

nehta

Connectivity Co-ordinator

High Level Design

Version 1.1 draft — 30 June 2009

National E-Health Transition Authority Ltd

Level 25

56 Pitt Street

Sydney, NSW, 2000

Australia.

www.nehta.gov.au**Disclaimer**

NEHTA makes the information and other material (“Information”) in this document available in good faith but without any representation or warranty as to its accuracy or completeness. NEHTA cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

Document Control

This document is maintained in electronic form. The current revision of this document is located on the NEHTA Web site and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is of the latest revision.

Copyright © 2009, NEHTA.

This document contains information which is protected by copyright. All Rights Reserved. No part of this work may be reproduced or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without the permission of NEHTA. All copies of this document must include the copyright and other information contained on this page.

Table of contents

Table of contents	iii
Document information	iv
Change history	iv
1 Introduction	6
1.1 Purpose	6
1.2 Scope	6
1.3 Normative references	6
1.4 Definitions, acronyms, abbreviations	6
1.4.1 Terminology	7
1.5 Document Overview	7
2 Qualities and Requirements	8
3 Notes on Diagrams	9
3.1 Objects Referenced	9
4 Environment Overview	11
4.1 Infrastructure Services	11
4.1.1 UHI	11
4.1.2 NASH	11
4.1.3 Endpoint Locator Service	12
4.2 Interaction Types	12
5 Process View	13
5.1 Primary Use Cases	13
5.2 High level overview	14
5.2.1 Outgoing Service Requests	14
5.2.2 Incoming Service Requests	15
5.3 Service Consumer Processing	15
5.3.1 Gather Address Information	18
5.3.2 Retrieve Required Interactions	19
5.3.3 Process Interactions	21
5.3.4 Common Send Interaction Processes	27
5.4 Service Provider Processing	30
5.4.1 Process Secure Payload	32
5.4.2 Invoke Service Provider Code	32
5.5 Acknowledge Processing	33
5.6 Acknowledgement Processing	33
6 Engineering View	34
6.1 Design considerations	34
6.2 Core components	34
6.2.1 Library Components	34
6.2.2 Domain Client Components	35
6.2.3 Adapter Components	36
Appendix A: Informative references	37

Document information

Change history

Version	Date	Comments
1.0	2008-10-07	Extension of the Concepts of Operation document.
1.1 draft	2009-06-30	Changes to align to current understanding of connectivity environment.

This page is intentionally left blank.

1 Introduction

1.1 Purpose

The purpose of this document is to provide a non-normative high level software design for one possible approach of an implementation focused on being conformant with the NEHTA Connectivity related specifications. One of the primary goals of this document is to draw out and highlight some of the subtleties and design constraints of implementing these specifications in a real world operational environment. This approach will open the path to greater adoption and understanding of the specifications and ultimately lead towards the interoperability goals they are trying to achieve.

1.2 Scope

This document can be viewed as a complete high level software design for an implementation aiming to be compliant with the NEHTA Connectivity related specifications but **MUST NOT** be considered the only way to implement the specifications or as the suggested way to implement these specifications.

The high level software design described here will be focussed on a generic solution that can either encompass all package requirements with some package specific components or be extended to cater for package specific requirements only.

This document only describes design related to interactions for business functions. It does not include design for components or behaviour required for administration or management aspects of the E-Health environment (e.g. publishing endpoint details to a ELS).

The intended audience of this document include Architects and Software Developers. This document should be used to form a working understanding of the processes required to support interactions within the e-health environment, aspects of caching used to improve performance and key error handling scenarios required to be handled by an implementation.

1.3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For updated references, the latest edition of the referenced document (including any amendments) applies.

[RFC2119] IETF, RFC 2119: *Keywords for use in RFCs to Indicate Requirement Levels*, S. Bradner, March 1997, <http://ietf.org/rfc/rfc2119.txt>

1.4 Definitions, acronyms, abbreviations

CDD	Clinical Document Delivery
ELS	Endpoint Locator Service (previously SIL)
HPI-I	Healthcare Provider Identifier – Individual
HPI-O	Healthcare Provider Identifier – Organisation
IHI	Individual Healthcare Identifier
NASH	National Authentication Service for Health
NEHTA	National E-Health Transition Authority
UHI	Unique Health Identifier

1.4.1 Terminology

The keywords **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** in this document are to be interpreted as described in IETF's RFC 2119 [RFC2119].

1.5 Document Overview

The core of this document has been broken into three major sections, Environment Overview, Process View and Engineering View.

The Environment Overview section describes the environment within which implementations are expected to operate.

The Process View section describes the processes and activities that are required to support the operation of business interactions within the environment described by the previous section.

Engineering View section describes the key components and collaborations that are needed to support the processes described in the Process View section.

2 Qualities and Requirements

Some of the qualities and requirements that have driven this high level design include:

- Provide an endpoint that complies with NEHTA specifications
- Provide an endpoint that has the ability to comply with future NEHTA domain specifications
- Designed with extensibility in mind to allow addition of future endpoint specifications
- Designed to allow progressive implementation of National Infrastructure services
- Ability to cache expensive remotely located resources (UHI, ELS, Certificate entities)
- Built on foundation libraries to provide base components to build alternative implementations
- Common connectivity related tasks abstracted/simplified
- Deployable for service consumer and provider environment (container based)
- Deployable for service consumer only type environment
- Provide persistence capabilities for messages involved in interactions
- Initial implementation must support at least Clinical Document Delivery (CDD)

3 Notes on Diagrams

The activity diagrams that follow show both the control flow and the data object flow in unison. Object flow is distinguished by the flow line traversing between two or more square boxes contained on the outer edge of activities (activity parameters) or actions (object nodes). Where there is only one possible channel of object and control flow only the object flow is shown. From a processing perspective an activity (or action) can only be considered active if the control has been passed to it by the control flow and all input object flows to the activity have delivered their associated objects. Control flows may also be assigned guard constraints (labels within []) that must be satisfied before the control path can be taken. Guard constraints are only used where two control flows exit and action to show under what conditions the various paths are taken.

The state diagrams identify states that may be encountered while interacting with remote services. The 'Error' final state identifies that a rule based error (validation or constraint violation) condition has been encountered and the operation must not be retried without first taking corrective action. The 'Failure' final state identifies that an environment type error has been encountered and it is possible that if the environmental issue is rectified a retry of the operation may succeed.

The activity diagrams should not necessarily directly drive the structure of the implementing code. Their main focus within this document is to indicate the primary actions required to perform the base operations to ensure core connectivity.

3.1 Objects Referenced

Objects that are referred to in later sections and their associated figures are briefly described here. The structure of these objects is described in only enough detail to provide the conceptual flow of information types in the following activity diagrams.

Request	<p>The request object provides details to allow a service request to be processed. Attributes of this object will include</p> <ul style="list-style-type: none"> • process • requestId • receiver[] • sender • destination[] • origin <p>process identifies the processing activity that is required from the Connectivity Coordinator. This may or may not align with technical services as some processes may include a number of technical services in their processing.</p> <p>requestId is provided to allow simple correlation capabilities.</p> <p>The duplication of receiver, sender, destination and origin is to allow this information to be passed in with the request if known rather than extracted from the payload.</p>
Address	<p>An Address object provides convenient access to details of the parties involved in the interaction and identifies</p>

	<p>the interaction itself. The attributes for address will include:</p> <ul style="list-style-type: none">• documentType• receiver[]• sender• destination[]• origin
Cert	A Cert object refers to either a X.509 certificate or a reference to such a certificate.
CryptoDetails	A CryptoDetails object provides details of what parties (via certification or certificate references) were involved in signing and/or encrypting the received message.
EnvelopeDetails	An EnvelopeDetails object details the parties involved in securing the transport aspects of the interaction.
KeyRef	A KeyRef object identifies the key that is associated with a given Certificate.
Interaction Interactions	A Interaction object (Interactions is list of type Interaction) represents the data structure that is returned by the ELS that identifies the interaction the target can support. Refer to [ELSA2008] .
Payload	A Payload object represents the information that is being conveyed in the business interaction. Usually a document or report.
PrivateKey	A PrivateKey object represents the private key component of an asymmetric key pair.
SecurePayload	A SecurePayload represents a payload that has been encrypted as per [XSP2008] .
SignedPayload	A SignedPayload represents a payload that has been signed as per [XSP2008] .
ELS	A ELS object represents an endpoint reference to Web service providing ELS capabilities.
Status	A Status object identifies whether the operation succeeded or failed.
Target	A Target object provides the end point details for Web service identified as fulfilling a role within an interaction.

4 Environment Overview

4.1 Infrastructure Services

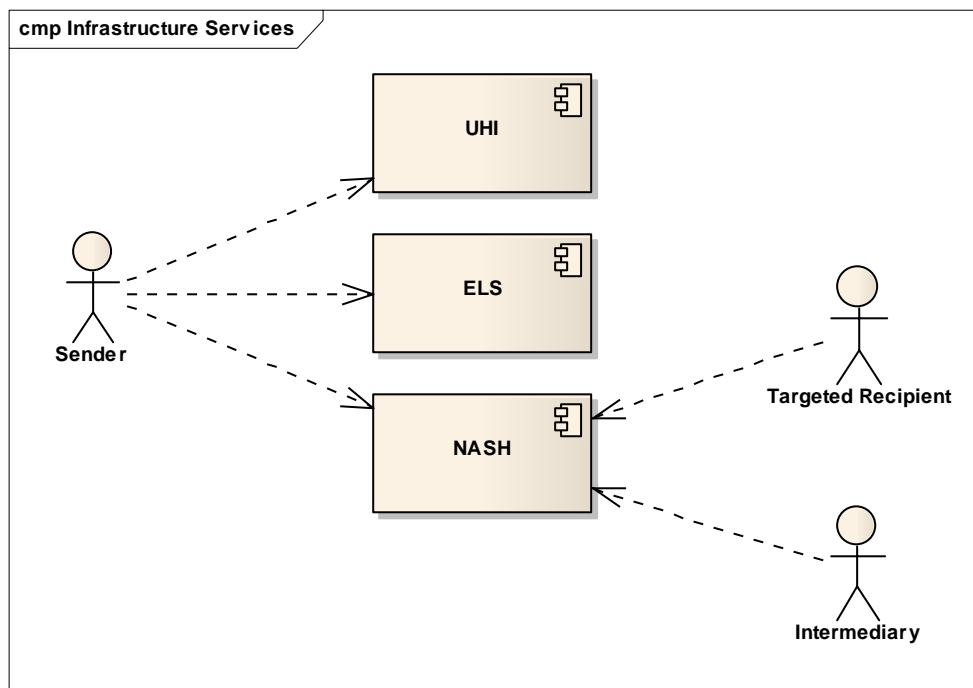


Figure 1: Infrastructure Services

In order for communication to occur within an environment that contains many parties that may have limited knowledge of one another a number of infrastructure related services are required to provide either authoritative information or bootstrap a process of information discovery to support interaction capabilities. NEHTA has identified three primary infrastructure services in this space, Unique Health Identifier (UHI), Nation Authentication Service for Health (NASH) and Endpoint Locator Service (ELS).

Details of how these components are involved in e-health interactions will be described in more detail in the Process View section. The Process View section hinges on many of the concepts identified in Connectivity Architecture [CONA2008].

4.1.1 UHI

The UHI is the authoritative source of e-health related identifiers. It stores information about identifiers and the entity that these identifiers are relate to and allows information to be retrieved about either aspect by trusted parties. It is also likely that the UHI will provide a discovery capability allowing the lookup of the ELS that is representative of the entity that will be the receiver in an e-health interaction.

4.1.2 NASH

The NASH provides authoritative cryptographic type information about entities involved in e-health interactions and is used to provide details to verify the identity of an entity involved in an interaction or to provide details required to obfuscate information that will be sent to such an entity. This functionality will be implemented by PKI technology and often delivered as X.509 certificates.

4.1.3 Endpoint Locator Service

This ELS provides an information discovery capability to find the appropriate endpoint and communication method to deliver information to a receiving party involved in an e-health interaction. For a more in-depth discussion on the concepts of ELS please refer to [ELSA2008].

4.2 Interaction Types

The interaction type used in an e-health interaction is driven by the capabilities of the parties involved in the interaction. Often this is driven primarily by the capabilities of the receiver, however there can be other influencing factors. These capabilities, or lack thereof, can also affect the roles required to achieve the desired result of the interaction. For example, if the target recipient is not capable of receiving a message via Web services directly a third party could be included to receive the message on behalf of the recipient. The recipient can then retrieve the message at some later time from the third party.

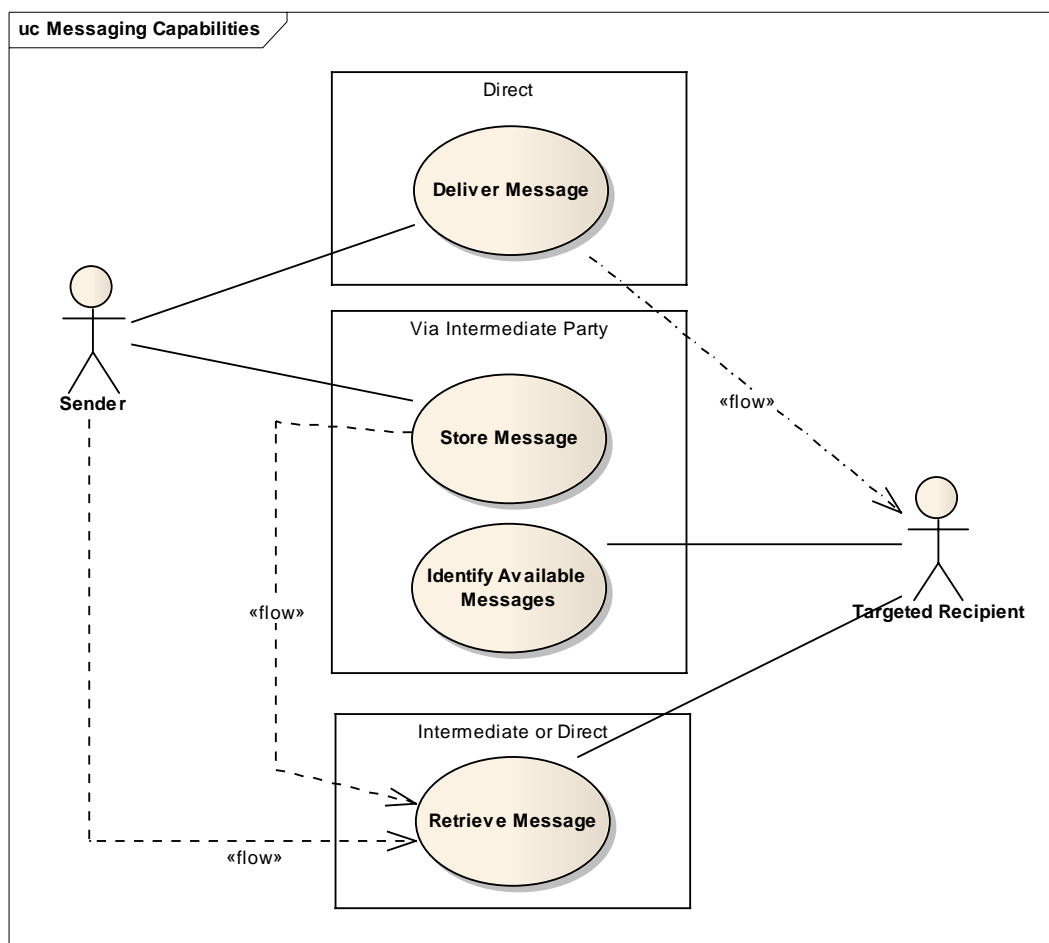


Figure 2: Messaging Capabilities

In order to support interactions between parties with these varying constraints on their messaging capabilities a number of messaging interactions have been identified to deal with the various communication scenarios. The diagram above provides a summary of the use cases that these interactions must support. For a more in-depth discussion on these communications patterns please refer to [CPIS2008].

5 Process View

This section describes processes that will provide the capabilities necessary to participate in e-health related interactions. The processes described here are only one solution to the problem domain and it should not be inferred that these are the preferred solution.

It is suggested that prior to reading this section you make yourself familiar with [CONA2008], [ELSA2008] and [CPIS2008].

5.1 Primary Use Cases

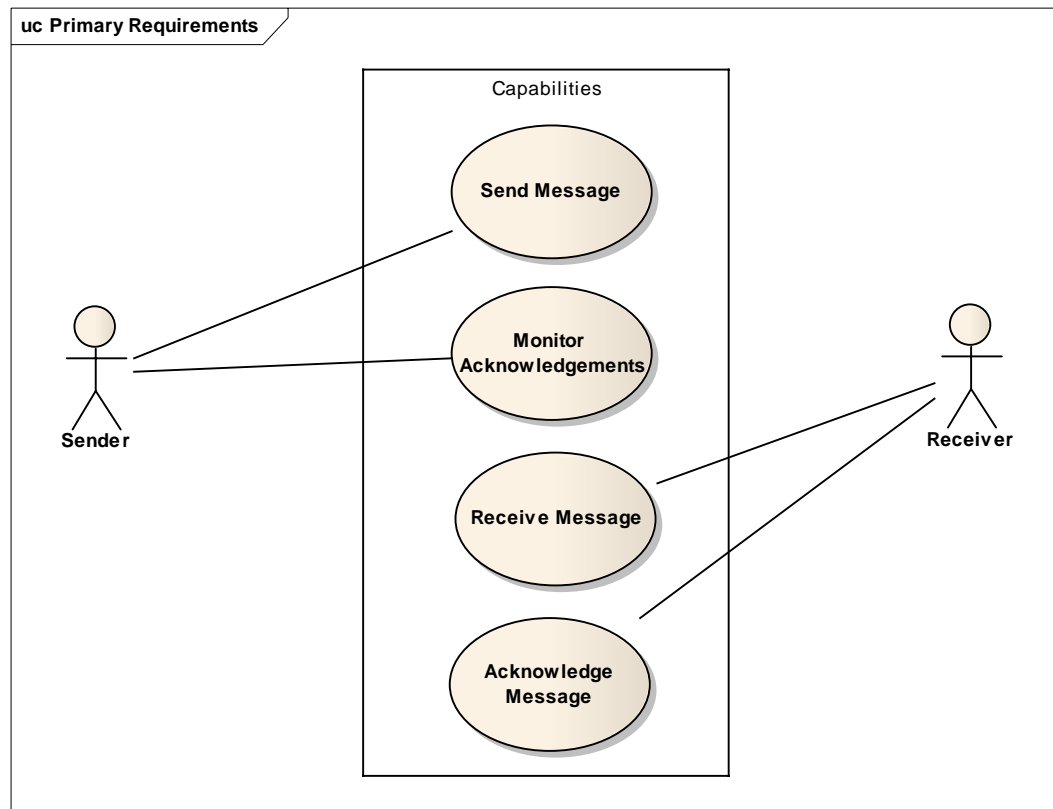


Figure 3: Primary Use Cases

The above diagram gives a high level view of how the roles of sender and receiver need to interact with the system.

The Sender needs to be able to:

- Send messages; and
- Monitor acknowledgement messages received in response to their sent messages

The Receiver needs to be able to:

- Receive messages; and
- Send acknowledgements (positive or negative) to the messages received

5.2 High level overview

5.2.1 Outgoing Service Requests

The following diagram gives a high level pictorial representation of the processes that take place within a system when it issues an outgoing service request to a domain service.

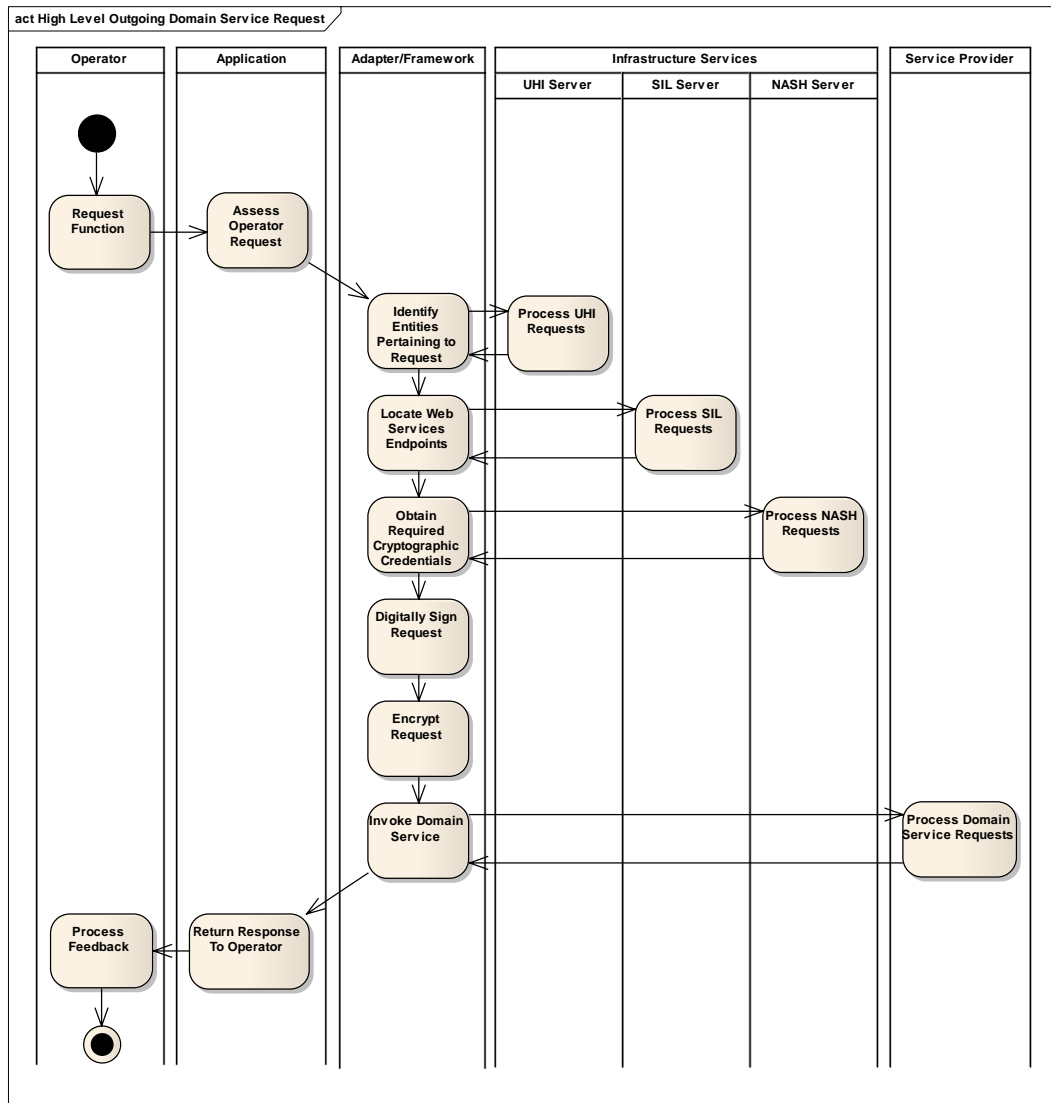


Figure 4: High Level Outgoing Domain Service Request

1. The system operator performs an operation on the Application.
2. An assessment is made by the Application on the operation which determines that an outgoing request needs to be made to an external service via the application's associated connectivity adapter.
3. The Adapter interacts with the HPI-O Server to obtain the endpoint of the ELS associated with the HPI-O identified as the target of the request.
4. The Adapter interacts with ELS to locate the domain service endpoints for communication to the identified organisation.
5. The Adapter interacts with the NASH to obtain/validate the currency of the PKI certificate for the identified organisation.
6. The Adapter digitally signs the request.
7. If required by the interaction the Adapter encrypts the request

8. The Adapter returns a response consistent with the operation performed to the Application which is presented to the Operator in the appropriate manner.

5.2.2 Incoming Service Requests

The following gives a high level pictorial representation of the processes that take place within a system when it acts as Domain Server and receives an incoming service request.

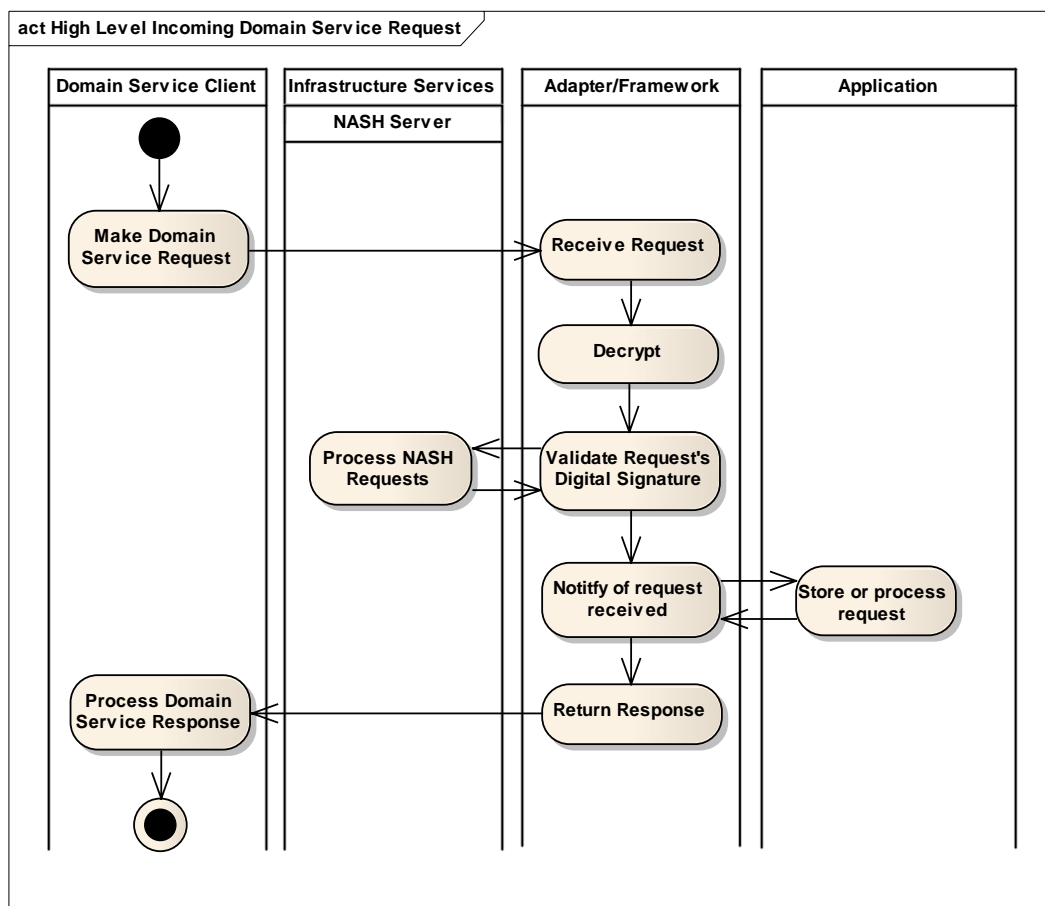


Figure 5: High Level Incoming Domain Service Request

1. The Domain Service Client makes a service request on the Domain Service Provider (Adapter).
2. The Adapter decrypt the incoming request
3. Validates the currency of the PKI certificate for the Domain Service Client and ensures the signature associated with the message is valid and is the signature of a trusted party (this may involve interaction with NASH).
4. Dispatches the request to the appropriate processing code.
5. Returns a response indicative of the success of the operation to the Domain Service Client.

5.3 Service Consumer Processing

The processing described here assumes that the application has already obtained and validated Identifiers as they were selected by the operator and prior to the dispatch of the request to the Connectivity Coordinator component.

The following diagram shows the core processing activities involved in requesting a service operation.

The first activity shown in the diagram is used to extract the addressing type details for the input message. These details are required to dispatch the request in later activities.

The second activity is required to lookup from the ELS associated with the target organisation and discover which interactions are supported.

The third step processes the Interactions identified in the previous activity to find an appropriate method to communicate with the target organisation with the final step allowing an appropriate response to be constructed to return to the service request initiator.

The first and last steps are essentially place holders to allow transformations that may or may not be required for a specific implementation. It may serve well to leave hooks for such capabilities however.

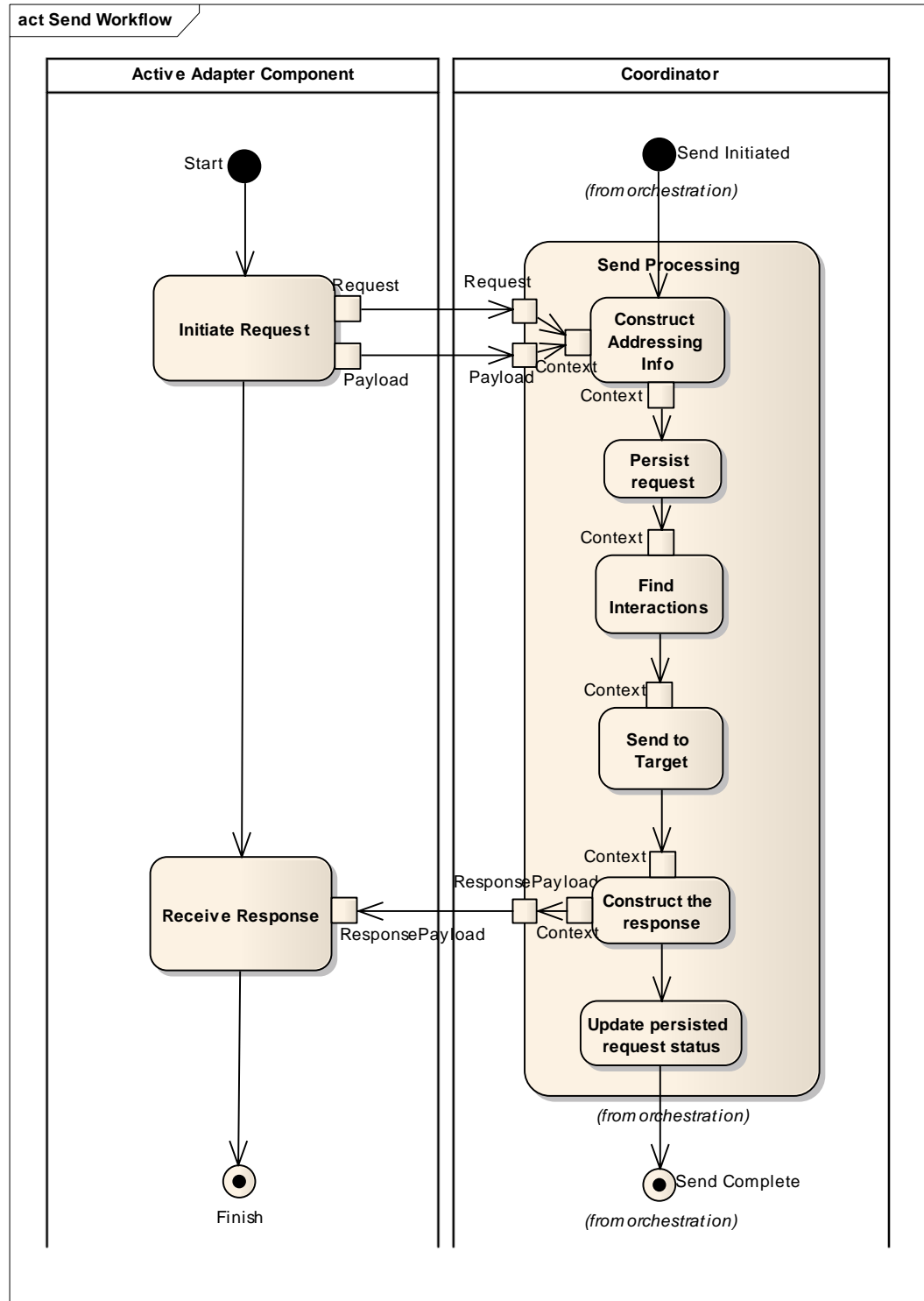


Figure 6: Service Consumer High Level Processing

The following sections provide a more detailed breakdown of the activities shown in the above diagram.

5.3.1 Gather Address Information

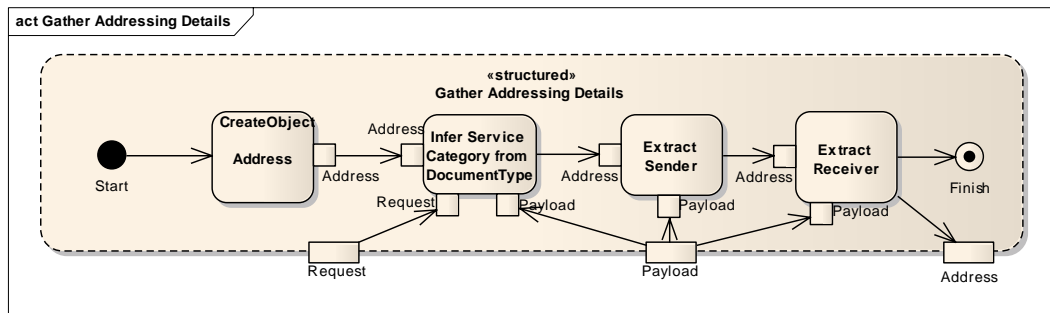


Figure 7: Gather Addressing Details

Inputs: Request, Payload

Outputs: Address

This activity assumes that interaction addressing type information will always be present somewhere within the business orientated payload. Such as in a deliver request. For retrieval type requests it is expected that the details for the Address object will come primarily from the Request object.

The purpose of the Gather Addressing Details activity is to construct and populate an Address object by extracting addressing related and document type details from the incoming payload. It is essentially a transformation capability. This activity **SHOULD BE** implemented as a generic component where possible but there may be instances where it may need to be a specific implementation aligned to package or similar. The generic capability could be achieved by using XPath for XML type payloads or offsets or field matching methods for binary or structured text to extract the relevant details.

5.3.1.1 Identifier Mapping

If identifier mapping is required (i.e. local to global or jurisdiction) this could be achieved by appending an additional action at the end of the flow in this activity to provide the mapping capabilities. An alternate solution, and one that provides a separation of concerns (extract details then perform translation), is to place the mapping activity directly off of the main work flow between the 'Gather Addressing Details' and the 'Retrieve Interactions' activities.

5.3.2 Retrieve Required Interactions

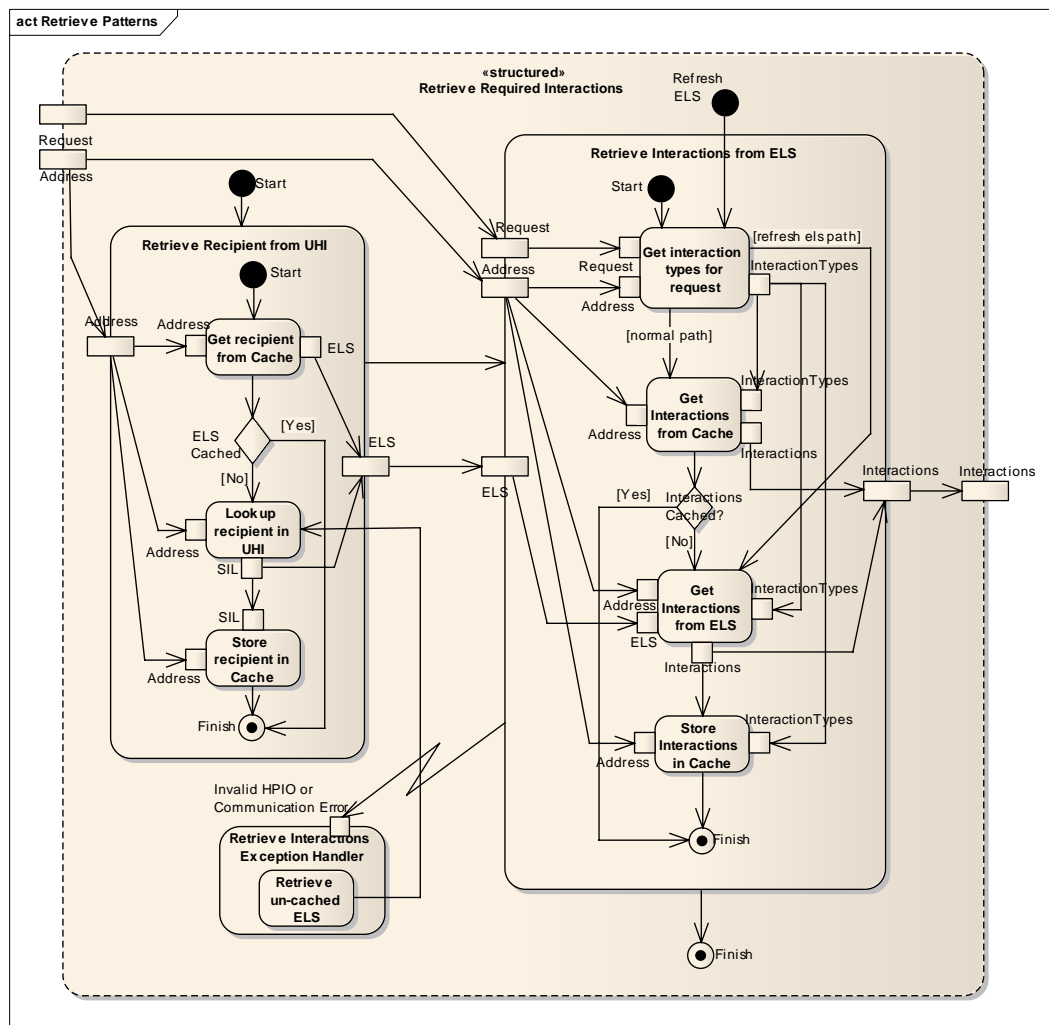


Figure 8: Retrieve Required Interactions

Inputs: Request, Address

Outputs: Interactions

Retrieving the service interactions that the receiver supports involves two key steps. The first step is to retrieve the recipient’s associated ELS endpoint reference. The second is to use the reference gained in the first step to contact the recipient’s associated ELS to retrieve the interactions that the recipient, or their delegated service provider, supports. These sub activities are described in more detail in the following sections.

5.3.2.1 Retrieve Recipient from UHI

Inputs: Address

Outputs: ELS

As it is expected that the mapping between a HPI-O and their nominated ELS will remain predominately static these mappings **SHOULD BE** cached to reduce the overheads of retrieving these details over the network for subsequent requests to the same recipient. This cache would be referred to in the first instance and if an entry cannot be found the details would be requested from the UHI with the results being stored in the cache to minimize the overhead for subsequent references to the same HPI-O.

Where the request must be issued to the UHI consider the following state diagram for the types of issues that may occur while interacting with the UHI.

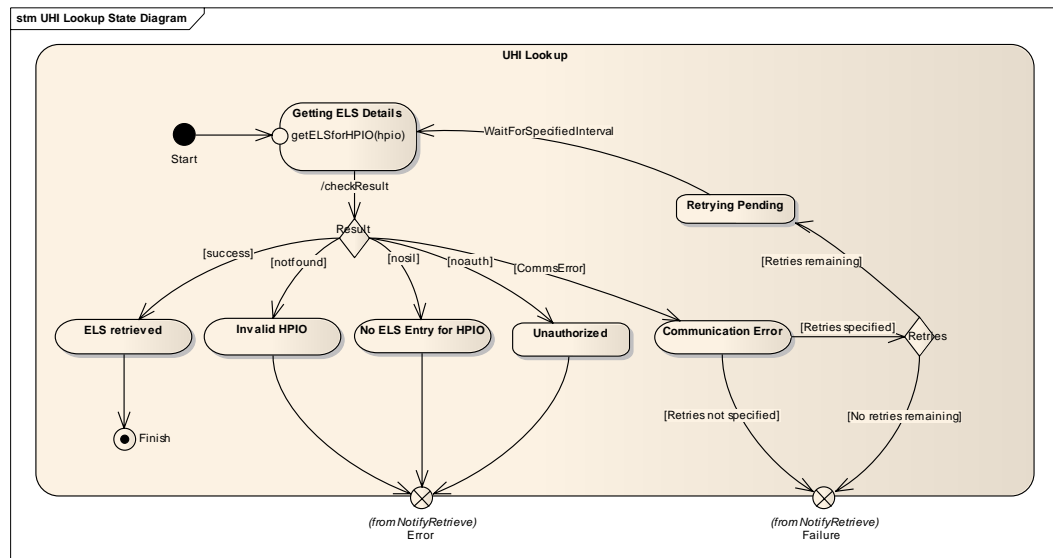


Figure 9: UHI Lookup States

5.3.2.2 Retrieve Interactions from ELS

Inputs: Request, Address, ELS

Outputs: Interactions

As it is expected that the mapping between a HPI-O's Business Service and the supported interactions for this Business Service will remain predominately static these mappings **SHOULD BE** cached to reduce the overheads of retrieving these details over the network for subsequent requests directed at the same Business Service for the same recipient. This cache would be referred to in the first instance and if an entry cannot be found the details would be requested from the associated ELS with the results being stored in the cache to minimize the overhead for subsequent references to the same HPI-O and Business Service.

The first action in this activity is to determine the InteractionTypes that this request requires to achieve its service request. This would more than likely be a combination of Service Category and Service Interface that has been associated with the process being executed within the Connectivity Coordinator and the type of document being operated on. This type of association **SHOULD BE** provided via configuration items.

The ELS specification states that an 'Invalid HPIO' or Communication type error should cause the reference held for the ELS to be refreshed. This is shown in Figure 8 by the exception handler towards the bottom of the diagram. The exception handler passes control back to the remote UHI lookup action, bypassing the cache lookup, to allow the refresh of the ELS endpoint and then continuing to refresh the Interaction entries. There is also a second entry point 'Refresh ELS' on the 'Retrieve Interaction from ELS Endpoint' to force the cache to be refreshed if an error has occurred during the service request that may be caused by a stale Interaction entry.

Where the request must be issued to the recipients nominated ELS consider the following state diagram for the types of issues that may occur while interacting with the ELS.

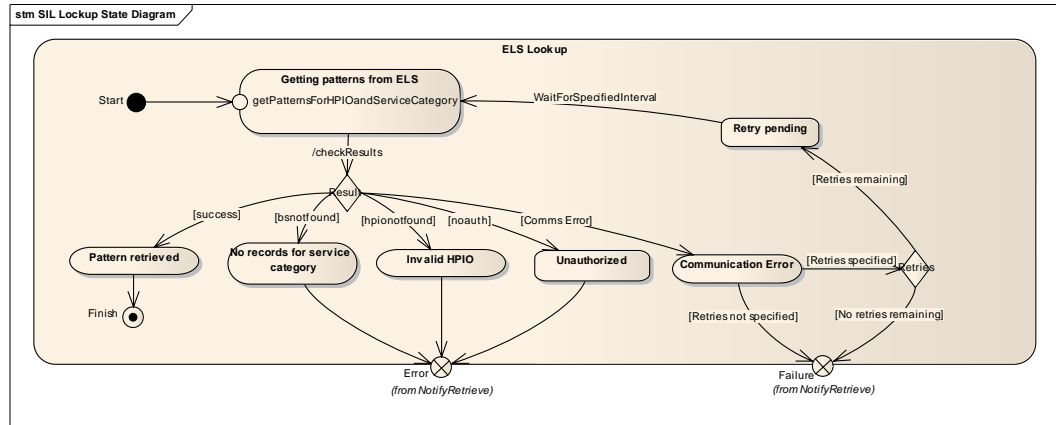


Figure 10: ELS Lookup States

5.3.3 Process Interactions

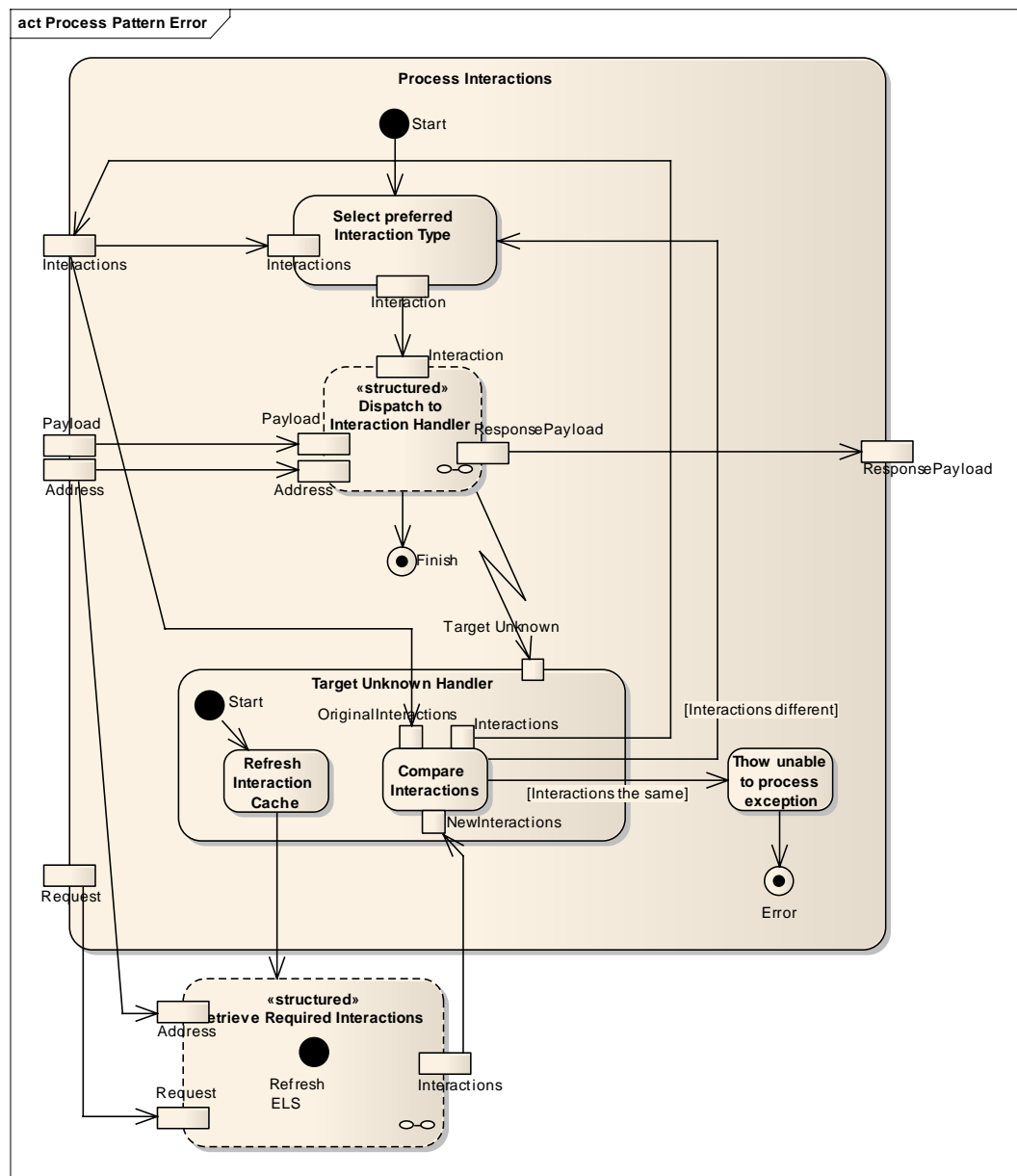


Figure 11: Process Interactions

Inputs: Interactions, Request, Address, Payload

Outputs: ResponsePayload

The primary goal of the Process Interaction activity is to provide a common dispatch and error handling capability.

The first action in this activity is to select the preferred Interaction for the request. This will often be an empty operation as the process of selecting the Interactions in Retrieve Required Interactions will have fine enough granularity to return only a single result. Interactions as defined by the ELS Architecture [ELSA2008] have no preference associated attribute and therefore this action will give the local instance an opportunity to apply its own business rules for endpoint selection if required. This may be as simple as matching domain names in the endpoints of the Interaction.

If a 'target unknown' type exception is encountered during the processing an exception handler must capture this exception and attempt to refresh the ELS Interactions cache. The handler must check the results of the refresh to ensure that the interactions have changed else the details are still incorrect and the problem **MUST BE** escalated to be rectified.

5.3.3.1 Dispatch to Interaction Handler

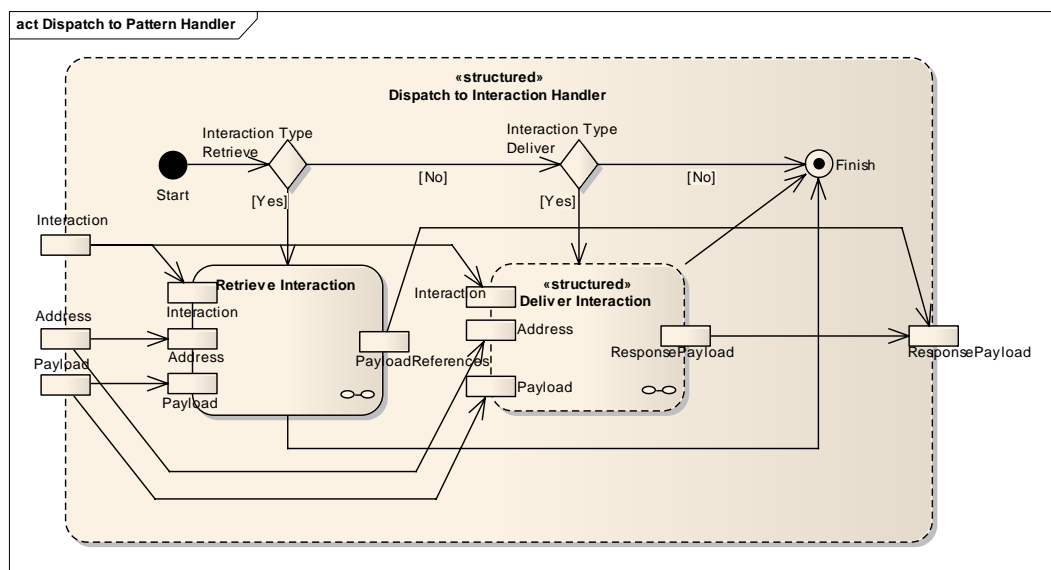


Figure 12: Dispatch to Interaction Handler

Inputs: Interaction, Address, Payload

Outputs: ResponsePayload

The Dispatch to Interaction Handler activity consists primarily of a case statement to identify the appropriate Interaction Handler to execute for the interaction that has been given as an input parameter. Both Dispatch to Interaction Handler and the Interaction Handlers themselves comply with the same interface. This allows the addition of new handlers without affecting the structure or behaviour of the encapsulating process, Process Interaction.

5.3.3.1.1 Retrieve Interaction

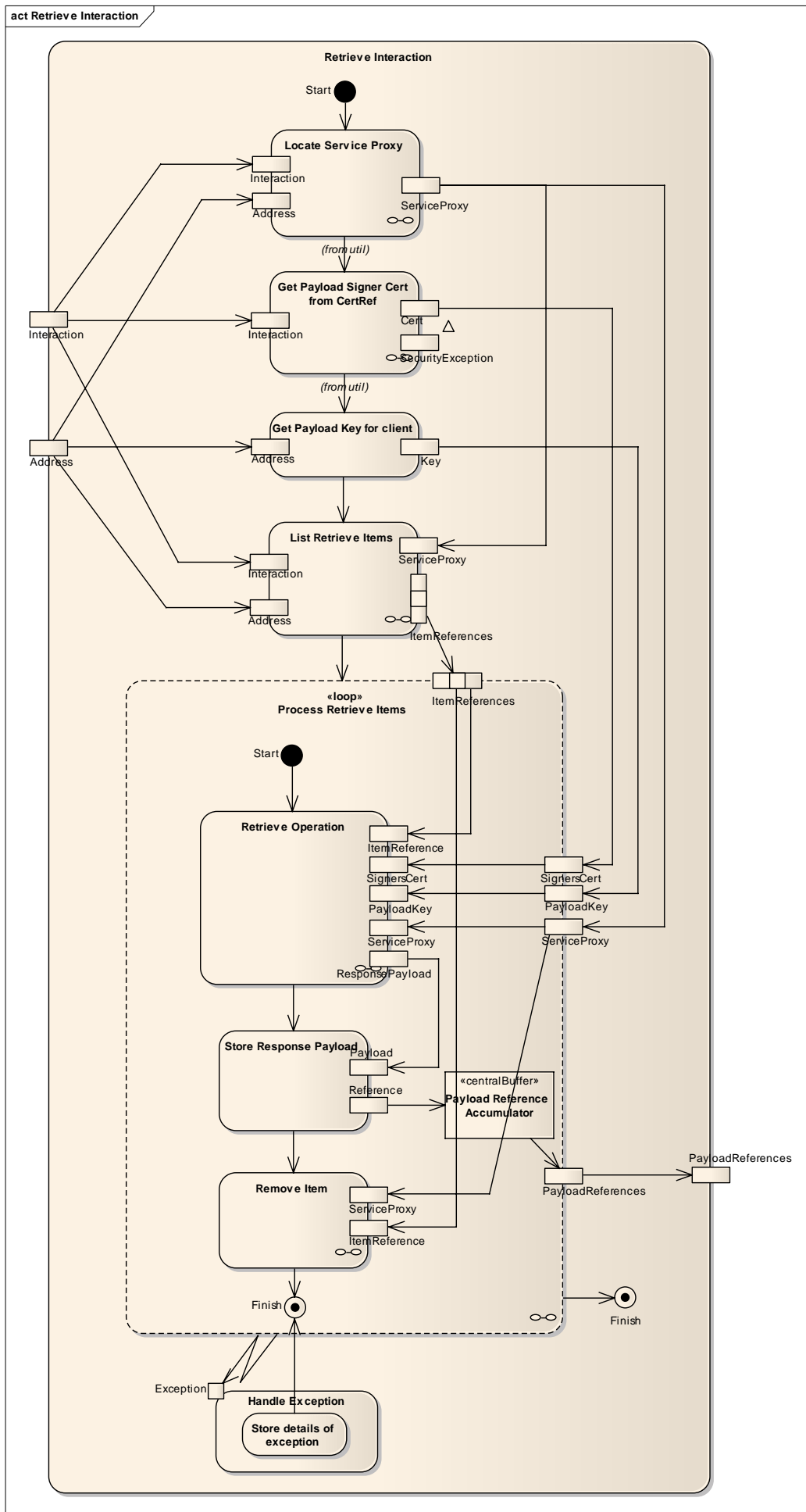


Figure 13: Retrieve Interaction

Inputs: Interaction, Address

Outputs: PayloadReferences

The Retrieve Interaction activity implements the steps required to retrieve messages from a designated intermediary that have been issued for the recipient party in question. The activity assumes that all messages available at an intermediary will be retrieve in a continuous set of automated operations rather than in an interactive manner of review and select items.

The purpose of the initial step in this activity is to locate or create an initialised service proxy that will be used to call the domain based Webservice operations on the endpoint specified in the Interaction object (see section Locate Service Proxy for further details).

After the service proxy has been located steps need to be performed to locate the details of the certificates and keys that will be used for the Secure Payload. The first step is to locate the certificate associated with key that will be used to sign the payload. The second step involves identifying the key that the receiver expects the payload to be encrypted for. This is derived from the Address destination field.

The remaining steps are responsible for the processing loop. First listing the available messages, retrieving each message, storing the message and notifying the intermediary that it can now be removed. These methods are invoked on the service proxy that was allocated in the first step of the greater activity. During this processing loop an array keeps track of the payload references to be returned to the caller. As an example these references could be file locations in a file system or the primary keys of the entries in a database.

Failures during processing will result in the details of the exception being stored. The process assumes that the operations are idempotent (i.e. later retrieving the record causing the error and storing again will not cause issues) and therefore the processing of the record originally in error will be completed in a future scan. This is guaranteed by the remove being the last operation.

5.3.3.1.2 Deliver Interaction

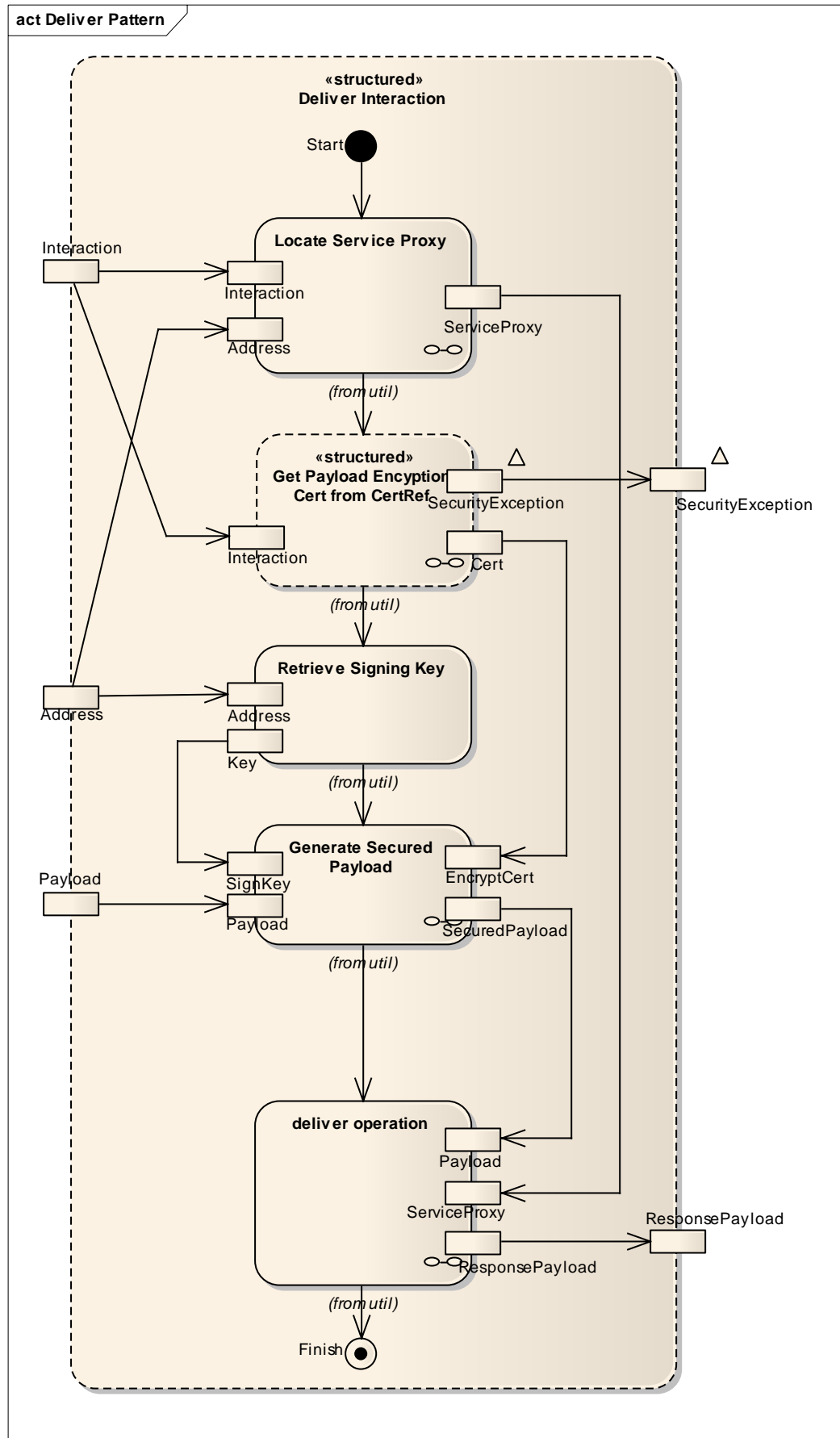


Figure 14: Deliver Interaction

Inputs: Interaction, Address, Payload

Outputs: ResponsePayload

The Deliver Interaction activity implements the steps required to deliver a payload targeted to an end recipient via a direct service interaction with the recipient or via a service interaction with the recipients delegated intermediary.

The purpose of the initial step in this activity is to locate or create an initialised service proxy that will be used to call the domain based Webservice operations on the endpoint specified in the Interaction object (see section Locate Service Proxy for further details).

After the service proxy has been located steps need to be performed to locate the details of the certificates and keys that will be used to construct the Secure Payload for the deliver operation. The first step involves locating the certificate identified in the ELS Interaction CertRef that will be used for encrypting data for the recipient party. Details of this activity are shown in Figure 15. The second step involves retrieving the key that will be used to sign the payload prior to encrypting. This is located through an association with the Address object's sender field.

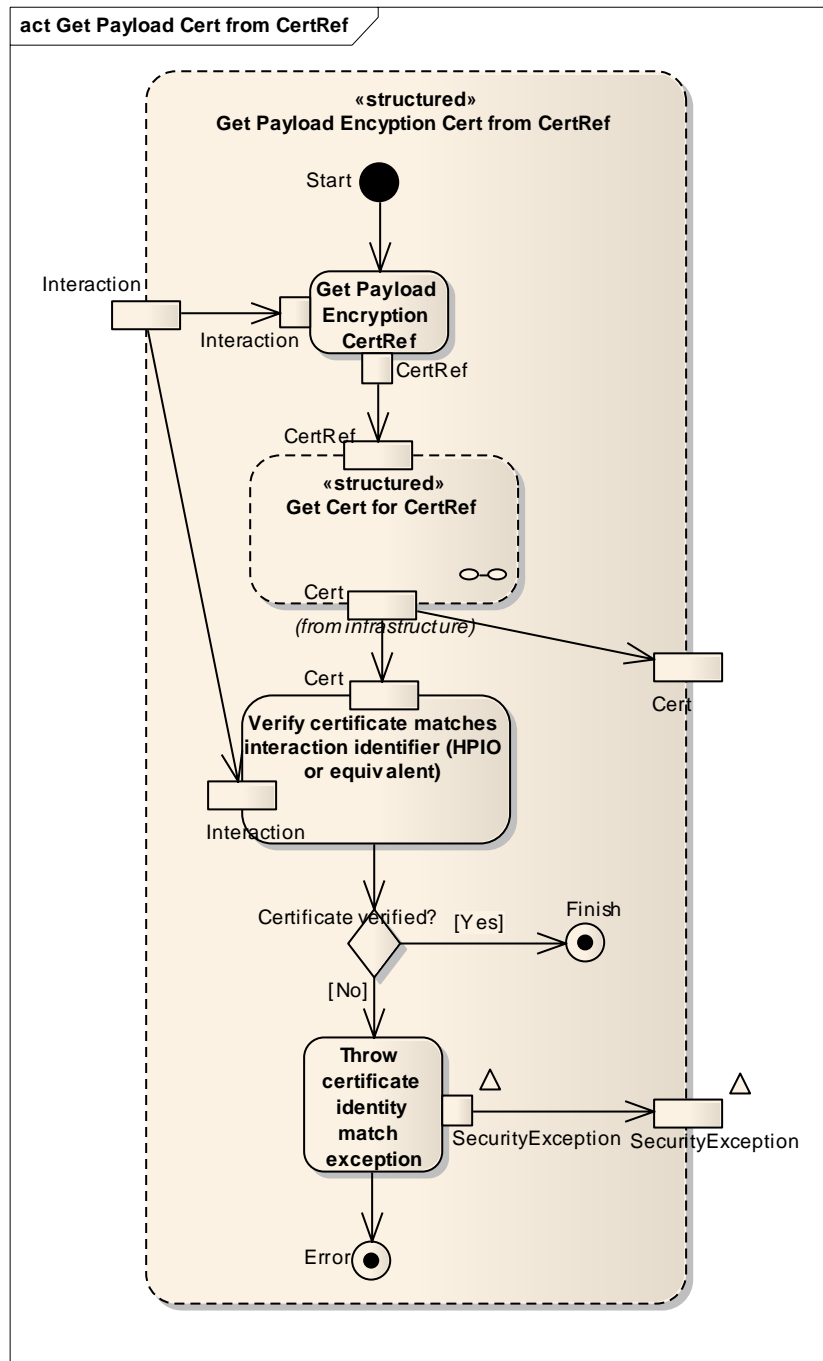


Figure 15 - Get Payload Encryption Cert from CertRef

Once the key and certificate required to construct the secure payload have been retrieved the secure payload is generated. This secure payload is then passed as a parameter to the method associated with deliver operation on the service proxy that was allocated in the first step.

5.3.4 Common Send Interaction Processes

There are number of processes that are common across all interactions used to send messages. These are described in the following sections.

5.3.4.1 Locate Service Proxy

The Locate Service Proxy activity first uses the address object to identify the sending party and retrieve their key for the mutually authenticated TLS session that will be established by the service proxy. This step may not be

possible in some environments. The next step is to identify the expected certificate that the server in the TLS session will be using. As a basic verification the common name contained in the certificate should be compared to domain name of the endpoint to ensure the TLS certificate is for the targeted endpoint. Once the keys and certificates have been retrieved the class or component implementing the service proxy must be located, possibly from a pool or instantiating a class instance, and initialised for operation. See Figure 16 below for an overview of this activity.

The service proxy component **SHOULD** implement an interface that is aligned with the interaction pattern rather than the domain specific interface as described in the WSDL. This will allow a standard process to call specific Webservices for different domain packages via in interface shared by the specific service proxies.

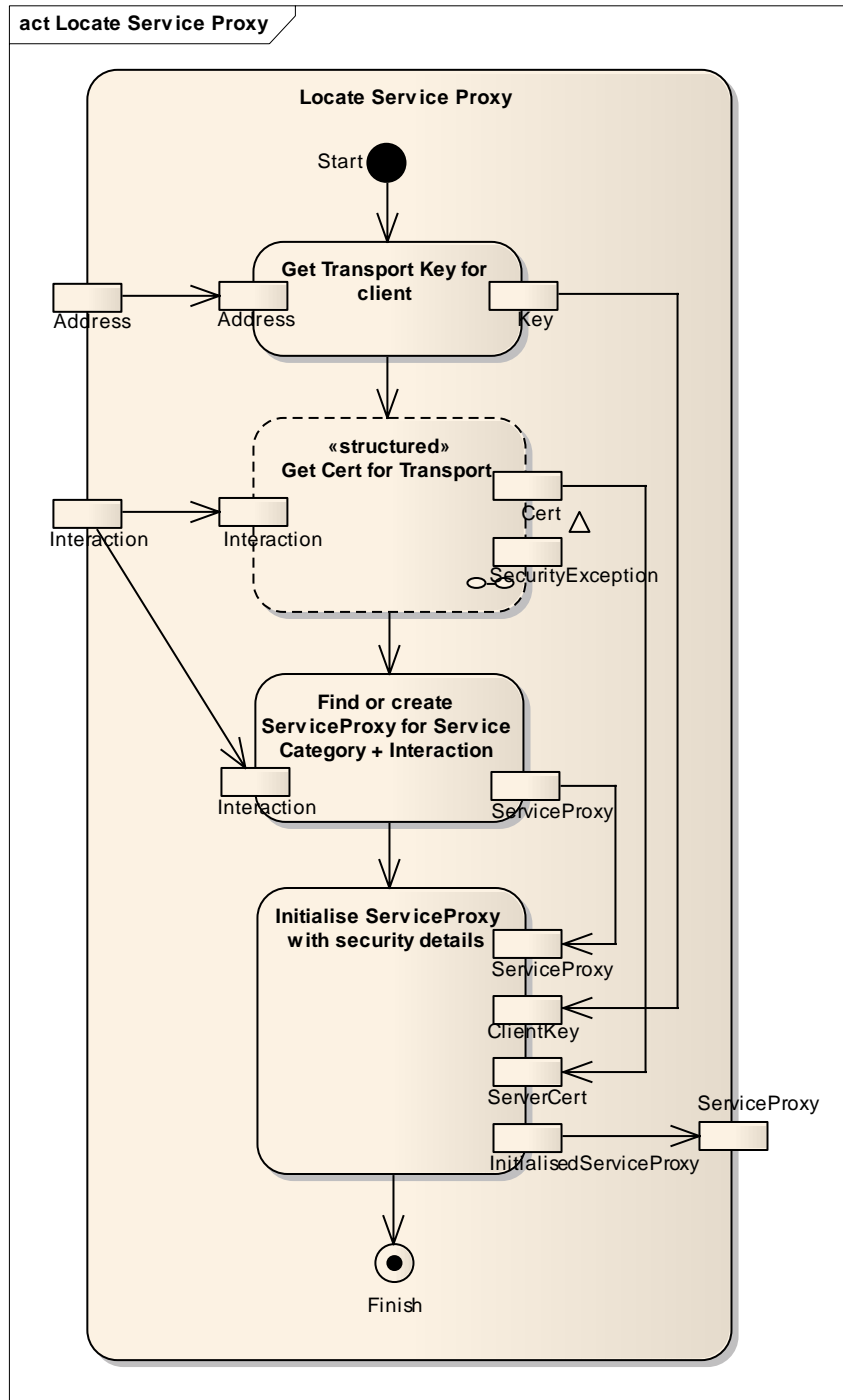


Figure 16: Locate Service Proxy

5.3.4.2 Get Certificate for Transport

The diagram below shows the key steps required to retrieve and verify the certificate to be used by the server in the TLS session that will be established for a service call.

The first step locates the Certificate Reference within the ELS Interaction that relates to the transport certificate that will be used by the remote endpoint. This reference is then passed to the general 'Get Cert for CertRef' activity (see Figure 18) to retrieve the certificate from either the local cache or remotely from the NASH certificate provider. Once the certificate has been retrieve, at a minimum, the certificate should be verified by comparing the domain name of the host specified in the certificate with the endpoint identified in the ELS Interaction record. Assuming this verification passes the certificate can be passed back to the call for use. If verification should fail an error path should be taken or exception thrown to ensure the caller is aware the verification has failed and the certificate must not be used.

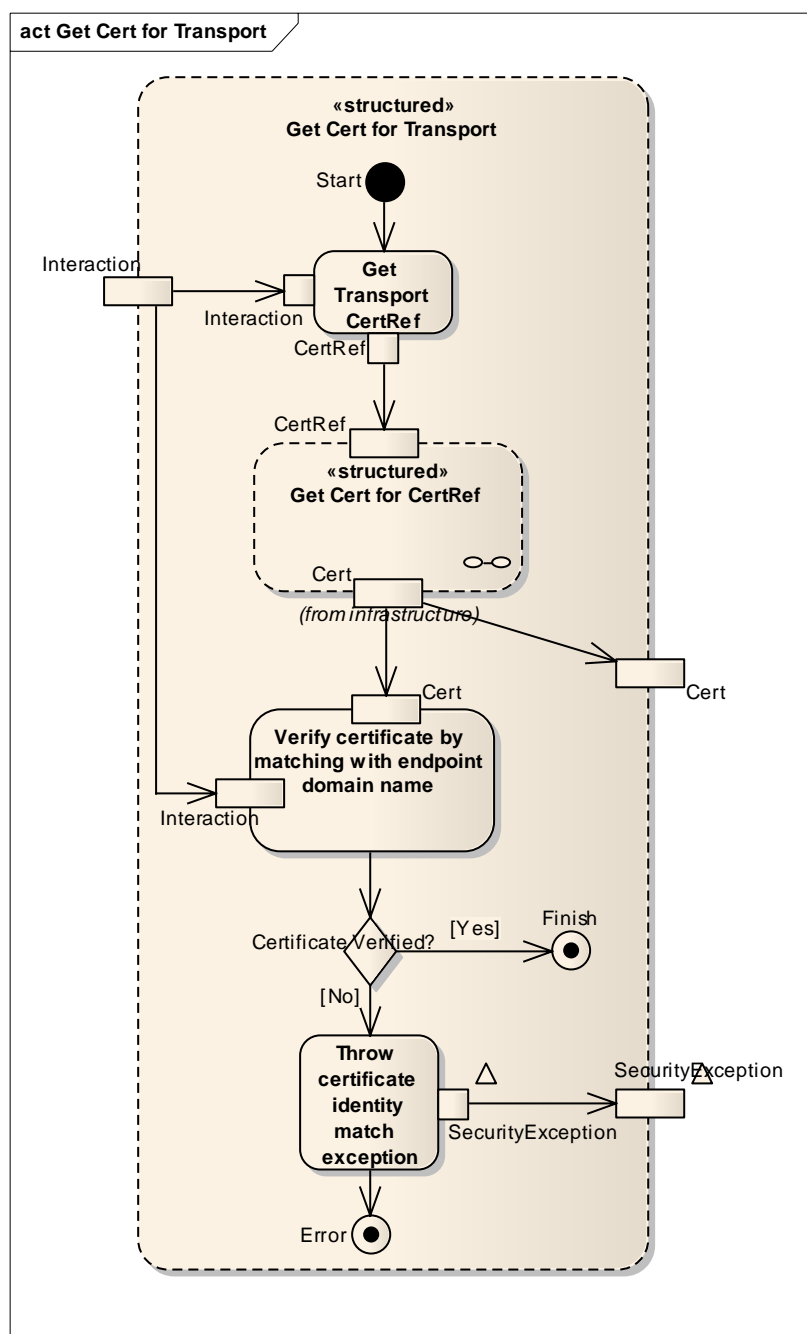


Figure 17: Certificate Retrieval

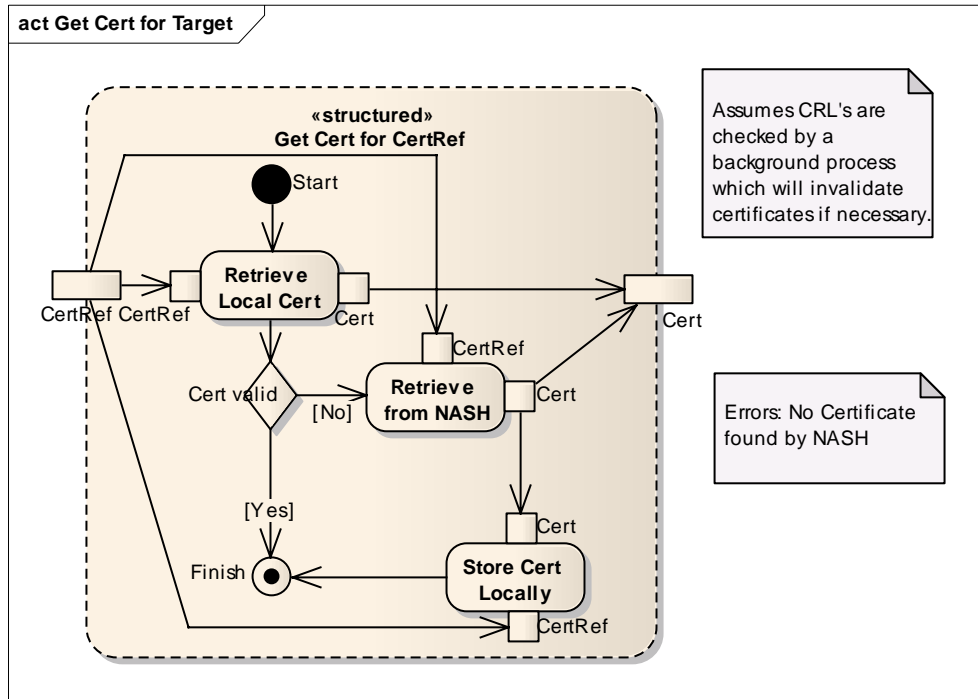


Figure 18: Certificate Caching

5.4 Service Provider Processing

The follow diagram shows the high level processing activities. The first activity is used to decrypt and verify the secure payload. This activity returns the original payload and details of the party that signed and encrypted the secure payload. The second activity passes the payload along with the details of the party who signed and encrypted the secure payload and the party that provided the transport layer security to the code nominated by the recipient for application specific processing. The transport layer security details must be provided by the proxy component receiving the request when initiating the Workflow Coordinator. The status returned from the application specific processing is then used to send either a successful or unsuccessful response.

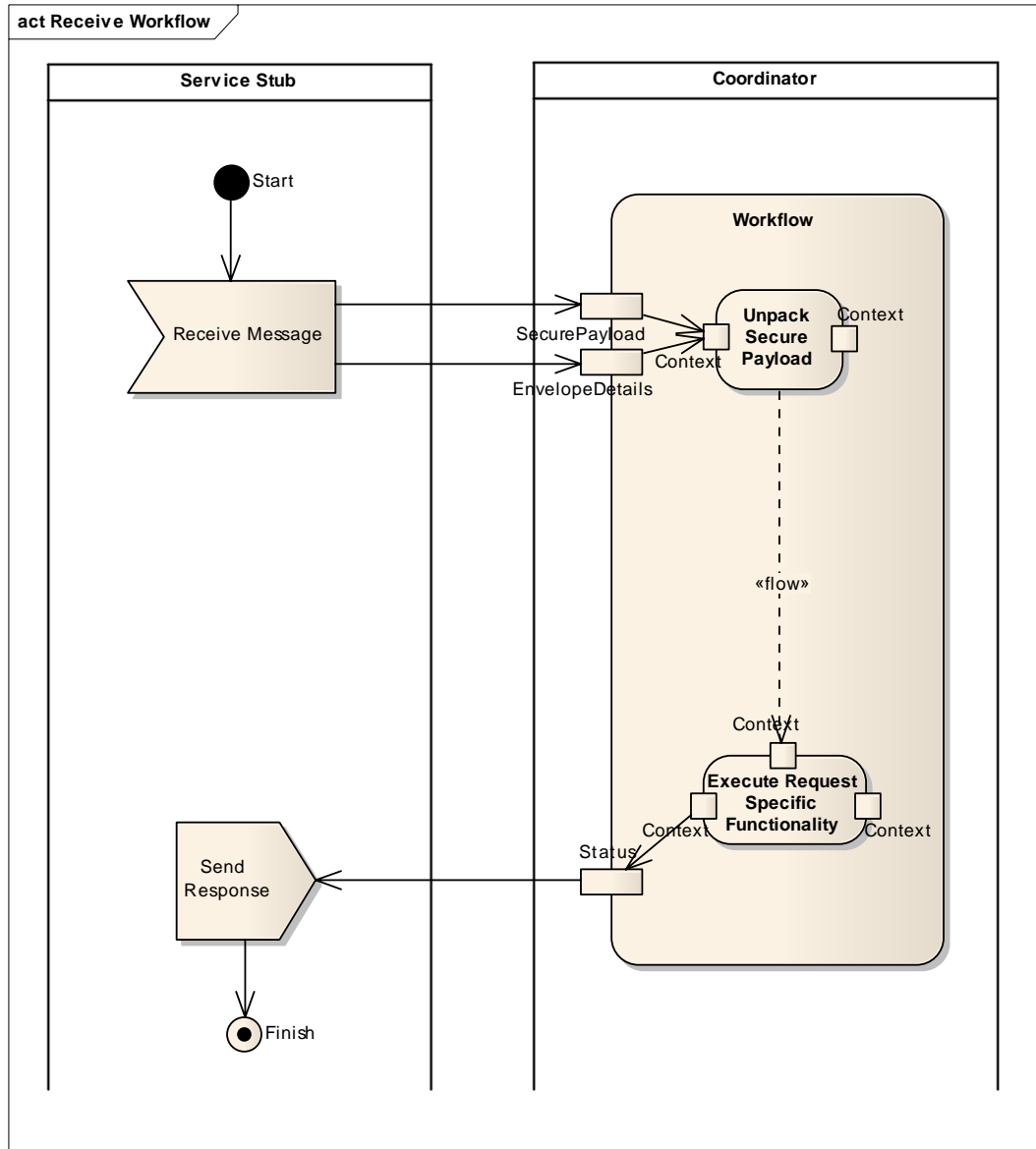


Figure 19: Receive Workflow

5.4.1 Process Secure Payload

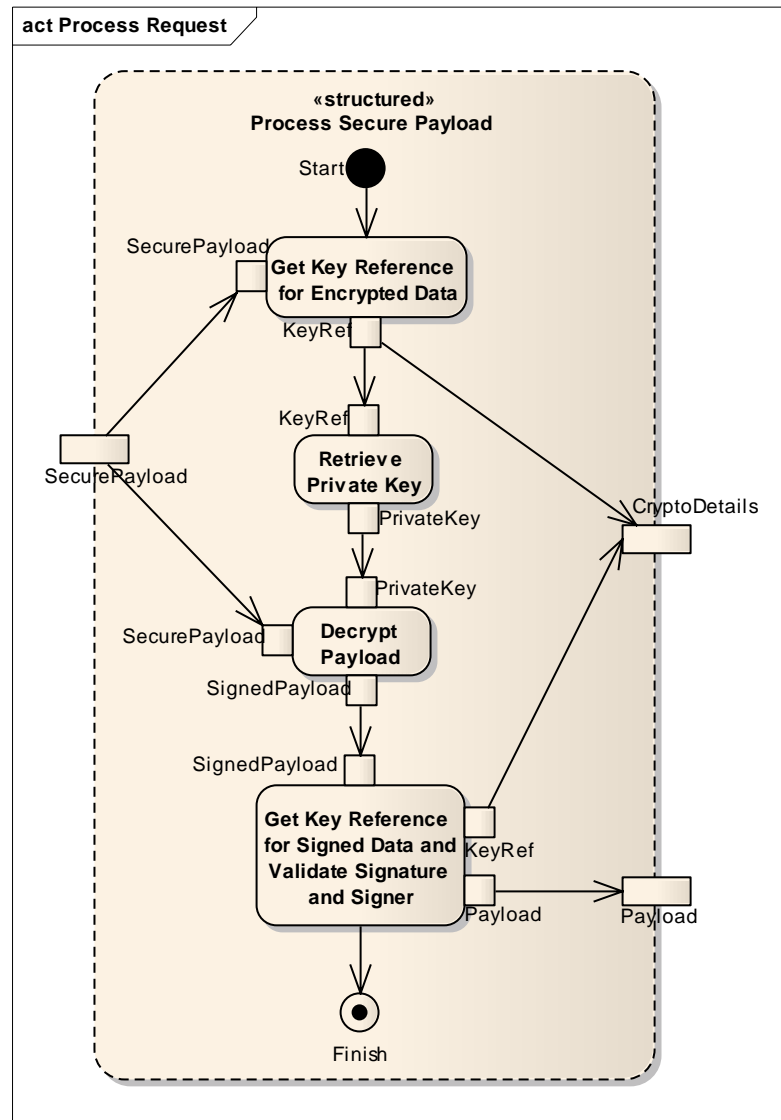


Figure 20: Process Secure Payload

Inputs: SecurePayload

Outputs: CryptoDetails, Payload

The Process Secure Payload activity is primarily focussed on deciphering the secured payload and returning the original payload and the details of who encrypted and signed the secure payload. This is achieved by first extracting the details of the certificate that was used to encrypt the payload from the appropriate location in the secure payload structure. This certificate is used to find the associated private key that must be stored locally and once available the key is used to decrypt the payload. Assuming the payload has also been signed the certificate found in the signers keyinfo area of the secure payload is used to verify that the signature is valid. This certificate must be validated to ensure it is signed by a trusted source. Once these steps have been completed successfully the decrypted payload and details of which certificate was used for encryption and the certificate associated with the signing key are returned to the caller. For further details on the processing of an XML Secure Payload please refer to [XSP2008].

5.4.2 Invoke Service Provider Code

Inputs: EnvelopeDetails, CryptoDetails, Payload

Outputs: Status

The Invoke Recipient Code activity provides the deciphered payload and details of how the message was secured prior to processing beginning. These details include information on who signed and encrypted the SOAP message components (EnvelopeDetails) and who signed and encrypted the secure payload (CryptoDetails). These details allow the recipients code base to determine the validity or trustworthiness of the message being processed. The result from the executed recipient code must be a response that represents the status of the execution (i.e. success, failure). This will be packaged up in the response back to the issuing client. It is expected that this step would either persistently store the received request for later processing or call an appropriate API for immediate processing.

5.5 Acknowledge Processing

There are no distinguishing concepts in regards to acknowledgement processing. Acknowledge processing is performed using the same process as Service Consumer processing.

5.6 Acknowledgement Processing

There are no distinguishing concepts in regards to acknowledgement processing. Acknowledgement processing is performed using the same process as Service Provider processing.

6 Engineering View

Pre-requisite reading for this section include

- Web Service Profile [WSP2009]
- Connectivity Architecture document [CONA2008]

6.1 Design considerations

The key design considerations from the engineering perspective include:

- Designed with extensibility in mind to allow addition of future endpoint specifications
- Designed to allow progressive implementation of National Infrastructure services
- Ability to cache expensive remotely located resources (UHI, ELS, Certificate entities)
- Built on foundation libraries to provide base components to build alternative implementations
- Common connectivity related tasks abstracted/simplified
- Deployable for service consumer and provider environment (container based)
- Deployable for service consumer only type environment
- Provide persistence capabilities for messages involved in interactions

These items form a part of the overall requirements described in the Qualities and Requirements section.

6.2 Core components

This section describes the key components that are leveraged to achieve the design considerations identified in the previous section. A summary of these components and their relationships is shown in **Figure 21** below followed by sub sections describing the components in more detail.

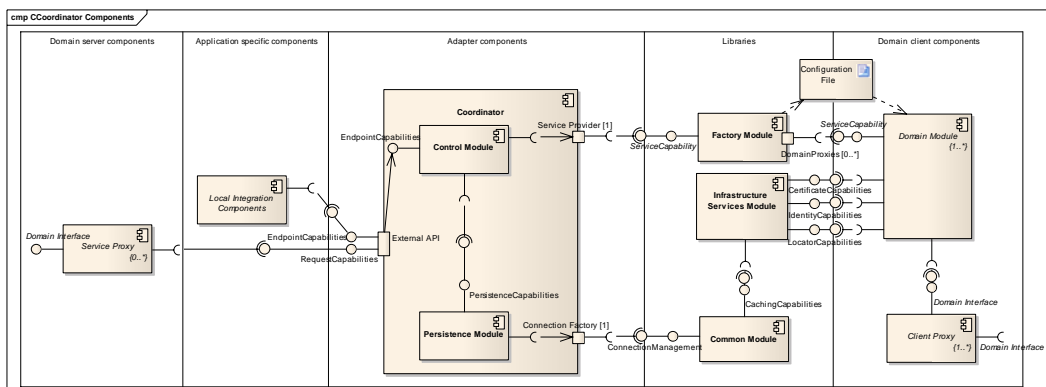


Figure 21: Core Components

6.2.1 Library Components

The library components provide the foundation on which other higher level components can implement endpoint capabilities (i.e. send/resend message etc.). The library components provide interfaces to allow the consumption of Services. These services being both common and domain specific services. The common services are generally required across all interactions. The

domain services are specialised components that are discovered via the common components of the libraries and represent the technical services underlying the business services described by an e-health related domain.

6.2.1.1 Factory Module

The Factory Module Component is the contact point to locate a domain related service (referred to generally in **Figure 21** as ServiceCapability) that is required by the higher level components. The Factory Module Component finds the component exposing this service by referring to a configuration item associated with the service interface identifier (or similar). This association identifies the component that provides the service capability.

6.2.1.2 Infrastructure Services Module

The Infrastructure Services Module provides access to the services that enable core connectivity. The three main services in this category include Certificate Capabilities, Identity Capabilities and Locator Capabilities. Certificate Capabilities provides access to the certificate management capabilities such as certificate retrieval and verification. Identity Capabilities provides access to details related to identities (i.e. UHI) and is primarily used to find the ELS endpoint associated with a specific targeted recipient. Locator Capabilities provides capabilities related to locating a domain related service endpoint (i.e. ELS). It provides lookup and verification capabilities for interaction records. The Infrastructure Services Module also consumes the Caching Capabilities service provided by the Common Module. This is used to cache the expensive network located resources of the core connectivity related services locally.

6.2.1.3 Common Module

The Common Module provides capabilities that are used by many other components. The two shown on the diagram are Connection Management and Caching Capabilities but it is expected this module would provide other base capabilities such as logging and XML Secure Payload processing [XSP2008]. The Connection Management interface allows components to locate a database connection for a specific purpose relieving them of the need to have additional configuration items. The Caching Capabilities provides an interface that exposes common caching operations. This component being in the Common Module relieves other components from needing to implement the configuration items and cache implementations. This also allows cached resources to be managed consistently via funnelling caching operations through this common component.

6.2.2 Domain Client Components

The Domain Module Component represents the component that provides the ServiceCapability. It is an abstract component that is replaced by a domain specific component in an implementation that fulfils its role. This component implements a common interface that represents the interaction pattern for which the domain service complies. The Domain Module Component provides a mapping from the common interface representation to the specific domain service. This concept allows the reuse of higher level components across different domains using the same interaction patterns. The Domain Module Component consumes common services, such as IdentificationCapabilities (UHI), LocatorCapabilities (ELS) and CertificateCapabilities (NASH/Certificates) provided by Infrastructure Service Module Component of the common libraries to achieve all aspects of connectivity for the domain service. The Domain Module Component **SHOULD** optimise the holding and releasing of all resources it accumulates during processing to ensure efficient communication.

6.2.3 Adapter Components

The Adapter Components are the key components that collaborate to provide the higher level functionality that is exposed by the Coordinator. These key components include the Coordinator Module and the Persistence Module. The Coordinator Module provides the functionality required by external components to perform e-health service interactions. The Coordinator Module leverages the capabilities exposed by the library components and the Persistence Module components to achieve this functionality. The Persistence Module provides persistence capabilities to the Coordinator Module to allow for such functionality as interaction tracking, retry capabilities and message correlation to name a few.

Appendix A: Informative references

- [WSP2009] NEHTA, Web Services Profile, Version 3.1, 24 April 2009.
- [XSP2008] NEHTA, XML Secure Payload Profile, Version 1.0, 1 December 2008.
- [CPIS2008] NEHTA, Concepts and Patterns for Implementing Services, Version 2.0, 1 December 2008
- [UHICO2007] NEHTA, Unique Healthcare Identification – Concept of Operations, Version 2.0, 3 September 2007.
- [CONA2008] NEHTA, Connectivity – Architecture, Version 1.0, 1 December 2008
- [ELSA2008] NEHTA, Service Instance Locator – Architecture, Version 1.1, 1 December 2008.