Warrants Data Exchange NCSC Grant

ARCHITECTURE DESIGN DOCUMENT

Version 1.0, 11/19/2012

Revision History

Version	Date	Author(s)	Revision Notes
1.0	11/19/12	Mark Fontana	Initial version.

Table of Contents

1 Introduction

1.1 System Overview

In 2005, at the direction of the Governor's Office, the Pennsylvania Justice Network (JNET), the Pennsylvania State Police (PSP), and the Administrative Office of Pennsylvania Courts (AOPC), began to develop an electronic warrant issuance process for all Magisterial District Judge and Common Pleas warrants. This directive was issued in response to concerns that approximately 60% of all warrants issued statewide were not being entered into state and federal warrant repositories. The primary objectives of this project were to collect, secure, automate, and enter all warrants into PSP's Commonwealth Law Enforcement Assistance Network (CLEAN) and the FBI's National Crime Information Center (NCIC) at the time of Judicial approval. Through this process, law enforcement personnel benefit in two significant ways; a reduction in data entry at the time of issuance, and the knowledge of all active warrants for subjects they encounter.

From a system perspective, all of the data mapping, data scrubbing, and data conversion required to ensure successful processing at NCIC is carried out on PSP-maintained components. Recently, JNET has been asked to migrate as much of that functionality to its system components, and to assess what changes can be made to improve the rate of successful automated entry of valid warrants data into NCIC and CLEAN. Around the same time (late Summer 2011), JNET applied for a grant from the National Center for State Courts (NCSC) who were seeking State participation in the development of a Warrants and Disposition Management Toolkit. In late April 2012, JNET Learned that it had been one of the States awarded a grant from NCSC.

1.2 Scope

The primary focus of this document is the components JNET will need to develop.

The AOPC and PSP components are not discussed in detail.

2 Design Considerations

2.1 Critical Requirements

1. Successfully migrate the data mapping, data scrubbing and data conversion functionality of the automated warrants data transfer system from PSP system components to JNET system components.
2. Improve the rate of successful initial entry of warrants into NCIC and CLEAN from the current level.
3. Improve the rate of successful cancelations of warrants from NCIC and CLEAN from the current level.
4. Add the capability to return a NIC to AOPC when warrant data is manually entered by Servicing ORIs.
5. Improve reporting so that local stakeholders can analyze and take action on reasons for warrants data rejections.
6. Create Warrant Correction UI that will allow authorized users to correct and resubmit rejected warrant entries.

2.2 Dependencies

The design of this system reuses the following JNET system components:

- JNET Utility Services (JUS)
- JNET Error Handling Framework (JEHF)
- JNET Auditing Service (JAS)
- JNET Common Services

3 Architectural Design

3.1 Conceptual Architecture

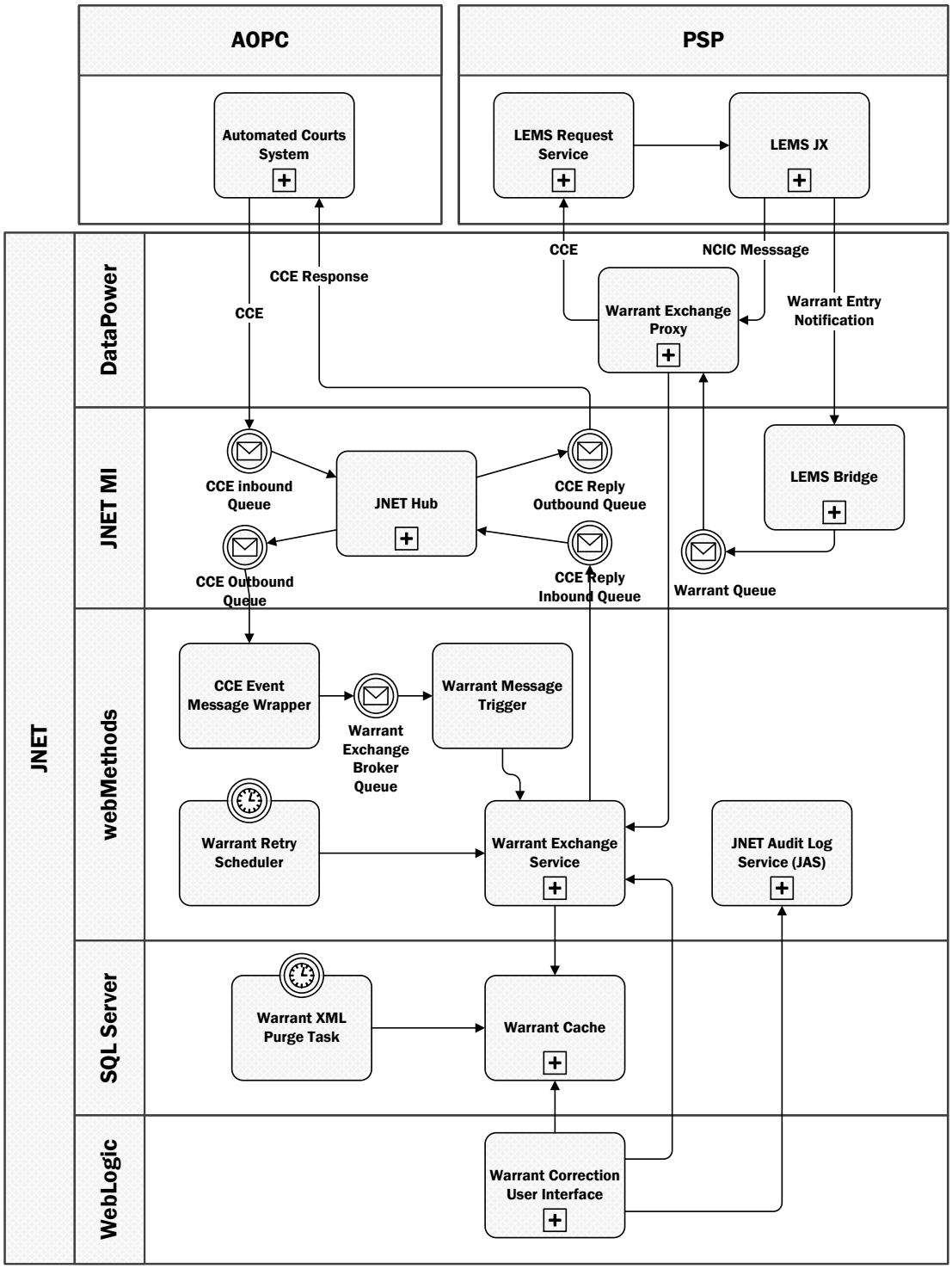


Figure 1 – Automated Warrants Entry Conceptual Architecture

3.1.1 System Components

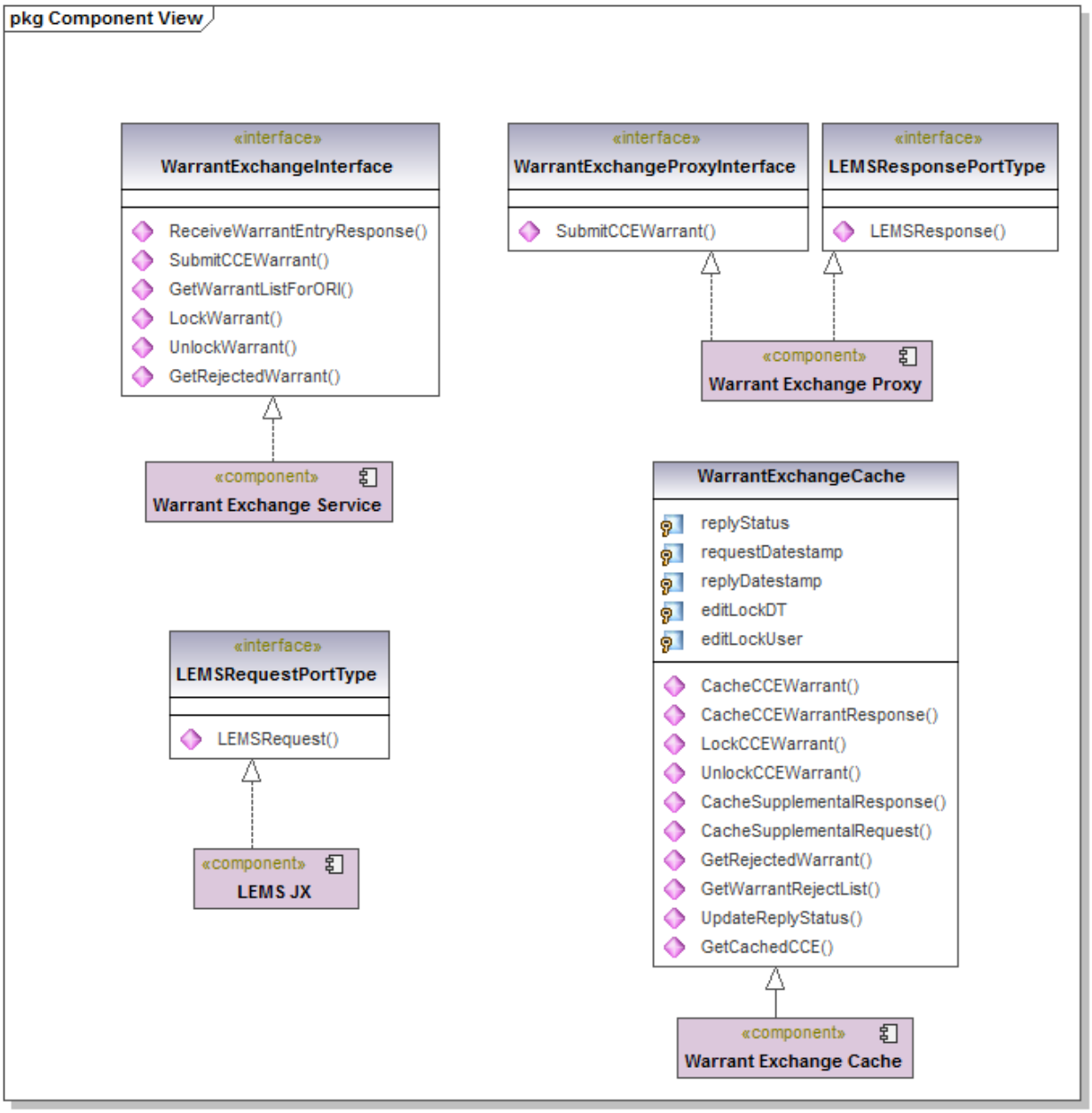


Figure 2 - System Component Interfaces

Table 1 – System Components

Component	Description
Automated Courts System	The Automated Courts System publishes Court Case Event (CCE) messages to the JNET MI at predefined events in the court case lifecycle. Included events are warrant initiation, warrant service and warrant

	cancellation.
JNET Hub	The JNET Hub is the primary message dispatcher within the traditional JNET Messaging Infrastructure (MI).
CCE Inbound/Outbound Queue	The CCE Inbound Queue receives CCE messages from AOPC. The CCE Outbound Queue contains CCE messages to be processed by the JNET Event Messaging Service (JEMS).
CCE Response Inbound/Outbound Queue	The CCE Response Inbound Queue receives CCE warrant responses from the Warrant Exchange Service. The CCE Response Outbound Queue contains CCE warrant responses destined to AOPC.
Warrant Error Correction Application	The Warrant Error Correction Application displays a list of rejected warrants a user is allowed to correct based on their ORI. The user can select a warrant from the list, make the necessary corrections and resubmit it to CLEAN/NCIC.
CCE Event Message Wrapper	The CCE Event Message Wrapper is an existing JEMS component that listens for CCE messages on the CCE Outbound Queue and publishes them to JEMS. This component will be modified to make an additional publication to the Warrant Exchange Broker Queue when the CCE Action indicates a warrant publication.
Warrant Exchange Broker Queue / Warrant Message Trigger	The Warrant Exchange Broker Queue is a new queue on the webMethods broker used to facilitate guaranteed delivery of CCE warrants to the Warrant Exchange service.
Warrant Retry Scheduler	The Warrant Retry Scheduler runs periodically to resubmit Warrant Cache entries in a pending or retry state to the Warrant Exchange Service.
Warrant Exchange Service	The Warrant Exchange Service provides the main processing logic for the warrants data exchange.
Warrant Exchange Cache	The Warrant Exchange Cache provides temporary storage of warrant requests and responses as they are processed through the automated warrants system. It also provides long term storage of reporting data.
Warrant XML Purge Task	The Warrant XML Purge Task is an SQL Server scheduled task that removes warrant XML from the Warrant Exchange Cache that are older than the configured cache lifetime.
Warrant Exchange Proxy	The Warrant Exchange Proxy proxies warrant entry requests and response with LEMS JX. It also provides the data scrubbing functionality.
JNET Error Handling Framework (JEHF)	The JEHF provides a common JNET framework for logging errors and notifying users
JNET Audit Log Service (JAS)	The JAS provides standard methods for writing to the JNET Audit Log.
LEMS Request Service	The LEMS Request Service is a web service used to submit messages to the LEMS JX.
LEMS JX	LEMS JX is the PSP message switch used to broker messages between law enforcement data systems including NCIC and the PSP Hot Files.

3.2 Logical Architecture

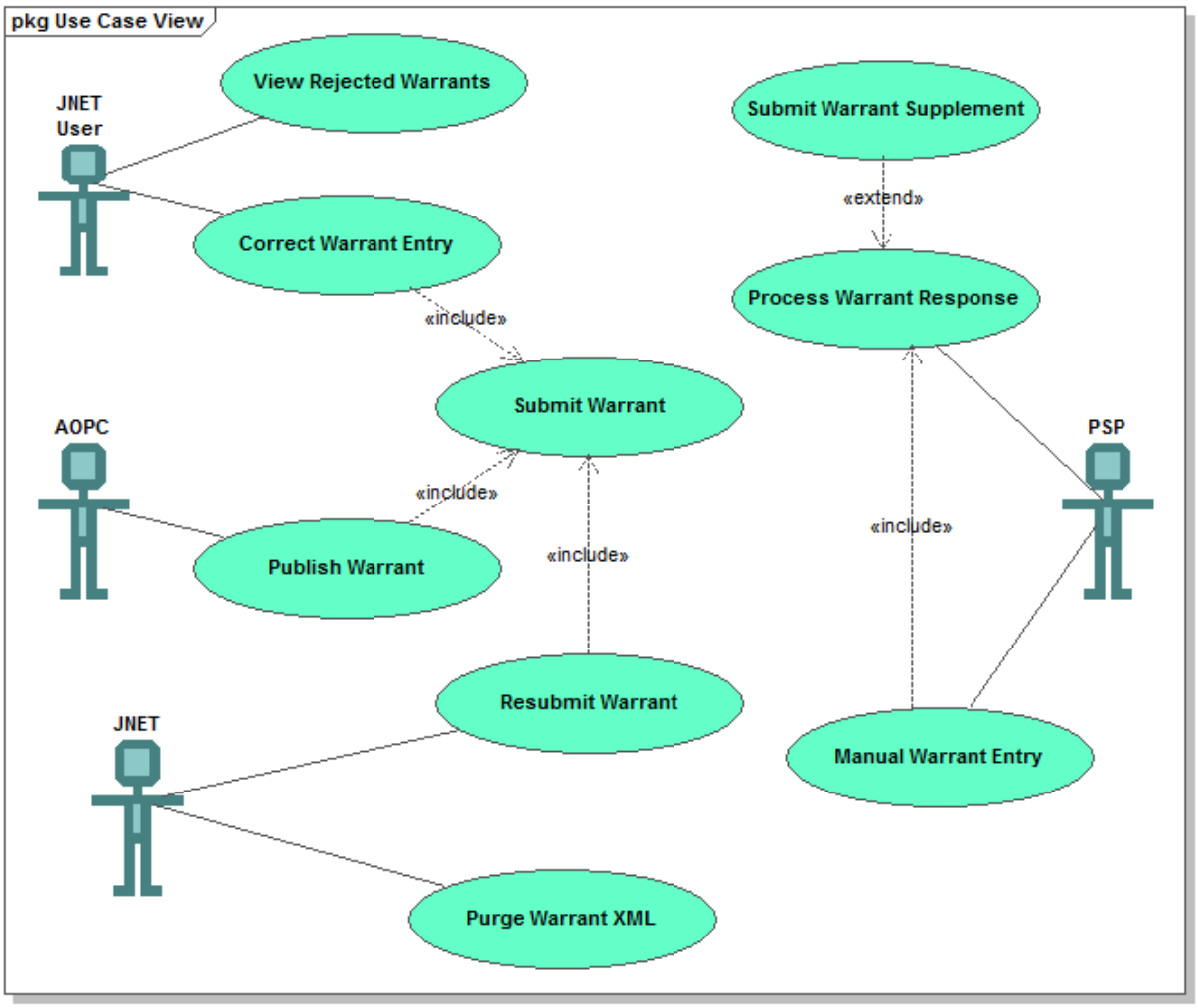


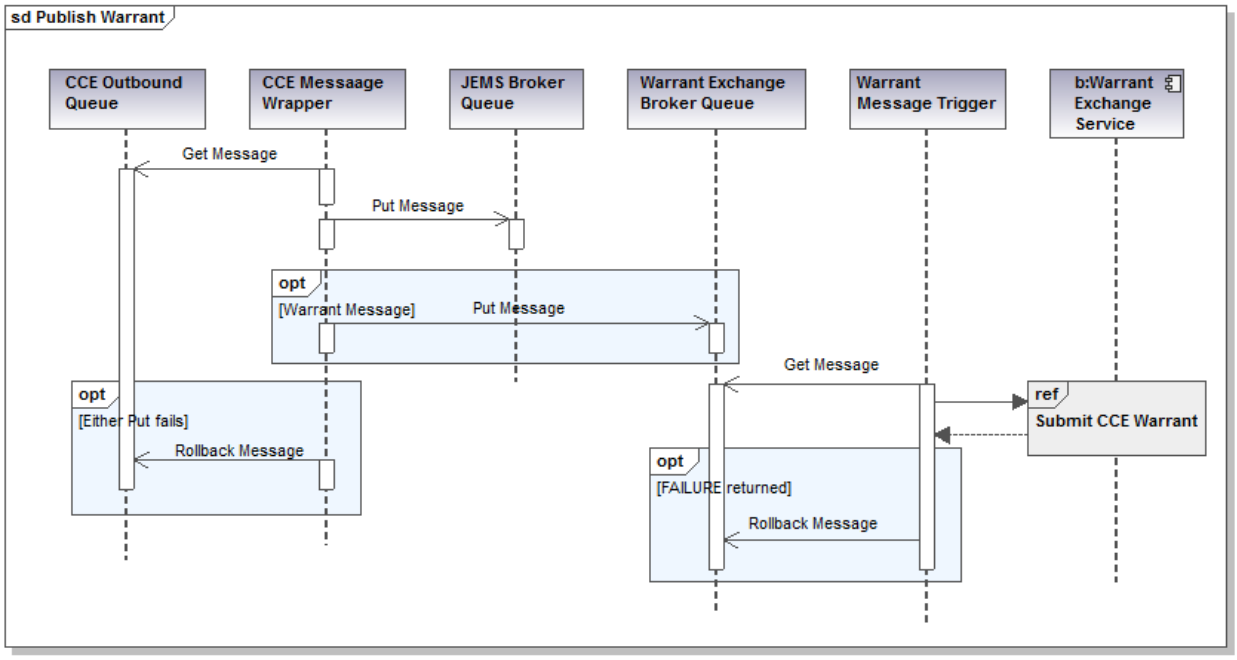
Figure 3 - System Use Cases

3.2.1 Publish Warrant

AOPC publishes a CCE warrant to JNET.

JNET sends the CCE warrant to PSP.

JNET saves the CCE XML and PSP acknowledgement status in the Warrant Exchange Cache.



3.2.1.1 Primary Flow

1. A CCE message published by AOPC arrives on the CCE Outbound Queue.
2. The existing JEMS CCE Message Wrapper gets the next message from the CCE Outbound Queue.
3. The CCE Message Wrapper puts the CCE message on the existing JEMS broker queue.
4. IF the CCE Message action indicates a warrant message THEN the CCE Message Wrapper puts the CCE message on the Warrant Exchange Broker Queue.
5. The Warrant Message Trigger gets the next message from the Warrant Exchange Broker Queue.
6. The Warrant Message Trigger calls the SubmitCCEWarrant() operation on the Warrant Exchange Service.
7. Include 3.2.2 Submit Warrant.

3.2.1.2 Alternative Flows

3.2.1.2.1 Put Message Failure

1. In steps 3 or 4, the CCE Message Wrapper fails to put the CCE Message on the JEMS or Warrant Exchange broker queues.
2. The CCE Message Wrapper rolls back the CCE message to the CCE Outbound Queue.

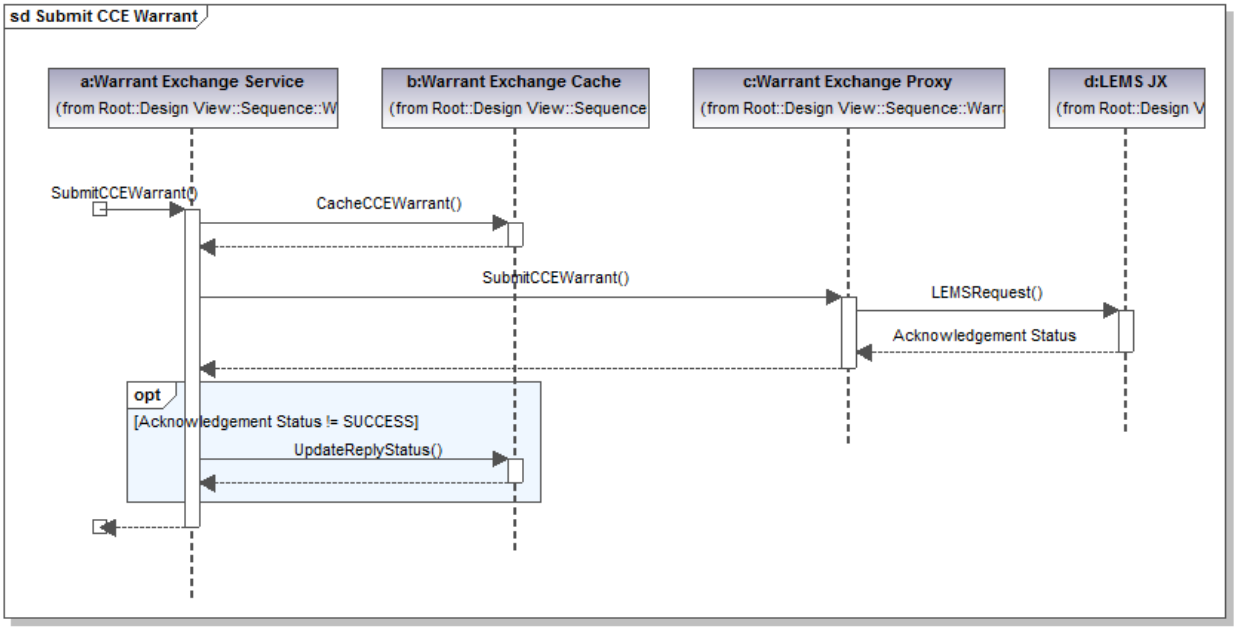
3.2.1.2.2 Submit CCE Warrant Failure

1. In step 6, the Warrant Exchange Service returns FAILURE.
2. The Warrant Message Trigger rolls back the CCE message to the Warrant Exchange Broker Queue.

3.2.2 Submit Warrant

JNET sends the CCE warrant to PSP.

JNET saves the CCE XML and PSP acknowledgement status in the Warrant Cache.



3.2.2.1 Primary Flow

1. The Warrant Exchange Service receives a SubmitCCEWarrant() request.
2. The Warrant Exchange Service calls CacheCCEWarrant() on the Warrant Exchange Cache.
3. The Warrant Exchange Cache saves the CCE XML with replyStatus = 'PENDING'.
    a. IF an automated retry THEN update status on latest exchange record only.
    b. IF a warrant correction THEN create a new exchange record on existing warrants record.
4. The Warrant Exchange Service submits the CCE to the Warrant Exchange Proxy on DataPower.
5. The Warrant Exchange Proxy scrubs invalid data and incomplete data sets from the CCE message.
6. The Warrant Exchange Proxy wraps the CCE message in the LEMSRequest() envelope and calls the LEMSRequest service on LEMS JX.
7. LEMS JX transforms and validates the request and queues it to NCIC and PA Hot Files.
8. LEMS JX returns SUCCESS or SUCCESSWITHWARNING acknowledgement status.
9. The Warrant Exchange Service returns a SUCCESS status.

3.2.2.2 Alternative Flows

3.2.2.2.1 Reject Status

1. In step 7 LEMS JX returns FAILURE acknowledgement status or a client fault.
2. The Warrant Exchange Proxy updates the replyStatus to REJECT.

3.2.2.2.2 Retry Status

1. In step 7 LEMS JX returns RETRY acknowledgement status or a server fault.
2. The Warrant Exchange Proxy updates the replyStatus to RETRY.

3.2.2.2.3 Cache Failure

1. In step 2 the Warrant Exchange Service fails to save the CCE to the Warrant Exchange Cache.
2. The Warrant Exchange Service returns FAILURE status.

3.2.3 Process Warrant Response

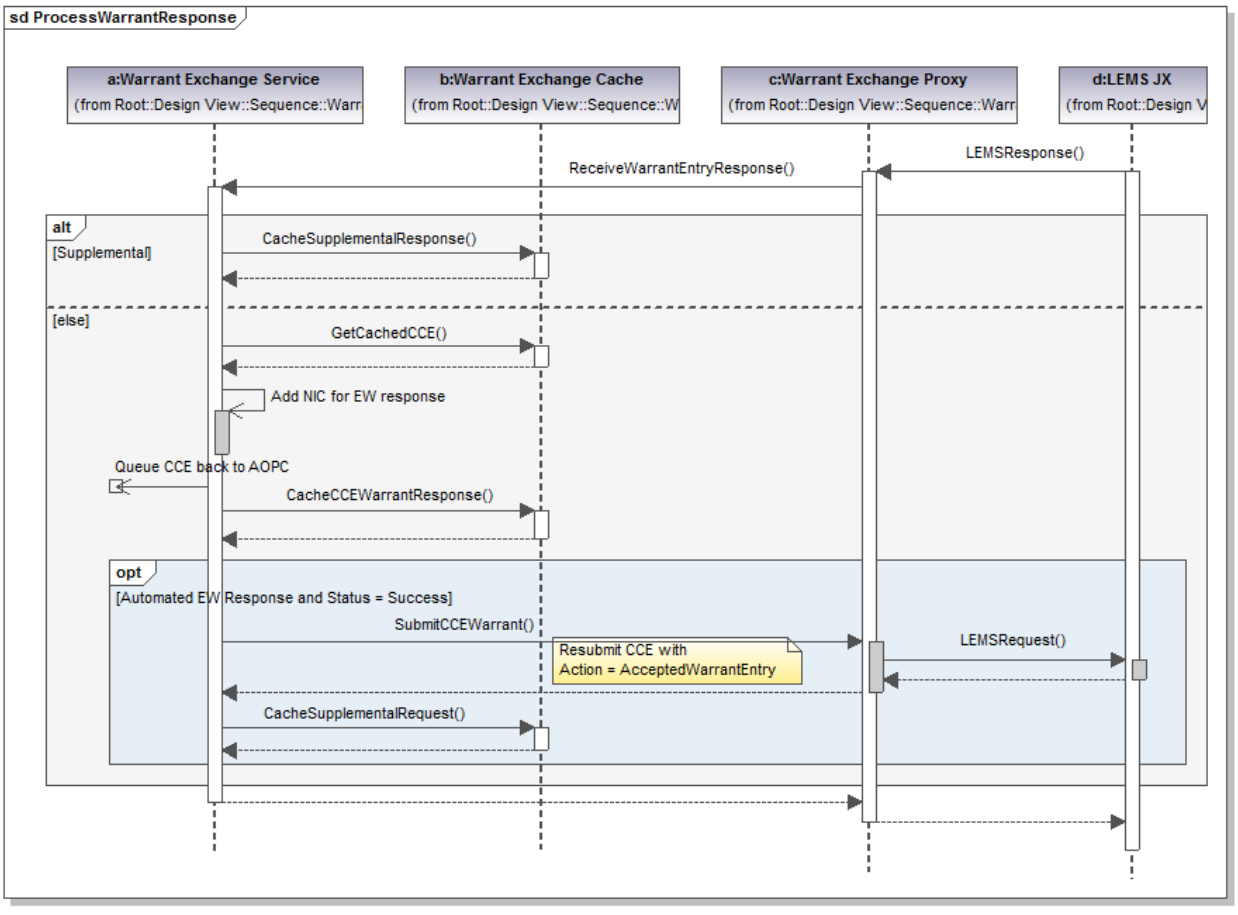
PSP sends warrant response to JNET.

JNET saves the warrant response and response status in the Warrant Cache.

JNET publishes the warrant response to AOPC.

In the case of a warrant entry success response, JNET adds the NIC number to the CCE response to AOPC.

In the case of a warrant entry success response, resubmits the CCE Warrant to PSP as a supplemental entry.



3.2.3.1 Primary Flow

1. LEMS JX receives a response from NCIC and Hot Files after processing a CCE warrant request from JNET.

2. LEMS JX sends the NCIC response message to the Warrant Exchange Proxy on DataPower.
3. The Warrant Exchange Proxy transforms the NCIC response and sends it to the Warrant Exchange Service on webMethods.
4. The Warrant Exchange Service determines the response is NOT from a supplemental request.
5. The Warrant Exchange Service gets the originating CCE message from the Warrant Exchange Cache using the warrant number from the response message.
6. The Warrant Exchange Service queues the CCE message along with the request status back to AOPC.
    a. If the response was from a warrant entry (EW) then the Warrant Exchange Service adds the NCIC number to the CCE message.
7. The Warrant Exchange Service updates the Warrant Exchange Cache with the response XML and response status.
8. The Warrant Exchange Service returns SUCCESS status to the Warrant Exchange Proxy.
9. The Warrant Exchange Proxy returns SUCCESS status to LEMS JX.

3.2.3.2 Alternative Flows

3.2.3.2.1 Receive Supplemental Response

1. In step 8 the Warrant Exchange Service determines the response is from a supplemental request.
2. The Warrant Exchange Service saves the supplemental response to the Warrant Exchange Cache.
3. Processing continues at step 8.

3.2.3.2.2 Send Supplemental Message

1. The NCIC response message indicates a warrant entry (EW) success and the response not from the CLEAN Warrant Entry Notification.
2. The use case is extended by Submit Warrant Supplement.

3.2.3.2.3 Cache Failure

1. In step 6 the Warrant Exchange Service fails to update the Warrant Exchange Cache.
2. RETRY status is returned in steps 8 and 9.

3.2.4 Publish Warrant Response

JNET publishes the warrant response status along with the original CCE to AOPC.

In the case of a warrant entry success response, JNET adds the NCIC number to the CCE response to AOPC.

3.2.5 Submit Warrant Supplement

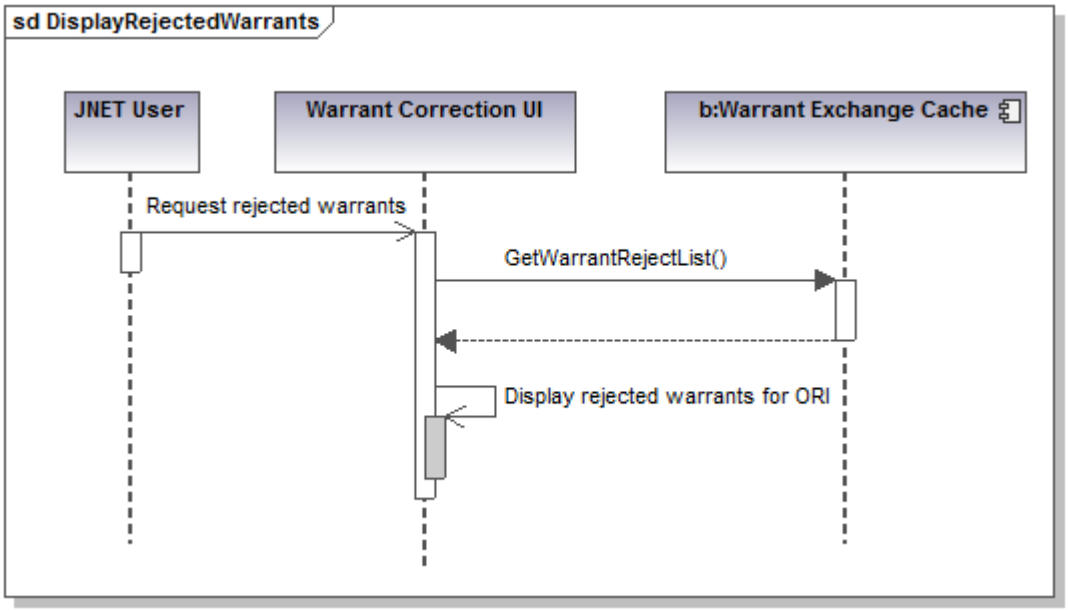
JNET receives a successful CCE response for a Wanted Person Entry from PSP.

JNET resubmits the originating CCE message to PSP after changing the action to Warrant Supplement.

PSP sends the appropriate supplemental entries to Hot Files and NCIC.

3.2.6 View Rejected Warrants

The Warrant Correction UI displays a list of rejected warrants a JNET user is allowed to edit based on that user's ORI.

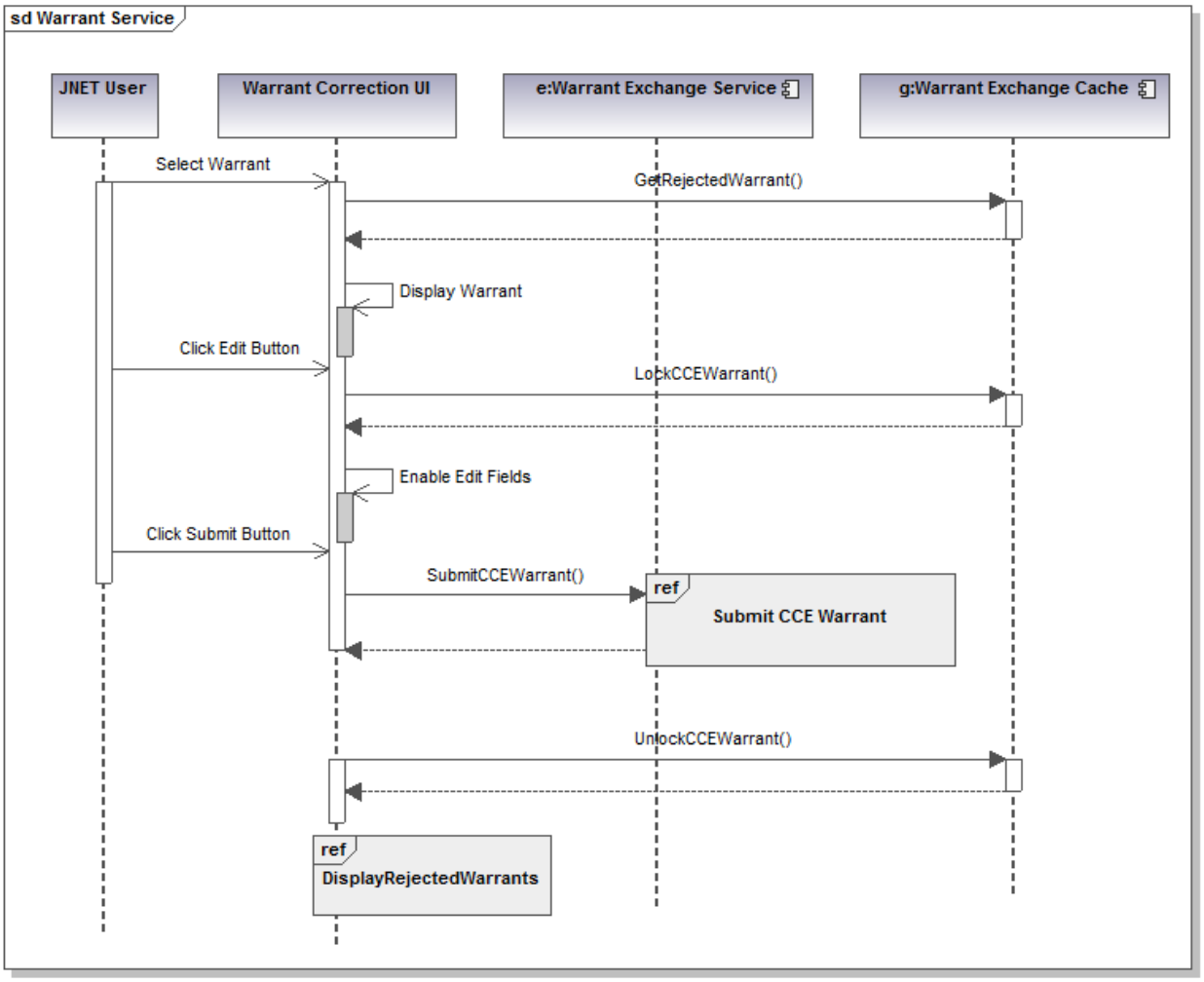


3.2.6.1 Primary Flow

1. An authorized user opens the Warrant Error Correction Summary page in the Warrant Correction UI.
2. The Warrant Correction UI calls the GetWarrantListForORI operation on the Warrant Exchange Cache with the user's derivative ORI from the LDAP.
3. The Warrant Exchange Cache determines the user's authorized ORIs from the user's derivative ORI.
4. The Warrant Exchange Cache selects all warrant entries for the user's authorized ORIs that are in REJECT status and the request date and time is less than 72 hours old.
5. The Warrant Exchange Cache returns the warrant list to the Warrant Correction UI.
6. The Warrant Correction UI displays the warrant list to the user.

3.2.7 Correct Warrant Entry

The Warrant Error Correction Application allows a JNET user to make corrections to a rejected warrant entry and resubmit the warrant to PSP.



3.2.7.1 Primary Flow

1. An authorized user selects a warrant to edit in the Warrant Correction UI.
2. The Warrant Correction UI calls the Warrant Exchange Cache to get the rejected warrant details.
3. The Warrant Correction UI displays warrant details with the edit fields disabled.
4. The user selects the Edit button.
5. The Warrant Correction UI calls LockCCEWarrant() on the Warrant Exchange Cache.
6. The Warrant Exchange Cache verifies the warrant record is not locked by another user.
    a. NOT(editLockUserID != user AND editLockDatetime < now() – 60 minutes)
7. The Warrant Exchange Cache locks the warrant record for the user
    a. editLockUserID = user, editLockDatetime = current date and time.
8. The Warrant Correction UI enables the edit fields.
9. The Warrant Correction UI enables the Submit button when the user edits a field.
10. The user makes the desired corrections and clicks the Submit button.
11. The Warrant Correction UI calls SubmitCCEWarrant() on the Warrant Exchange Web Service.
12. Include 3.2.2 Submit Warrant.
13. The Warrant Exchange Web Service returns SUCCESS.

14. The Warrant Correction UI calls UnlockCCEWarrant() on the Warrant Exchange Cache.
15. The Warrant Exchange Cache unlocks the warrant record.
    a. editLockUserID = NULL, editLockDatetime = NULL
16. The Warrant Correction UI includes View Rejected Warrants.

3.2.7.2 Alternative Flows

3.2.7.2.1 Record Locked

1. In step 6 the Warrant Correction UI does not verify the warrant record is not locked by another user.
2. The Warrant Correction UI notifies the user and does not enable the edit fields and Submit button.

3.2.7.2.2 User Cancel

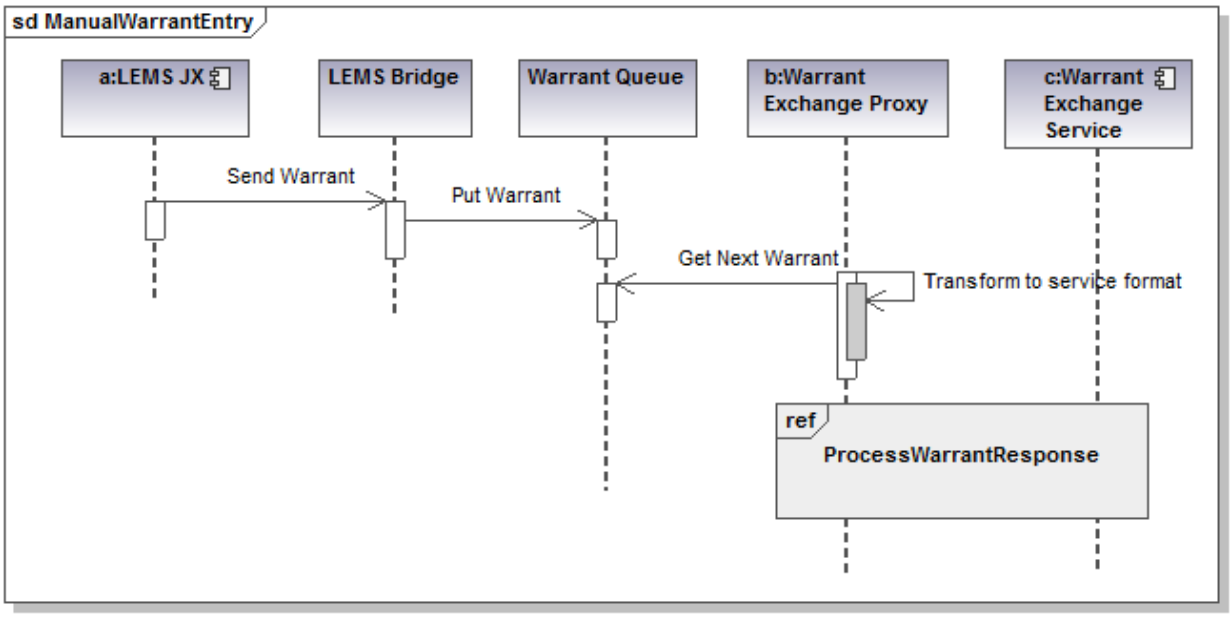
1. The user clicks browser back button or closes the window after warrant record has been locked.
2. The Warrant Correction UI calls UnlockCCEWarrant on the Warrant Exchange Cache.

3.2.8 Manual Warrant Entry

PSP publishes wanted person entries to JNET.

JNET updates the status of matching warrant entries in the Warrant Cache not in a success state.

JNET publishes the warrant status back to AOPC for all updated warrant entries.



3.2.8.1 Primary Flow

1. A warrant is manually entered on CLEAN.
2. LEMS JX sends a warrant notification to the LEMS Bridge on the JNET MI.
3. The LEMS Bridges puts the warrant notification on the Warrant Queue.

4. The Warrant Exchange Proxy removes the top record from the Warrant Queue.
5. The Warrant Exchange Proxy transforms the warrant notification message into the NCIC XML response format.
6. Include 3.2.3 Process Warrant Response.

3.2.8.2 Alternative Flows

3.2.9 Resubmit Warrant

At regular intervals, JNET resubmits entries in the Warrant Cache that are in a retry state or a pending state over 2 hours.

3.2.10 Purge Warrant XML

At regular intervals, JNET removes warrant XML from the Warrant Cache that are older than a configurable time period. Statistical information remains in the Warrant Cache for reporting purposes.

4 Infrastructure Impact Analysis

4.1 Network Architecture

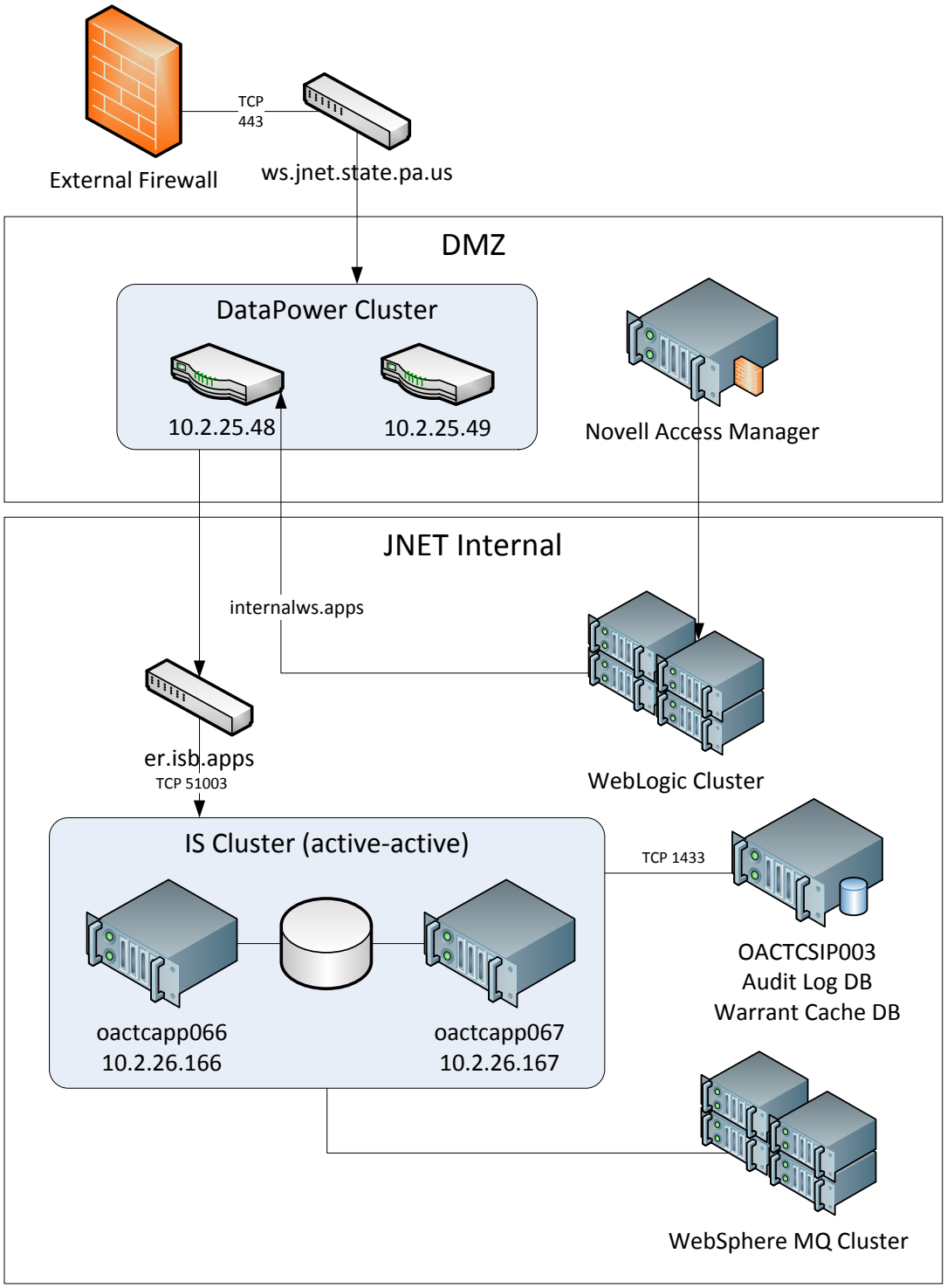


Figure 6 – Network Architecture

4.2 Availability Considerations

High availability and failover is provided through the following features:

- A Load balancer/Content switch in front of the two ISB servers providing Active-Active IS cluster.
- A Load balancer/Content switch in front of two DataPower appliances providing Active-Active DataPower cluster.
- The JNET Audit_Log and Warrant Exchange Cache DB are hosted on MS SQL Server utilizing the SQL Server Cluster capabilities for failover.

5 Security Design

5.1 AOPC Interface

AOPC Court Case Event Messages are sent and received over WebSphere MQ channels on an internal network.

5.2 LEMS JX Interface

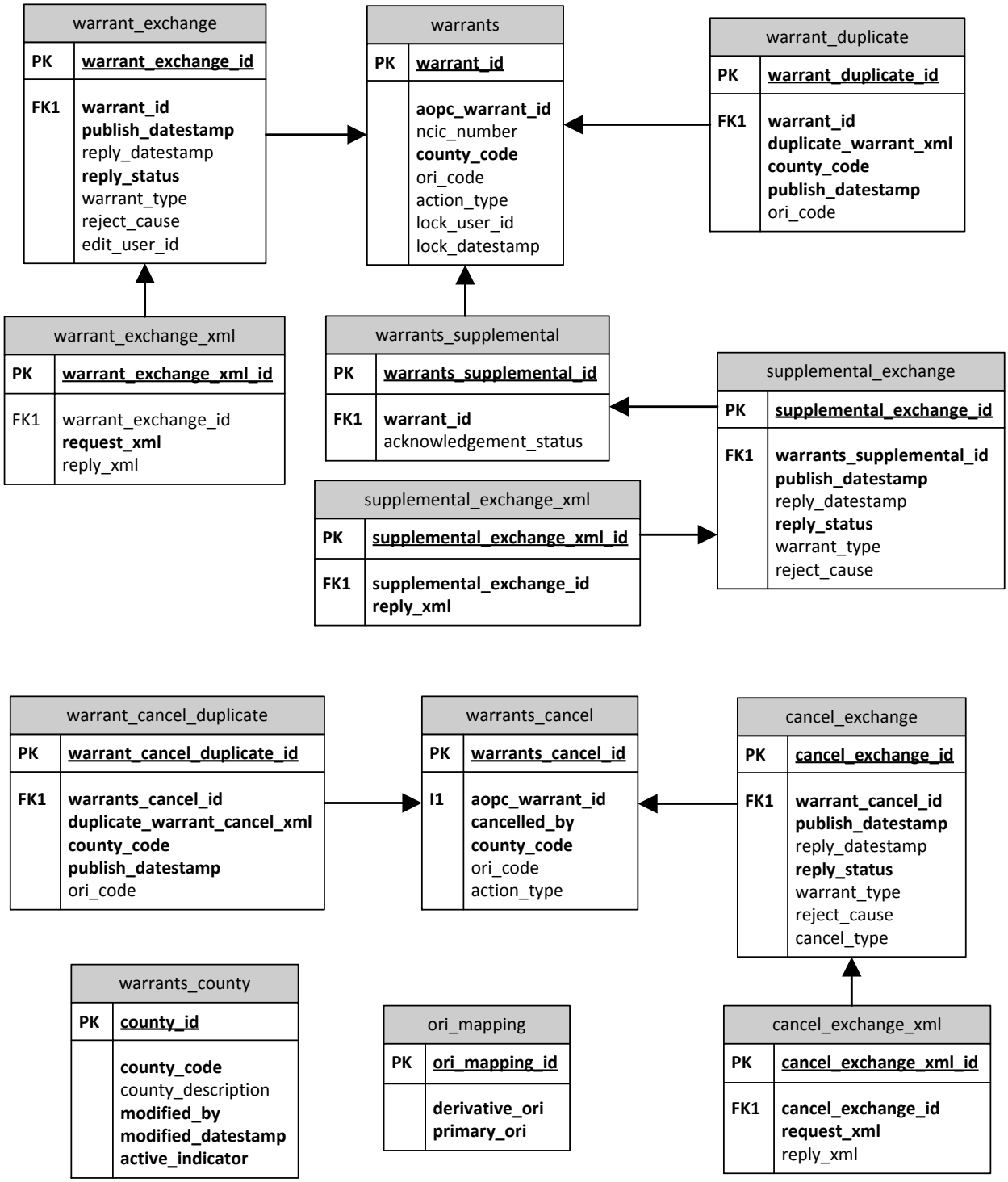
5.2.1 Encryption

Communications between the justice and public safety computer system consuming the service and the LEMS Request Service are encrypted using HTTPS with FIPS 140-2 compliant Transport Layer Security (TLS).

5.2.2 Mutual Authentication

Authentication of the justice and public safety computer system consuming the service and authentication of the LEMS Request and Response Service are accomplished by Transport Layer Mutual Authentication using X.509 certificates. The PSP must approve the certificate issuing authority of the client certificate presented by the justice and public safety computer system.

6 Data Design



7 External System Interface

7.1 AOPC Interface

The AOPC interface is defined in the AOPC Court Case Event Published IEPD-1.0.6O.

7.2 PSP Interface

The PSP Interface is defined in the following SSPs

- LEMS Request Service SSP v.3 0 0
- LEMS Response Service SSP v.3 0 0

8 Audit Logging

8.1 Warrant Submission

The Warrant Exchange Proxy will audit log the warrant submission to LEMS JX with the following fields:

Table 1 – Warrant Submit Audit Header

Field	Description
User_Id	'WARRANT'
User_Org	'AOPC'
System_Org	'AOPC'
AppName_Domain_Value_ID	'Warrant Exchange'
Module_Name	'warrant-request-mpg'
User_Def_Tracking_Num	Warrant Number
Action_Start	Time request sent to LEMS JX
Action_End	Time acknowledgement is received from LEMS JX
Action_Domain_Value_ID	UPDATE

Table 2 - Warrant Submit Audit Detail

Field_Name	Field_Value
AckStatus	Acknowledgement status code from LEMS JX
AckDescription	Acknowledgement status description from LEMS JX

9 Error Handling

9.1 JNET Error Handling Framework

All errors occurring in the DataPower and wed Methods ISB are logged to JEHF using the JEHF ISB service. For more detailed information, please the *JUS Handbook* or the *JNET Error Handling Framework usage guidelines*.

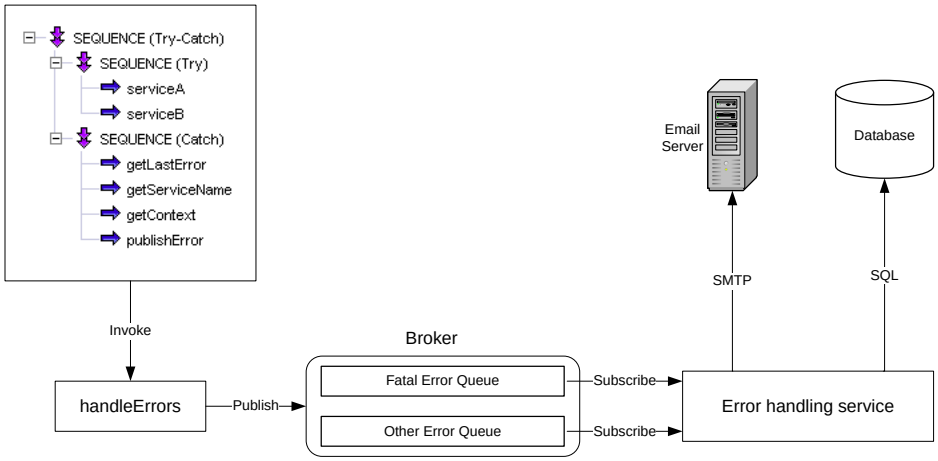


Figure 9 – Error Handling Framework

The above diagram illustrates the logical architecture of the error handling framework.