High performance and reliable communication channel model which enable messages to propagate in a consistence manner among heterogeneous distributed databases.
3. Capabilities for handling failures and reporting errors and
4. Provides propagation management statistics

### Derived Requirements

Some derived requirements are introduced for easy maintenance and monitor the status of the message propagation. Therefore the following statistics of the message propagation are needed:

1. Propagation Job ID
2. Number of propagation channels
3. Total number of messages propagated for each message channel
4. Total number of bytes propagated for each message channel
5. Average size of propagated messages and
6. Average time to propagate a message.

# constraints

This section describes the constraints including environment and system requirement constraints .

## Environment Constraints

The message propagation mechanism currently can work with MySQL&Oracle database systems.

## System Requirement Constraints

.

# system overview

This section describes the overview of the message propagation. Messages can be propagated from one queue to another, allowing applications to communicate with each other without being connected to the same database or to the same queue. The destination queue can be a heterogeneous remote database. It provides high performance, reliable message propagation between heterogeneous databases such as Oracle & MySQL databases. The following diagram shown the interaction of the message propagation.

MySQL Advance queues

Propagation

Propagation

Oracle

Advance queues

MySQL Advance queues

Propagation

MySQL Advance queues

Figure 1: Message propagation illustration

A message is marked as processed in the source queue immediately after the message has been propagated, even if the consumer has not dequeued the message at the remote queue. Similarly, when a propagated message expires at the remote queue, the message is moved to the exception queue of the remote queue, and not to the exception queue of the local queue. The AQ system does not currently propagate the exceptions to the source queue.

# system architecture

This section details the design of message propagation for distributed heterogeneous databases.

## System Architecture Design

The message propagation is deisgned to imitate the funcationalities of Oracle AQ propagation and can propagate messages between heterogeneous databases such as MySQL & Oracle. The majority features is based on the Oracle AQ propagation mechanism. Serveral modules are defined to meet the requirements of the propagation such as:

1. Propagation pool management;
2. Propagation thread/channel work item;
3. Propagation status statistics;
4. Interaction with heterogeneous databases
5. Interaction with remote destination queue by communication module.

The design of above modules can facility the funcationalities which needed to realize the message propagation mechanism for heterogeneous databases. The overview of interaction for these modules are shown below:

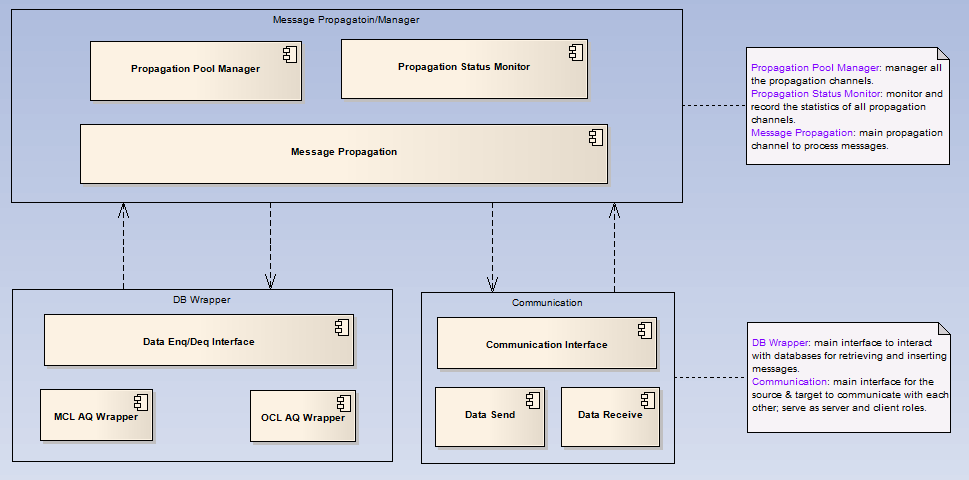


Figure 2: Message Propagation Architecture

## Module Decomposition Design

This section provides the decomposition structure of the message propagation for heterogeneous database systems. Based on the existing Oracle AQ propagation features, three main modules are introduced for main funcationalities and two modules for management such as:

1. Message Retrieve;

▪ To retrieve message data from the source queue by predefined design mechanism. The message retrieve strategies is to filter the message base on the message recipients and each recipient has it own channel to process the message.

1. Message Dispatch/Communication;

▪ To propagate message data to destination queue by predefined communication protocol. Currently the following protocols is required in the system:

1. Connection request
2. Data message request
3. Acknowledge reply
4. Connection request reply: request success; request exception
5. Data message request reply: request success; request exception

Based on the above communication protocols, there are three option communication tools and the following table shown the advantages and drawbacks of these tools:

|  |  |  |
| --- | --- | --- |
| Item | Advantages | Drawbacks |
| ACE | ▪ mature ▪ easy to use ▪ high performance  ▪ open source | ▪ need ACE libs |
| Corba | ▪ mature ▪ easy to use ▪ open source | ▪ need Corba libs  ▪ need know Corba IDL object |
| TCP socket | ▪ high performance, use OS APIs ▪ mature | ▪ need know TCP socket |

1. Message Parse/Apply.

▪ To parse the message from communication channel and apply to destination queue.

1. Propagation Pool Manager

▪ To manager all the message propagation channels.

1. Propagation Status Monitor

▪ To gather propagation status statistics to easy management and maintainence.

## Design Rational

The message propagation for heterogeneous database systems is based on the Oracle AQ propagation technologies.

## Related cots

The related cots of the message propagation listed below:

mcl --- mysql connection library for interacting with MySQL database.

ocilib --- oracle connection library for AQ funcationalities support.

## Detail Design

### Sequence Diagram

This section elaborates the sequences of the message propagation library logic including dequeue data from source queue, propagate data to destination queue.

● **sequence of message propagation**:

1. message producers dequeue message from source queue based on the message recipient with FIFO order.
2. Data Producer launches data propagation and waits for acknowledge reply.
3. Message communication channel transfers data to target message consumer.
4. Message transit to target consumer by using low level communication mechanism such as TCP/IP.
5. Data consumer enqueue data into target message Queue.
6. Data consumer send an acknowledge response to data producer via propagation channel.
7. Data producer gets the acknowledge response from propagation channel.
8. Data producer commits the current dequeue operation and go to step 1 for starting a new propagation transaction.

The following diagram illustrated the message propagation flow:

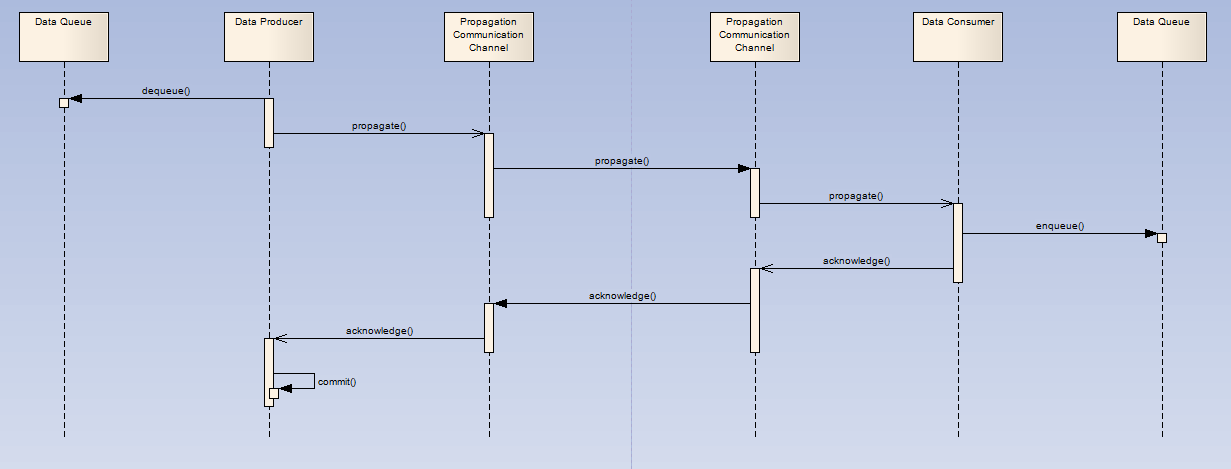


Figure 3: Propagation sequence

As the above diagram shown: A simplified two-phase commit protocol is introduced that the data producer needs to get acknowledge reply from data consumer, then data producer determines whether it should go to next propagation operation or retry current propagation operation. The following scenarios needs to be addressed:

1. Data should be enqueued only once;

Data in queue has an unique message ID, it makes sure that no duplicated data exist in data queue; but if other process dequeue data, it may cause that duplicated data are enqueued.

1. Data Consumer enqueue successfully, but the acknowledge message is failed to send to data producer;
2. Data producer is crashed and failed to commit data;
3. Data consumer get data from propagation communication channel , but routine in the data consumer hang up;
4. Data consumer cannot enqueue data into its data queue.

For item 1,2,3 mentioned above, a message duplicate strategy is introduced to hanle these considerations; for item 4 & 5, error handling will handle these considerations. Please refer to section 5.5.7 for a detail description of the error handling.

### Threading and Concurrence

This section describes the solution of the threading and concurrence of the message propagation module. It’s details the mechanism of the message retrieve and dispatch. The message retrieve or the so called dequeue strategy is based on the recipient list of the queue. Each recipient/agent subscribes its own interesting messages identified by the agent. The following diagram illustrated the retrieve mechanism.

Recipient1

Message

Buffers

Recipientn

The above diagram shown that each message recipient has its own filter and dequeue message from the source queue.

For the propagation channel, the number of the propagation channels are depend on the recipient list of the source queue. It creates propagation channel for each recipient based on each queue. For example, if there are two queues have the same recipient address, two propagation channels will be created for each queue.

### Message Protocol

This section defines and details the message protocol which used in the message propagation system. There were three kinds of messages traveral through distrubted heterogeneous databases.

1. **Connection request**

When connection request sent to destination server, the following info are defined to send to destination server for processing the request.

▪ Queue name – the destination queue name which need to accept data from source queue

▪ Agent name – the agent name which the message belong to

▪ Ip address – the Ip address of the sender/source queue

▪ hostname – the hostname of the sender/source queue

1. **Message request**

For message request to the destination queue, the following info are defined to send to the destination queue for processing the message request.

▪ Queue name – the queue name which the message belong to

▪ OMUID – the orignal message unique ID

▪ OMSID – the orignal message sequence ID

▪ Priority – the message priority of the message

▪ length – the length of the message payload data

▪ payload – the message payload data

1. **Acknowledge reply**

For each request above, the server will issue response according to the request.

Acknowledge reply for connection request:

▪ Connection code – the response code of the connection request

0 – success.

1 – cannot accept the connection because there was a duplicate connection.

2 – cannot accept the connection because the queue which the connection specified doesn’t exist.

3 – cannot accept the connection because other unknown exception occurred.

Acknowledge reply for message request:

▪ Package ID – the unique ID of the message package which received from source

▪ Message count – the number of messages which successfully processed by target queue

▪ MMSID – the maximum message sequence ID which processed by target queue

▪ Response code – the response code of the message resquest

0 – success

25207 – message enqueue failed because of enqueue option disabled.

25215 – message enqueue failed because of user\_data type and queue type do not match.

03135 – message enqueue failed because of database connection lost/offline.

70011 – message enqueue failed because of disk space full.

88888 – message enqueue failed because of other unknown exceptions occurred.

### Class Diagram

The interaction of the message propagation classes shown below:

### Componment Diagram

### Dependencies

### Error Handling

Propagation has built-in support for handling failures and reporting errors. Currently, there are two kinds of error handing such as:

1. Propagation uses an exponential backoff scheme for retrying propagation from a schedule that encountered a failure. If a schedule continuously encounters failures, then the first retry happens after 30 seconds, the second after 60 seconds, the third after 120 seconds and so forth. If the retry time is beyond the expiration time of the current propagation schedule, then the next retry is attempted at the start time of the next schedule. A maximum of 16 retry attempts is made, after which the propagation schedule is automatically disabled.

* The specified recipient address is invalid;
* Remote database is unavailab;
* Remote queue is not enable for enqueuing;
* Remote database server disk space full.

2. Propagation try 5 times for enqueuing and the interval time is 30 seconds, then the message will put into the exception queue for further recovery/reference.

* Message format invalid;
* Payload type not match.

### Propagation Clustering

The propagation proxy will be running and deploy at two different kinds of platforms: Center CMS server and Station SMS server. At center CMS server, the propagation proxy are running as stateless and being connected by all the station SMS propagation proxies. More than one propagation proxies will be deployed at center and two propagation proxies will deployed at stations. Therefore, a clustering mechanism need to design for those propagation proxies at both center and stations. The following sections details the clustering mechanism in two mode: normal mode and exception mode.

* Normal mode
* Station

During normal mode, the propagation proxies are controlled by systemController and only one proxy running in Control mode. No additional strategy needed to apply at station proxies.

* Center

As all the propagation proxies are running as stateless and the OS is Sun Solairs, so the cronjob of the OS can be used for the failed over mechanism. Another strategy is needed to manager the connections from stations for the purpose of loading balance.

* Exception mode

### Configuration Information

## Testability

## Maintainability

## Impact Analysis

## Deployment Considerations

## Flow-on Effects

## Extensibility and Reuse

# gui-component design information

# Appendix A: Software module design review

| **DTL (C955) Design Module Review** | | | **Date:** | | |
| --- | --- | --- | --- | --- | --- |
| **Doc. No.:** | | | | | |
| **Subject :** | | | | | |
| **Reviewer** : | | | | | |
| **Status** : | | | | | |
|  |  |  | | |  |
| **Status :** | 1.Draft | 2. Conditional Accepted | | 3. Accepted | 4. Rejected |

| **No** | **Ref** | **Page No** | **Comments** | **Response** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Review Outcome (completed immediately following the review)

| **Signature of Reviewer**  **Name** |  | **Date:** |
| --- | --- | --- |
| **Signature of Independent Verifier**  **Name** |  | **Date:** |

Review Closeout (completed when review actions are completed)

| **All actions completed and closed** | **✓/🗶** | |
| --- | --- | --- |
| **Signature**  **Name** |  | **Date:** |