

PASR**Preparatory Action on the enhancement of the European industrial potential in the field of Security Research**

Project acronym: SUPHICE

Project full title: Secure Unplanned Provisioning of High Integrity Communications across Europe

Proposal/Contract no.: SEC4-PR-017100-SUPHICE

D3.2 Interface Definition

Project Document Number: D32-O-PU-003-TRT

Project Document Date: 12.06.2006

Workpackage Contributing to the Project Document: WP3

Deliverable Type and Security: O¹-PU²

Author(s): Sarah Butler, Mark Irons, Martin Jigstam

Abstract: This document is the second deliverable of WP3. It is the definition of the interfaces for each element in the unplanned security architecture.

Keywords: WP3, security, communications, Interfaces, Policy, Rules

¹Type: P - Prototype, R - Report, D - Demonstrator, O - Other

² Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

History

Version	Date	Description, Author(s), Reviser(s)
0.1	22/02/2005	Document creation, Sarah Butler (TRT)
1.0	31/03/2005	Comments from review added and issued, Sarah Butler (TRT)
1.1	12/12/2005	Updates for second deliverable, Mark Irons / Sarah Butler (TRT)
2.0 Draft	14/12/2005	Draft version of second deliverable for review by team
2.0	21/12/2005	Version 2 issued
3.0 Draft	08/06/2006	Draft Public version (TRT)
3.0	12/06/2006	Version 3 issued

Executive Summary

The SUPHICE system will support rapid deployment and configuration of an IP network in response to unplanned requirements for secure communications. The SUPHICE system provides secure unplanned communication between a number of LANs connected across the Internet or closed WAN via IP crypto devices.

The SUPHICE system is concerned with the network level of a system and not the application level. So an IP cryptographic device will provide access to the equipment on the network it protects.

This document defines the interfaces for each element of the SUPHICE unplanned security architecture. All the relevant interfaces are Web Services interfaces.

Contents

	Page
1 Introduction	6
1.1 Purpose and Scope	6
1.2 Architecture	7
1.3 Structure	7
1.4 Definitions	7
1.5 Abbreviations	8
2 Interface Definition	9
3 Interface Design Considerations	11
3.1 Message Format	11
3.2 Interface Development Approach.....	11
3.3 Security.....	11
3.4 WSDL and XML Schema Versioning	12
4 Crypto Services Registry	13
4.1 Use Cases	13
4.2 Publishing Services	13
4.2.1 Publishing tModels Describing SUPHICE Web Service Types.....	14
4.2.2 Publishing Organisations Offering SUPHICE Web Services	15
4.2.3 Publishing Implementations of SUPHICE Web Services	15
4.2.4 Definition of Key Names for Categorising SUPHICE Web Services.....	16
4.3 Discovering Services.....	17
4.3.1 Discovering Implementations of SUPHICE Web Services.....	17
5 AuthService	19
5.1 SubmitAuthRequestOperation	19
5.1.1 Description.....	19
5.1.2 Request Parameters	19
5.1.3 Sample Request.....	19
5.1.4 Response Parameters.....	21
5.1.5 Sample Response	22
5.1.6 Errors.....	22
6 CryptoDeviceService	23
6.1 DeliverAuthDecisionOperation	23
6.1.1 Description.....	23
6.1.2 Request Parameters	23
6.1.3 Sample Request.....	23
6.1.4 Response Parameters.....	24
6.1.5 Errors.....	24
6.2 CreateSecurityAssociationOperation	24
6.2.1 Description.....	24
6.2.2 Request Parameters	24
6.2.3 Sample Request.....	24
6.2.4 Response Parameters.....	25
6.2.5 Errors.....	25
6.3 DeleteSecurityAssociationOperation	25
6.3.1 Description.....	25
6.3.2 Request Parameters	25
6.3.3 Sample Request.....	25
6.3.4 Response Parameters.....	26
6.3.5 Errors.....	26
6.4 GetSuphiceServiceDetailsOperation	26
6.4.1 Description.....	26
6.4.2 Request Parameters	26
6.4.3 Response Parameters.....	26
6.4.4 Errors.....	27

7	<i>CryptoParamsService</i>	28
7.1	<i>GetAlgorithmOperation</i>	28
7.1.1	Description.....	28
7.1.2	Request Parameters	28
7.1.3	Sample Request.....	28
7.1.4	Response Parameters.....	28
7.1.5	Sample Response	29
7.1.6	Errors.....	29
7.2	<i>GetCAOperation</i>	29
7.2.1	Description.....	29
7.2.2	Request Parameters	29
7.2.3	Sample Request.....	29
7.2.4	Response Parameters.....	30
7.2.5	Sample Response	30
7.2.6	Errors.....	30
7.3	<i>GetSoftwareModuleOperation</i>	30
7.3.1	Description.....	30
7.3.2	Request Parameters	30
7.3.3	Sample Request.....	31
7.3.4	Response Parameters.....	31
7.3.5	Sample Response	31
7.3.6	Errors.....	31
8	<i>Error Codes</i>	32
9	<i>References</i>	33

1 Introduction

1.1 Purpose and Scope

The purpose of this document is to define the interfaces for each element of the SUPHICE unplanned security architecture (defined in [2]). All the relevant interfaces are Web Service interfaces and have been defined using WSDL (Web Services Description Language) [6] and XSD (XML Schema Definition) [3], [4], [5]. This document does not contain the WSDL files but a description of the interfaces. Where appropriate, this document uses UML [1] diagrams to aid in the description of the architecture.

This document forms one part of SUPHICE deliverable D3.2n. A complete list of items comprising this deliverable is provided in Table 1 below:

File Name	Description
D32-0-CO-002-TRT Interface Definition.doc	This document
AuthService.wsdl	WSDL file defining the AuthService
CryptoDeviceService.wsdl	WSDL file defining the CryptoDeviceService
CryptoParamsService.wsdl	WSDL file defining the CryptoParamsService
Suphice.xsd	XSD file defining XML types used by WSDL files
SuphiceXSD.html (and associated *.png diagrams)	HTML documentation of the XML types defined in Suphice.xsd
Xmlmime.xsd	XML Schema defined in the Assigning Media Types to Binary Data in XML Specification http://www.w3.org/TR/xml-media-types [8]

Table 1 – Items Comprising Deliverable D3.2n

1.2 Architecture

The SUPHICE system provides secure unplanned communication between a number of LANs connected across the Internet or closed WAN via IP crypto devices. In order to describe the architecture in more detail the simple case will be shown where only two LANs wish to connect. Of course, the same process could be used to add more LANs to the network. Figure 1 shows an architecture for such a system, where the two LANs happen to be from different countries (they could also be from different agencies within the same country). The main elements are as follows:

- IP crypto devices
- a cryptographic services registry
- an authorisation server and policy repository with the use of policy based management
- a distribution server for cryptographic parameters

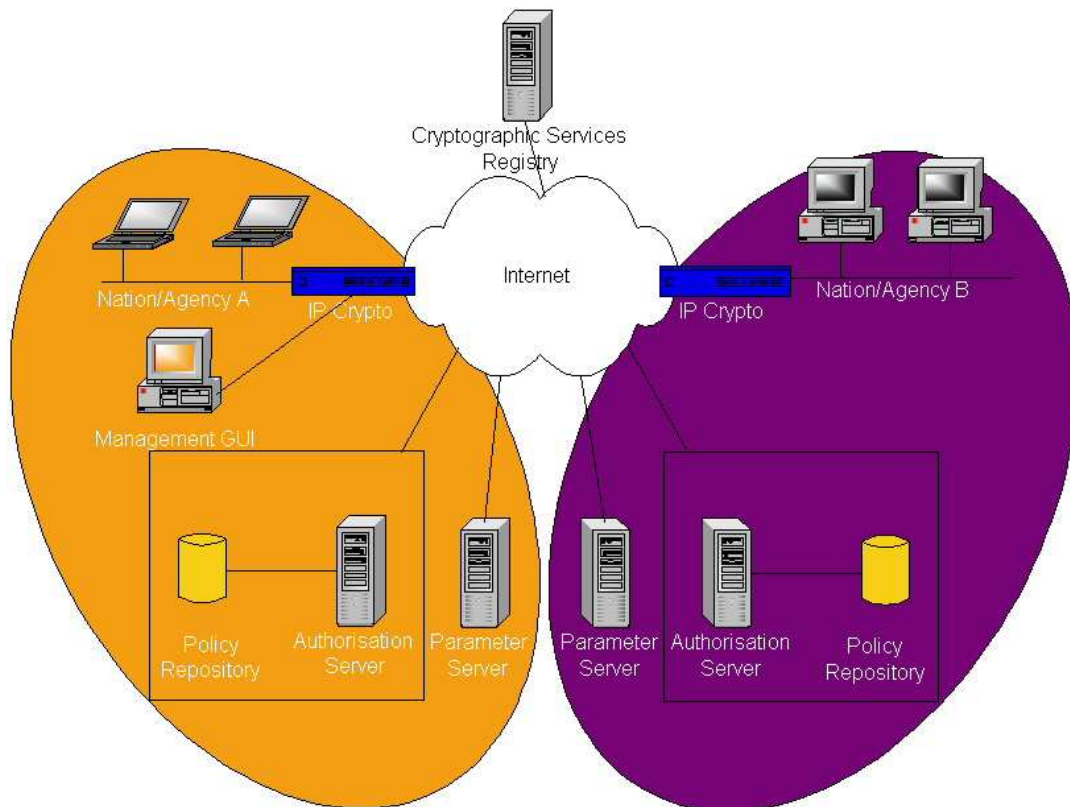


Figure 1 – Secure Unplanned Network Architecture

1.3 Structure

Section 2 provides an overview of the interfaces defined by the SUPHICE architecture. In Section 3 an introduction to Web Services technology is provided, along with details of design decisions taken. The interfaces for the SUPHICE secure ad-hoc service solution are described in sections 4 through to 7. A summary of SUPHICE interface error codes is provided in Section 8.

1.4 Definitions

Element: The SUPHICE architecture is composed of four main types of element. The Cryptographic Services Registry, the Authorisation Server, the Parameters Server, and the IP cryptos.

IP cryptos: An IP crypto is a cryptographic device that can encrypt and decrypt Internet Protocol packets, and also interact with the other elements of SUPHICE.

Key material or key: This is the information needed by the IP cryptos to create a secure tunnel. It may be certificates or pre-loaded cryptographic keys.

Service: IP cryptos are said to provide a *service*. This service will give secure access to a LAN located behind the IP crypto at a defined security level.

1.5 Abbreviations

API	Application Programming Interface
CA	Certificate Authority
EJB	Enterprise Java Bean
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JAX-RPC	Java API for XML-based Remote Procedure Call
J2EE	Java 2 Enterprise Edition
JSR	Java Specification Request
LAN	Local Area Network
OASIS	Organization for the Advancement of Structured Information Standards
SA	Security Association
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TRT(UK)	Thales Research & Technology (UK) Limited
UDDI	Universal Description Discovery and Integration
UML	Unified Modelling Language
WAN	Wide Area Network
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WS-I	Web Services Interoperability organisation
XML	eXtensible Markup Language
XSD	Xml Schema Definition

2 Interface Definition

Table 2 below lists all the interfaces of a SUPHICE system and defines the communications technologies they use. This document will focus on the specification of the web services interfaces.

Client	Server	Interface Type
Crypto Device	Cryptographic Service Registry	Web Service
Crypto Device	Authorisation Server	Web Service
Authorisation Server	Crypto Device	Web Service
Crypto Device	Parameter Server	Web Service
User	Authorisation Server	Web Site
User	Parameter Server	Web Site
Crypto	Crypto	Proprietary (not part of SUPHICE)
Management App	Crypto Device	Proprietary (not part of SUPHICE)

Table 2 – Interface Type

A UDDI registry [10] will provide the functionality of the Cryptographic Services Registry. The publish and inquiry interfaces exposed by a UDDI registry are standard and are defined in [10]. Section 4 of this document defines how the UDDI interfaces are used within the context of the SUPHICE architecture.

The diagram below shows the interfaces defined by SUPHICE (i.e. it excludes the Cryptographic Services Registry interfaces). The operations exposed by each interface are shown. These operations are discussed in more detail later in this document.

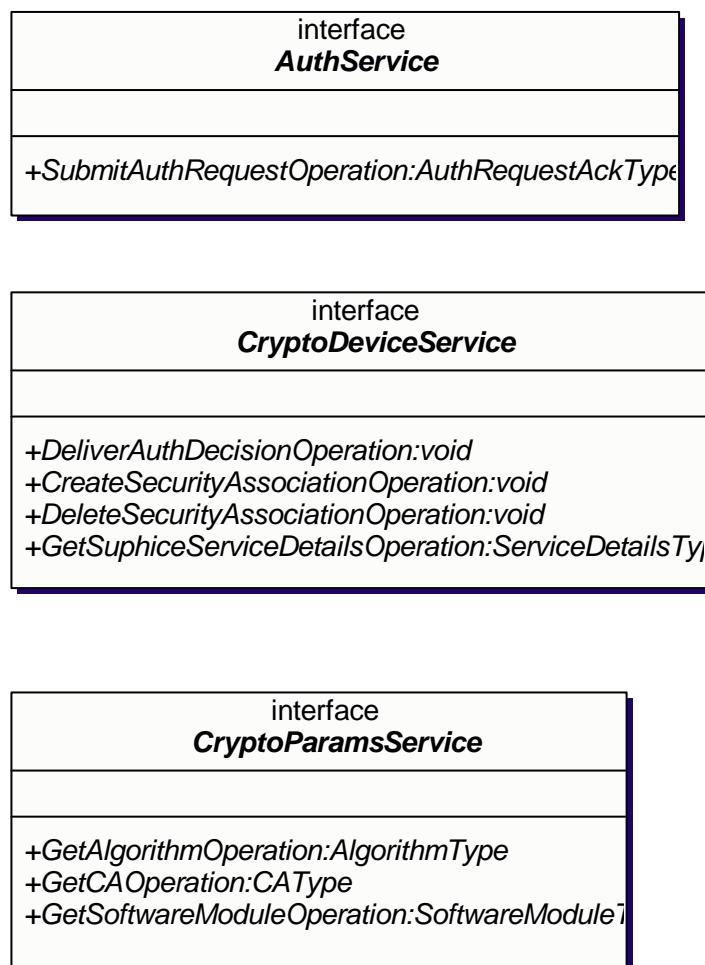


Figure 2 – SUPHICE Interfaces Class Diagram

The sequence diagram below shows the order that the operations from the above class diagram are called.

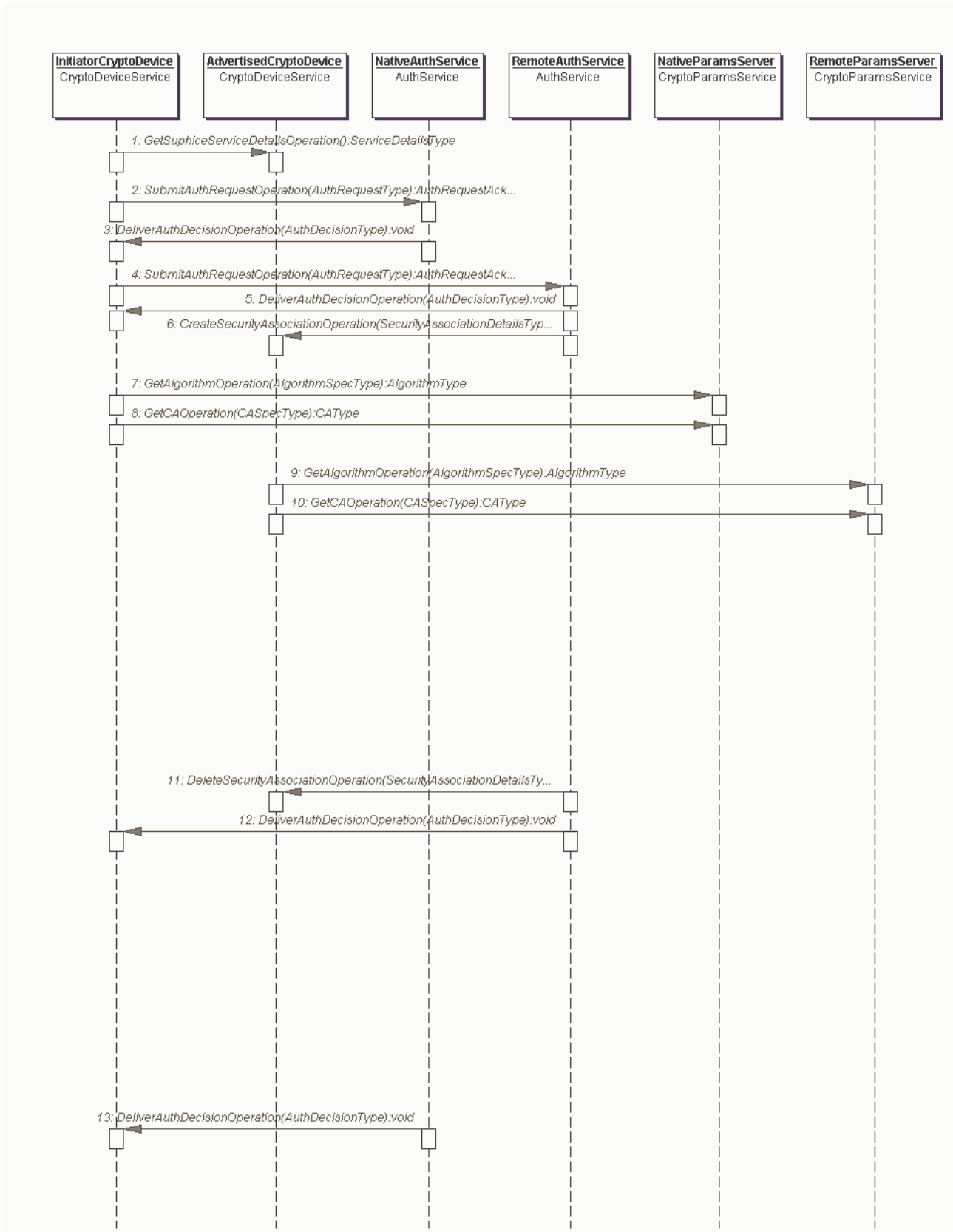


Figure 3 – SUPHICE Interfaces Sequence Diagram

3 Interface Design Considerations

The following sections provide some information relating to interface design considerations.

3.1 Message Format

The payload of a message passed between elements participating in a SUPHICE system will contain an XML document, rather than a number of object parameters. The use of XML documents means that messages can be validated against a defined XML schema in the web service's interaction layer, before being passed to the processing layer, thus ensuring that only valid messages reach the processing layer. Another advantage of XML documents is that the XML schema can evolve over time and document validation will continue to work without requiring modification to the source code. Furthermore, an XML representation of, for example, an authorisation request can stand alone as a legal business document which has meaning outside of the context of the web services call. There are some disadvantages of using XML documents, for example the additional processing overhead of creating, validating, and parsing the XML documents. The additional overhead of XML tags and other formatting introduced through the use of XML documents is not thought to be significant when compared with object parameters. In general a web services solution is unlikely to optimise the use of network bandwidth. However, in cases where use of network bandwidth is an important issue, steps can be taken to minimise its use. For example, a filter can be employed at the service endpoint to compress data using a non-proprietary compression algorithm (e.g. gzip), before it is put onto the wire. The data is still sent using HTTP, however the HTTP headers are changed to indicate that the type of the data is binary/gzip, rather than text/xml. A similar filter may be employed at the client end to decompress the received data.

3.2 Interface Development Approach

Two approaches are possible when developing a web services interface. One approach is to specify the interface in the target implementation language and use a tool to transform this into WSDL. The other approach is to specify the interface in WSDL and use a tool to transform this into the target implementation language. The implementation language first approach is tempting because it is almost certainly easier to specify the interface in a native language and have a tool generate the WSDL. However, a degree of control over the WSDL creation is lost if this approach is used. This loss of control over the generated WSDL can result in a less extensible and more unstable service definition. For example it may be impossible to evolve the service definition without forcing a change to the generated WSDL, which could have a knock on effect requiring changes to the service's clients. The WSDL first approach, on the other hand, allows for full control over all aspects of the service definition, resulting in stable, and extensible interfaces. The disadvantage is that a greater knowledge of WSDL [6] is required, together with knowledge of the WS-I Basic Profile [7] to ensure service interoperability. SUPHICE is aiming to define a set of stable service interfaces, which it is hoped will be widely adopted throughout the EC. Therefore, a WSDL-first approach has been used for the definition of the SUPHICE web service interfaces.

3.3 Security

Given the nature of SUPHICE it is clear that security of the web services interfaces is of paramount importance. The list below provides a summary of the security issues to be considered.

- **Authentication** – web service providers may need to be sure who is making requests of them, and web service clients need to be sure that they are making a request of a genuine service.
- **Authorisation** – having authenticated the client a web service provider must further ensure that the client is authorised to make a particular request.
- **Confidentiality** – ensures that only the web service provider and client involved in a particular conversation are able to decipher the messages sent between the two parties.
- **Integrity** – ensures that messages are not altered as they travel across the network.
- **Non-repudiation** – ensures that the sending party cannot later deny having sent a message and that the receiving party cannot later deny having received as well.
- **Availability** – interface should always be available and resilient to a denial of service attack.

This document does not attempt to prescribe any particular solution to securing the web services interfaces, as this will be considered in detail as part of SUPHICE Task 1.4 (Unplanned Services Security Analysis). However, a number of possible options have been identified and these are briefly described below.

1. Application layer security, e.g. digitally signing and encrypting the XML documents passed between components. There are a number of emerging Java specifications to help in this area, e.g. JSR 105 - XML Digital Signature APIs [11], JSR 106 - XML Digital Encryption APIs [12].
2. Transport layer security, e.g. defining security constraints within the web services interface specification that restrict access to the service to HTTP over SSL.
3. Network layer security, e.g. using additional encryption devices to protect authorisation servers, parameters servers, and services registries.

3.4 WSDL and XML Schema Versioning

SUPHICE web service interface definitions and associated XML schema type definitions may need to evolve over the lifetime of the project. In order to manage this evolution a date-based version identifier has been incorporated into the target namespace of WSDL service and XSD type definitions. The change in namespace effectively notifies applications of a change to the service or schema (because the application would not recognise the new namespace). Action must then be taken to evaluate any compatibility problems with the new service or schema. This approach has the advantage that an application cannot blindly continue to use a changed service or schema definition with potentially dangerous consequences. However, there is also a disadvantage in that applications must be updated to use the new namespace. The general format of SUPHICE defined namespaces is shown below:

For XSD schemas, namespaces are of the form:

```
http://suphice.com/schemas/<name>/<yyyy_mm_dd>
```

For WSDL services, namespaces are of the form:

```
http://suphice.com/services/<name>/<yyyy_mm_dd>
```

Where *<name>* is replaced with the name of the schema or service (e.g. authservice), and *<yyyy_mm_dd>* is replaced with the release date of the service or schema (e.g. 2005_03_17).

4 Crypto Services Registry

SUPHICE services are published in a UDDI registry so that potential users can discover them. This technique results in loose coupling between a network of service providers and service consumers. This section defines how the standard UDDI publishing and inquiry interfaces are used within the SUPHICE architecture.

The UDDI specifications define a mechanism to publish and discover information about Web Services. In this case the term *Web Services* is used to describe specific business functionality exposed by an organisation, usually through an Internet connection, for the purpose of providing a means for another organisation or software program to use the service.

The core component of the UDDI project is the UDDI business registry. Conceptually, the information provided in a UDDI business registry consists of three components: *white pages* including address, contact, and known identifiers; *yellow pages* including industrial categorisations based on standard taxonomies; and *green pages*, the technical information about services that are exposed by the organisation. Green pages include references to specifications for Web Services, as well as support for pointers to various file and URL based discovery mechanisms if required. For more information about UDDI refer to <http://www.uddi.org>.

4.1 Use Cases

The use case diagram in Figure 4 shows how UDDI is used within the SUPHICE architecture. The three publication based use cases are described in Section 4.2. The discovery use case is described in Section 4.3.

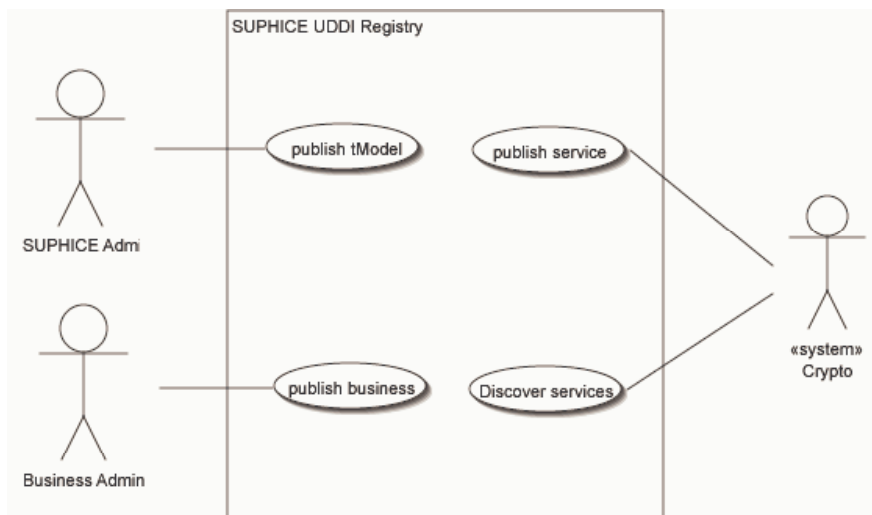


Figure 4 – Crypto Services Registry Use Case Diagram

4.2 Publishing Services

The sequence diagram in Figure 5 shows how SUPHICE services are published in a UDDI registry. This process is broken down into three phases. In the first stage a SUPHICE system administrator must initialise the registry by publishing SUPHICE defined tModels to the registry. This stage need only be performed once. Next, business administrators can publish details of their organisation to the registry. This must be performed once per organisation. Finally crypto devices can publish the service they provide to the registry. This must be performed once per service. A crypto device must know the identification key of the parent organisation and the SUPHICE CryptoDeviceService tModel before it can publish a service. The business administrator must provide this information to the device. These stages are described in more detail in the following sections.

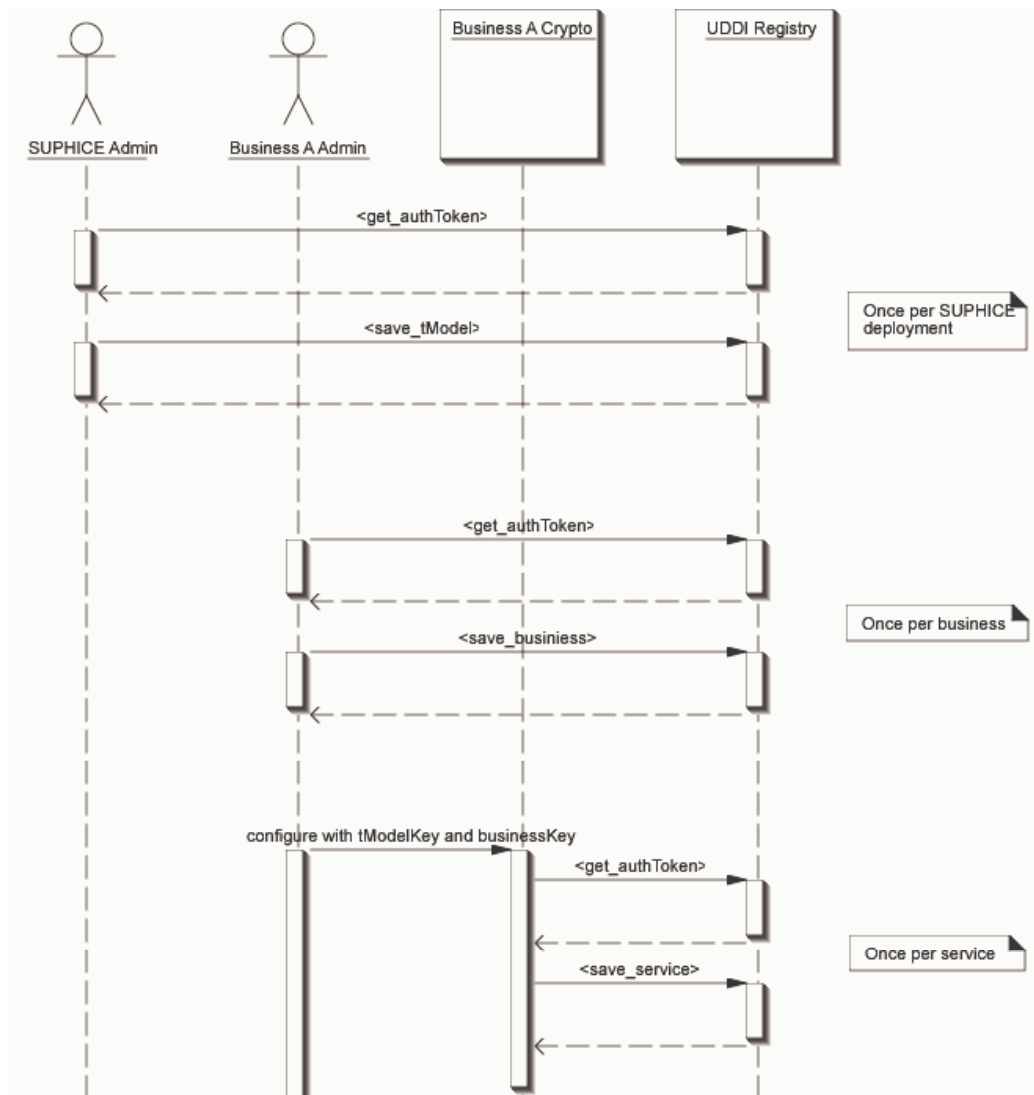


Figure 5 – Service Publication Sequence Diagram

4.2.1 Publishing tModels Describing SUPHICE Web Service Types

Each type of SUPHICE Web Service is described in a WSDL document that contains the `wSDL:message`, `wSDL:portType` and `wSDL:binding` elements, but not the `wSDL:service` and `wSDL:port` elements since they are specific to a deployed web service. In a live system these WSDL documents would be made available at a URL corresponding to the document namespace, as shown below:

`http://suphice.com/services/authservice/2005_12_31`

`http://suphice.com/services/cryptodeviceservice/2005_12_31`

`http://suphice.com/services/cryptoparamsservice/2005_12_31`

Implementations of the SUPHICE `CryptoDeviceService` will be advertised in a UDDI registry. SUPHICE has defined a UDDI `<tModel>` that will be published in the registry for this web service type. This `<tModel>` serves to describe the web service type and includes a reference to the WSDL document in the `<overviewURL>` element. The `<tModel>` is categorised with the `soapSpec` and `wSDLSpec` types from the `uddi-org:types` category system.

An example `<save_tModel>` operation that could be used to publish a web service conforming to the SUPHICE `CryptoDeviceService` web service type is shown below. The `tModelKey` is a zero length string that is assigned by the registry the first time the `<tModel>` is published. The assigned `tModelKey` is

contained in the <tModelDetail> data structure that is returned by the save operation. Subsequent publications of the same <tModel> must specify the actual key.

```
<save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo>authorisation token returned from get_authToken</authInfo>
  <tModel tModelKey="key assigned by the registry">
    <name>suphice-com:services:cryptodeviceservice:2005_12_31</name>
    <description>
      Web service type for SUPHICE CryptoDeviceService
    </description>
    <overviewDoc>
      <description>WSDL Service Interface Document</description>
      <overviewURL>
        http://suphice.com/services/cryptodeviceservice/2005_12_31?wsdl
      </overviewURL>
    </overviewDoc>
    <categoryBag>
      <keyedReference
        tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
        keyName="uddi-org:types" keyValue="types:wsdlSpec"/>
      <keyedReference
        tModelKey="UUID:C1ACF26D-9672-4404-9D70-39B756E62AB4"
        keyName="uddi-org:types" keyValue="types:soapSpec"/>
    </categoryBag>
  </tModel>
</save_tModel>
```

The tModelKey that has been assigned to the web service type suphice-com:services:cryptodeviceservice:2005_12_31 is shown below:

uuid:39210D30-665D-11DA-BA0C-917373408E73.

4.2.2 Publishing Organisations Offering SUPHICE Web Services

An organisation that provides SUPHICE services is modelled with a UDDI <businessEntity> element. The organisation need only publish its details to the UDDI registry once.

The following example shows how the UDDI <save_business> operation can be used by an organisation to publish details about itself in the UDDI registry.

```
<save_business generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo> authorisation token returned from get_authToken</authInfo>
  <businessEntity businessKey="">
    <name>UK Fire Service</name>
    <description>
      Provides fire extinguishing and rescue services throughout the UK
    </description>
    <contacts>
      <contact useType="Technical Support">
        <description>Contact for access to SUPHICE services</description>
        <personName>Joe Bloggs</personName>
        <phone>+44 (118) 9876543</phone>
        <email>joe.bloggs@fire.uk</email>
      </contact>
    </contacts>
  </businessEntity>
</save_business>
```

4.2.3 Publishing Implementations of SUPHICE Web Services

A UDDI <businessService> element is published in the UDDI registry for each implementation of the SUPHICE CryptoDeviceService web service type. The <businessService> contains a <bindingTemplate> that corresponds to the wsdl:port element. An <accessPoint> is contained within the <bindingTemplate> element. This must specify the location of a crypto device that implements the SUPHICE CryptoDeviceService. The <bindingTemplate> also contains a <tModelInstanceDetails> element that references the SUPHICE <tModel> used to specify the type of the web service implementation.

The information contained in the <categoryBag> element is used to categorise the service according to the UDDI General Keywords Category System. SUPHICE has defined a number of key names (described in Section 4.2.4) that must be used by crypto devices to categorise the service they are publishing. These keys may be used as search criteria when querying the registry.

The following example shows how a crypto device can use the UDDI <save_service> operation to publish a SUPHICE service in the UDDI registry.

```
<save_service generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo> authorisation token returned from get_authToken </authInfo>
  <businessService
    businessKey="business key for the outer busniessEntity"
    serviceKey="">
    <name>Fire Control HQ CryptoDeviceService</name>
    <description>
      The CryptoDeviceService of the UK Fire Service's Fire Control HQ
    </description>
    <bindingTemplates>
      <bindingTemplate bindingKey="">
        <accessPoint URLType="http">
          http://control.fire.uk/suphice/CryptoDeviceService
        </accessPoint>
        <tModelInstanceDetails>
          <tModelInstanceInfo tModelKey="uuid:39210D30-665D-11DA-BA0C-917373408E73">
            <instanceDetails>
              <overviewDoc>
                <overviewURL>
                  http://control.fire.uk/suphice/CryptoDeviceService?wsdl
                </overviewURL>
              </overviewDoc>
            </instanceDetails>
          </tModelInstanceInfo>
        </tModelInstanceDetails>
      </bindingTemplate>
    </bindingTemplates>
    <categoryBag>
      <keyedReference
        keyName="suphice.com:LANDetails.name"
        keyValue="Fire Control HQ"
        tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
      <keyedReference
        keyName="suphice.com:LANDetails.securityLevel"
        keyValue="Restricted"
        tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
      <keyedReference
        keyName="suphice.com:LANDetails.organisation"
        keyValue="UK Fire Service"
        tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
      <keyedReference
        keyName="suphice.com:LANDetails.location"
        keyValue="Reading"
        tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
      <keyedReference
        keyName="suphice.com:DeviceDetails.deviceModel"
        keyValue="Thales e-Security Datacryptor 2000"
        tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
    </categoryBag>
  </businessService>
</save_service>
```

4.2.4 Definition of Key Names for Categorising SUPHICE Web Services

SUPHICE has defined a number of Key Names to be used with the UDDI General Keywords taxonomy for categorising SUPHICE CryptoDeviceServices. By categorising services in this way clients are able to search the registry for a sub-set of the registered services that match their requirements. The following key names have been defined.

```

suphice.com:LANDetails.name
suphice.com:LANDetails.securityLevel
suphice.com:LANDetails.organisation
suphice.com:LANDetails.location
suphice.com:DeviceDetails.deviceModel

```

4.3 Discovering Services

The sequence diagram in Figure 6 shows how SUPHICE services are discovered in a UDDI registry. This process is invoked when a user of the crypto device's management tool initiates a search for SUPHICE services.

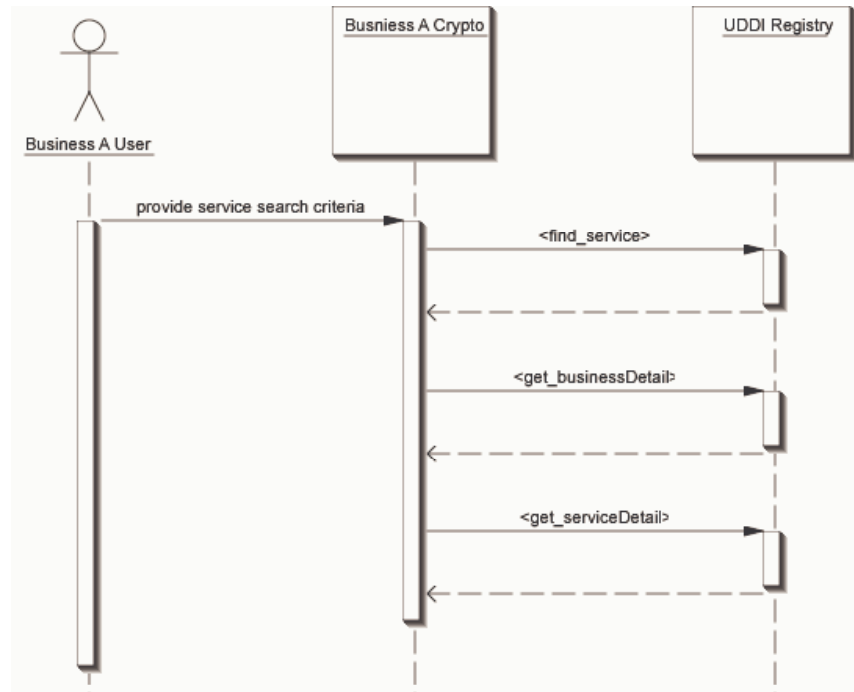


Figure 6 – Service Discovery Sequence Diagram

4.3.1 Discovering Implementations of SUPHICE Web Services

Consumers of SUPHICE web services can use the service categorisation key names specified in Section 4.2.4 to define search criteria to filter the services returned by the `<find_service>` operation.

The example `<find_service>` operation shown below can be used to query the registry for SUPHICE `CryptoDeviceServices` offered by the *UK Fire Service*, operating at a security level of *Restricted*, and located in *London*. The name of the service and the business under which the service is registered are unspecified. Only implementations of the SUPHICE `CryptoDeviceService` will be returned due to the inclusion of the `<tModelKey>` describing this type in the `<tModelBag>`.

PUBLIC

```
<find_service businessKey="" generic="2.0" xmlns="urn:uddi-org:api_v2">
  <findQualifiers>
    <findQualifier></findQualifier>
  </findQualifiers>
  <name>%</name>
  <categoryBag>
    <keyedReference
      keyName="suphice.com:LANDetails.organisation"
      keyValue="UK Fire Service"
      tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
    <keyedReference
      keyName="suphice.com:LANDetails.securityLevel"
      keyValue="Restricted"
      tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
    <keyedReference
      keyName="suphice.com:LANDetails.location"
      keyValue="London"
      tModelKey="uuid:a035a07c-f362-44dd-8f95-e2b134bf43b4"/>
  </categoryBag>
  <tModelBag>
    <tModelKey>uuid:39210D30-665D-11DA-BA0C-917373408E73</tModelKey>
  </tModelBag>
</find_service>
```

The <find_service> operation returns a <service_list> data structure containing the businessKey, serviceKey and name of matching services. Further information about the returned services can be obtained by invoking the <get_businessDetail> and <get_serviceDetail> operations.

5 AuthService

The `AuthService` provides an interface to SUPHICE capable cryptographic devices that allows the devices to submit authorisation requests. A device submits an authorisation request either because it wants to provide a service to another device, or because it wants to use a service provided by another device. When a device submits a valid authorisation request a unique request identifier is immediately returned to the device. The authorisation request is queued for later processing by the Authorisation Server's business logic, which uses a mix of rules-based technology and delegation to a human authority via a web-browser to reach a decision. When a decision has been reached it is conveyed to the requesting device via a call to the `DeliverAuthDecisionOperation` of the `CryptoDeviceService`. The location of this web-service, which must be implemented by the requesting device, is conveyed to the Authorisation Server as part of the initial request from the device.

In addition to the service documentation provided in this document the complete definition of the `AuthService` is available in the WSDL file `AuthService.wsdl`. The XML document types used by this service are defined in the XSD file `Suphice.xsd` and additional schema documentation is provided in `SuphiceXSD.html`.

5.1 SubmitAuthRequestOperation

5.1.1 Description

A SUPHICE capable encryption device can use the `SubmitAuthRequestOperation` either to submit a request to use a service provided by another device, or to submit a request to provide a service to another device. The prototype for this operation is shown below.

```
AuthRequestAck SubmitAuthRequestOperation(AuthRequestType authRequest)
```

Note that `AuthRequestType` is an abstract type. When a device submits a request to its native authorisation server to permit it to provide a service to another device it must use the concrete type `NativeAuthRequestType`. When a device submits a request to a remote authorisation server to permit it to use a service provided by another device it must use the concrete type `RemoteAuthRequestType`. The `RemoteAuthRequestType` contains additional elements that allow the calling device to indicate the cryptographic parameters that it would like to use. This is illustrated in the examples in Section 5.1.3 below.

5.1.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
<code>AuthRequest</code>	Yes	<code>AuthRequestType</code>	Details of the authorisation request for which approval is sought.

5.1.3 Sample Request

A sample request document is shown below for a request to a native authorisation server.

```

<?xml version="1.0" encoding="UTF-8"?>
<NativeAuthRequest
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <InitiatorServiceDetails>
    <ContactDetails>
      <FirstName>Mark</FirstName>
      <LastName>Irons</LastName>
      <Email>mark.irons@host.domain</Email>
      <Telephone>+44 188 9234567</Telephone>
    </ContactDetails>
    <DeviceDetails>
      <Name>dc2k.fire.uk</Name>
      <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
      <PublicIP>10.0.0.1</PublicIP>
      <CryptoDeviceServiceURL>
        http://10.0.0.1/services/cryptodeviceservice
      </CryptoDeviceServiceURL>
      <AuthServiceURL>
        http://10.0.0.100/services/authservice
      </AuthServiceURL>
    </DeviceDetails>
    <LANDetails>
      <Name>Fire Control HQ</Name>
      <SecurityLevel>Restricted</SecurityLevel>
      <Organisation>UK Fire Service</Organisation>
      <Location>Reading</Location>
    </LANDetails>
  </InitiatorServiceDetails>
  <AdvertisedServiceDetails>
    <ContactDetails>
      <FirstName>Sarah</FirstName>
      <LastName>Butler</LastName>
      <Email>sarah.butler@host.domain</Email>
      <Telephone>+44 118 92301234</Telephone>
    </ContactDetails>
    <DeviceDetails>
      <Name>dc2k.ambulance.de</Name>
      <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
      <PublicIP>10.0.0.2</PublicIP>
      <CryptoDeviceServiceURL>
        http://10.0.0.2/services/cryptodeviceservice
      </CryptoDeviceServiceURL>
      <AuthServiceURL>
        http://10.0.0.200/services/authservice
      </AuthServiceURL>
    </DeviceDetails>
    <LANDetails>
      <Name>Ambulance HQ</Name>
      <SecurityLevel>Restricted</SecurityLevel>
      <Organisation>Ambulance Service</Organisation>
      <Location>Germany</Location>
    </LANDetails>
  </AdvertisedServiceDetails>
  <StartDate>2005-03-17T08:00:00</StartDate>
  <EndDate>2005-04-01T17:00:00</EndDate>
</NativeAuthRequest>

```

A sample request document is shown below for a request to a remote authorisation server. This request contains additional elements to indicate the name of the algorithm and key that the calling device would like to use. This information is specified by the native authorisation server.

```

<?xml version="1.0" encoding="UTF-8"?>
<RemoteAuthRequest
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <InitiatorServiceDetails>
    <ContactDetails>
      <FirstName>Mark</FirstName>
      <LastName>Irons</LastName>
      <Email>mark.irons@host.domain</Email>
      <Telephone>+44 188 9234567</Telephone>
    </ContactDetails>
    <DeviceDetails>
      <Name>dc2k.fire.uk</Name>
      <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
      <PublicIP>10.0.0.1</PublicIP>
      <CryptoDeviceServiceURL>
        http://10.0.0.1/services/cryptodeviceservice
      </CryptoDeviceServiceURL>
      <AuthServiceURL>
        http://10.0.0.100/services/authservice
      </AuthServiceURL>
    </DeviceDetails>
    <LANDetails>
      <Name>Fire Control HQ</Name>
      <SecurityLevel>Restricted</SecurityLevel>
      <Organisation>UK Fire Service</Organisation>
      <Location>Reading</Location>
    </LANDetails>
  </InitiatorServiceDetails>
  <AdvertisedServiceDetails>
    <ContactDetails>
      <FirstName>Sarah</FirstName>
      <LastName>Butler</LastName>
      <Email>sarah.butler@host.domain</Email>
      <Telephone>+44 118 92301234</Telephone>
    </ContactDetails>
    <DeviceDetails>
      <Name>dc2k.ambulance.de</Name>
      <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
      <PublicIP>10.0.0.2</PublicIP>
      <CryptoDeviceServiceURL>
        http://10.0.0.2/services/cryptodeviceservice
      </CryptoDeviceServiceURL>
      <AuthServiceURL>
        http://10.0.0.200/services/authservice
      </AuthServiceURL>
    </DeviceDetails>
    <LANDetails>
      <Name>Ambulance HQ</Name>
      <SecurityLevel>Restricted</SecurityLevel>
      <Organisation>Ambulance Service</Organisation>
      <Location>Germany</Location>
    </LANDetails>
  </AdvertisedServiceDetails>
  <StartDate>2005-03-17T08:00:00</StartDate>
  <EndDate>2005-04-01T17:00:00</EndDate>
  <CAName>Universal</CAName>
  <AlgorithmName>3DES</AlgorithmName>
</RemoteAuthRequest>

```

5.1.4 Response Parameters

On success the SubmitAuthRequestOperation returns an AuthRequestAck which contains a unique request identifier. If unsuccessful, a SuphiceError construct is returned. Note that a decision on the request is not returned by this operation. The decision is conveyed to the caller asynchronously at some time in the future when the Authorisation Server invokes the DeliverAuthDecisionOperation on the CryptoDeviceService. This web-service must be implemented by the calling device at a URL specified in the AuthRequest. The table below shows details of this operation's response parameters.

Parameter	Type	Description
AuthRequestAck	AuthRequestAckType	Acknowledges to the caller that the authorisation request was received and has been queued for processing. Includes

		a unique request identifier that the caller can use to relate a received authorisation decision to an issued request.
--	--	---

5.1.5 Sample Response

A sample response document is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthRequestAck
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmldom="http://www.w3.org/2005/05/xmldom"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <AuthRequestId>12345</AuthRequestId>
</AuthRequestAck>
```

5.1.6 Errors

A list of possible errors that may be returned by this operation is show below. For a full description of SUPHICE error codes refer to Section 8.

CouldNotProcessAuthRequest

6 CryptoDeviceService

All cryptographic devices that participate in a SUPHICE system must implement the `CryptoDeviceService`. Implementation of this service is one of the criteria that must be fulfilled if a device is to qualify as SUPHICE capable. The operations defined by the `CryptoDeviceService` are invoked by an Authorisation Server and are used to inform the device of a decision on a previously submitted authorisation request, and to tell the device to create or delete a security association with a peer device. The `CryptoDeviceService` also defines an operation that is invoked by other cryptographic devices to obtain a complete description of the service provided by the device.

Note that for demonstration purposes it may not be possible to implement a web-service inside the cryptographic device, for example due to memory or processing limitations of the demonstration devices. In this case a software proxy may provide the implementation of the `CryptoDeviceService` on behalf of the device. A proprietary interface would exist between the software proxy and the device, which would forward web-services requests to the device. Further discussion of such a software proxy and the proprietary proxy to device interface are outside the scope of this document.

In addition to the service documentation provided in this document the complete definition of the `CryptoDeviceService` is available in the WSDL file *CryptoDeviceService.wsdl*. The XML document types used by this service are defined in the XSD file *Suphice.xsd* and additional schema documentation is provided in *SuphiceXSD.html*.

6.1 DeliverAuthDecisionOperation

6.1.1 Description

A SUPHICE Authorisation Server can use the `DeliverAuthDecisionOperation` to notify a SUPHICE capable encryption device of the outcome of a previously submitted authorisation request. The `DeliverAuthDecisionOperation` can also be used to revoke a previously accepted request. The prototype for this operation is shown below.

```
void DeliverAuthDecisionOperation(AuthDecisionType authDecision)
```

Note that `AuthDecisionType` is an abstract type. When an Authorisation Server invokes this operation it must use one of the concrete types `AcceptAuthDecisionType` or `RejectAuthDecisionType` to indicate whether a request was accepted or rejected. The `AcceptAuthDecisionType` contains additional elements that allow the Authorisation Server to indicate the cryptographic parameters that must be used, and the location of a SUPHICE Parameters Server where these parameters can be obtained. The `RejectAuthDecisionType` contains an additional element that allows the Authorisation Server to specify a reason why a request was rejected.

6.1.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
<code>authDecision</code>	Yes	<code>AuthDecisionType</code>	Details of the outcome of a previously submitted authorisation request, including an identifier to allow the receiver to correlate the decision to a request.

6.1.3 Sample Request

A sample request for the case where an authorisation request has been granted is shown below. In this example the Authorisation Server has mandated that the request is granted on condition that the *Universal CA* is used with the *Triple DES* algorithm. The request will also specify where the CA and algorithm can be obtained.

```
<?xml version="1.0" encoding="UTF-8"?>
<AcceptAuthDecision
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <AuthRequestId>12345</AuthRequestId>
  <CAName>Universal</CAName>
  <AlgorithmName>3DES</AlgorithmName>
  <CryptoParamsServiceURL>http://suphice.eu/ParamsService</CryptoParamsServiceURL>
</AcceptAuthDecision>
```

A sample request for the case where an authorisation request is denied is shown below. Note this message also contains a reason that the request was rejected.

```
<?xml version="1.0" encoding="UTF-8"?>
<RejectAuthDecision
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <AuthRequestId>67890</AuthRequestId>
  <Reason>Insufficient security clearance level</Reason>
</RejectAuthDecision>
```

6.1.4 Response Parameters

This operation has no response parameters.

6.1.5 Errors

This operation throws no errors.

6.2 CreateSecurityAssociationOperation

6.2.1 Description

A SUPHICE Authorisation Server can use the CreateSecurityAssociationOperation to instruct a SUPHICE capable encryption device to create a new security association to the specified peer, using the specified algorithm and CA. Only an Authorisation Server that is native to the cryptographic device can invoke this operation and this could be enforced through the use of XML digital signatures to sign the SOAP messages. The prototype for this operation is shown below.

```
void CreateSecurityAssociationOperation(
  SecurityAssociationDetails securityAssociationDetails)
```

6.2.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
SecurityAssociationDetails	Yes	SecurityAssociationDetails	Details of the security association to create, including the address of the remote peer, the algorithm to use, and the key material to use.

6.2.3 Sample Request

A sample request for the case where a device is requested to create a security association to a peer device whose public interface address is 10.0.0.1, using the *Universal CA* and *Triple DES* algorithm is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurityAssociationDetails
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmlmime="http://www.w3.org/2005/05/xmlmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <DeviceDetails>
    <Name>dc2k.fire.uk</Name>
    <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
    <PublicIP>10.0.0.1</PublicIP>
    <CryptoDeviceServiceURL>
      http://10.0.0.1/services/cryptodeviceservice
    </CryptoDeviceServiceURL>
    <AuthServiceURL>
      http://10.0.0.100/services/authservice
    </AuthServiceURL>
  </DeviceDetails>
  <AlgorithmName>3DES</AlgorithmName>
  <CAName>Universal</CAName>
</SecurityAssociationDetails>
```

6.2.4 Response Parameters

This operation has no response parameters.

6.2.5 Errors

This operation throws no errors.

6.3 DeleteSecurityAssociationOperation

6.3.1 Description

A SUPHICE Authorisation Server can use the DeleteSecurityAssociationOperation to instruct a SUPHICE capable encryption device to delete a security association to the specified peer, due to a previously agreed authorisation being revoked. Only an Authorisation Server that is native to the cryptographic device can invoke this operation. The prototype for this operation is shown below.

```
void DeleteSecurityAssociationOperation(
  SecurityAssociationDetails securityAssociationDetails)
```

6.3.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
SecurityAssociationDetails	Yes	SecurityAssociationDetails	Details of the security association to delete, including the address of the remote peer, the algorithm to use, and the key material used by the association.

6.3.3 Sample Request

A sample request for the case where a device is requested to delete a security association to a peer device whose public interface address in 10.0.0.1 is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurityAssociationDetails
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmlmime="http://www.w3.org/2005/05/xmlmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <DeviceDetails>
    <Name>dc2k.fire.uk</Name>
    <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
    <PublicIP>10.0.0.1</PublicIP>
    <CryptoDeviceServiceURL>
      http://10.0.0.1/services/cryptodeviceservice
    </CryptoDeviceServiceURL>
    <AuthServiceURL>
      http://10.0.0.100/services/authservice
    </AuthServiceURL>
  </DeviceDetails>
  <AlgorithmName>3DES</AlgorithmName>
  <CAName>Universal</CAName>
</SecurityAssociationDetails>
```

6.3.4 Response Parameters

This operation has no response parameters.

6.3.5 Errors

This operation throws no errors.

6.4 GetSuphiceServiceDetailsOperation

6.4.1 Description

This operation is provided to allow a remote cryptographic device to discover full service details from this device before the remote device issues an authorisation request to its native Authorisation Server. The prototype for this operation is shown below.

```
ServiceDetailsType GetSuphiceServiceDetailsOperation()
```

6.4.2 Request Parameters

The operation has no request parameters

6.4.3 Response Parameters

A sample response document is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceDetails
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <ContactDetails>
    <FirstName>Mark</FirstName>
    <LastName>Irons</LastName>
    <Email>mark.irons@host.domain</Email>
    <Telephone>+44 188 9234567</Telephone>
  </ContactDetails>
  <DeviceDetails>
    <Name>dc2k.fire.uk</Name>
    <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
    <PublicIP>10.0.0.1</PublicIP>
    <CryptoDeviceServiceURL>
      http://10.0.0.1/services/encryptdeviceservice
    </CryptoDeviceServiceURL>
    <AuthServiceURL>
      http://10.0.0.100/services/authservice
    </AuthServiceURL>
  </DeviceDetails>
  <LANDetails>
    <Name>Fire Control HQ</Name>
    <SecurityLevel>Restricted</SecurityLevel>
    <Organisation>UK Fire Service</Organisation>
    <Location>Reading</Location>
  </LANDetails>
</ServiceDetails>
```

6.4.4 Errors

This operation throws no errors.

7 CryptoParamsService

The CryptoParamsService defines operations that can be used by SUPHICE capable encryption devices to download encryption algorithms, key material, and software functionality modules. A device will typically invoke these operations to obtain data that is required before a secure connection can be established with a peer device. The location of a suitable Parameters Server is provided to the cryptographic device as part of the authorisation decision.

In addition to the service documentation provided in this document the complete definition of the CryptoParamsService is available in the WSDL file *CryptoParamsService.wsdl*. The XML document types used by this service are defined in the XSD file *Suphice.xsd* and additional schema documentation is provided in *SuphiceXSD.html*.

7.1 GetAlgorithmOperation

7.1.1 Description

A SUPHICE capable encryption device can use the GetAlgorithmOperation to download an encryption algorithm from a Cryptographic Parameters server. The prototype for this operation is shown below.

```
AlgorithmType GetAlgorithmOperation(AlgorithmSpecType algorithmSpec)
```

7.1.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
algorithmSpec	Yes	AlgorithmSpecType	A specification of the required encryption algorithm, including the name of the algorithm and the type of the target device.

7.1.3 Sample Request

A sample request document is shown below. In this example the *Triple DES* algorithm is being requested, suitable for installation into a *Thales e-Security Datacryptor 2000* device.

```
<?xml version="1.0" encoding="UTF-8"?>
<AlgorithmSpec
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <AlgorithmName>3DES</AlgorithmName>
  <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
</AlgorithmSpec>
```

7.1.4 Response Parameters

On success GetAlgorithmOperation returns an Algorithm which includes a binary implementation suitable for installation into the requesting device. If unsuccessful, a SuphiceError construct is returned. The table below shows the response parameters for this operation.

Parameter	Type	Description
algorithm	AlgorithmType	The requested encryption algorithm, including the specification used in the request and a hex encoded binary representation of the algorithm suitable for installation into the target device.

7.1.5 Sample Response

A sample response document is shown below. The HexBinaryData element contains the hex encoded binary algorithm implementation. The xmlmime:contentType attribute on the HexBinaryData element indicates the MIME media type and will always be application/octet-stream.

```
<?xml version="1.0" encoding="UTF-8"?>
<Algorithm
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmlmime="http://www.w3.org/2005/05/xmlmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <AlgorithmSpec>
    <AlgorithmName>3DES</AlgorithmName>
    <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
  </AlgorithmSpec>
  <HexBinaryData xmlmime:contentType="application/octet-stream">
    DEADBEEF
  </HexBinaryData>
</Algorithm>
```

7.1.6 Errors

A list of possible errors that may be returned by this operation is show below. For a full description of SUPHICE error codes refer to Section 8.

AlgorithmNotFound

7.2 GetCAOperation

7.2.1 Description

A SUPHICE capable encryption device can use the GetCAOperation to download key material from a Cryptographic Parameters server. The prototype for this operation is:

```
CAType GetCAOperation(CASpecType caSpec)
```

7.2.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
cASpec	Yes	CASpecType	A specification of the required CA, including the name of the CA and the type of the target device.

7.2.3 Sample Request

A sample request document is shown below. In this example the name of the CA being requested is *Universal*, and is suitable for a *Thales e-Security Datacryptor 2000* device.

```
<?xml version="1.0" encoding="UTF-8"?>
<CASpec
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmlmime="http://www.w3.org/2005/05/xmlmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <CAName>Universal</CAName>
  <DeviceModelType>Thales e-Security Datacryptor 2000</DeviceModelType>
</CASpec>
```

7.2.4 Response Parameters

On success GetCAOperation returns a CA which includes a binary representation of the CA suitable for installation into the requesting device. If unsuccessful, a SuphiceError construct is returned. The table below shows the response parameters for this operation.

Parameter	Type	Description
cA	CAType	The requested CA, including the specification used in the request and a hex encoded binary representation of the CA suitable for installation into the target device.

7.2.5 Sample Response

A sample response document is shown below. HexBinaryData element contains a hexadecimal encoded representation of the binary CA. The xmlmime:contentType attribute on the HexBinaryData element indicates the MIME media type and will always be application/octet-stream.

```
<?xml version="1.0" encoding="UTF-8"?>
<CA
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmlmime="http://www.w3.org/2005/05/xmlmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <CASpec>
    <CAName>Universal</CAName>
    <DeviceModelType>Thales e-Security Datacryptor 2000</DeviceModelType>
  </CASpec>
  <HexBinaryData xmlmime:contentType="application/octet-stream">
    CAFEBABE
  </HexBinaryData>
</CA>
```

7.2.6 Errors

A list of possible errors that may be returned by this operation is show below. For a full description of SUPHICE error codes refer to Section 8.

CANotFound

7.3 GetSoftwareModuleOperation

7.3.1 Description

A SUPHICE capable encryption device can use the GetSoftwareModuleOperation to download a software module (e.g. an IPv6 module) from a Cryptographic Parameters server. The prototype for this operation is:

```
SoftwareModuleType GetSoftwareModuleOperation(
  SoftwareModuleSpecType softwareModuleSpec)
```

7.3.2 Request Parameters

The table below shows details of this operation's request parameters.

Parameter	Required?	Type	Description
-----------	-----------	------	-------------

softwareModuleSpec	Yes	SoftwareModuleSpecType	A specification of the required software module, including the name of the module and the type of the target device.
--------------------	-----	------------------------	--

7.3.3 Sample Request

A sample request document is shown below. In this example the name of the module being requested is IPv6, and is suitable for a *Thales e-Security Datacryptor 2000* device.

```
<?xml version="1.0" encoding="UTF-8"?>
<SoftwareModuleSpec
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <SoftwareModuleName>IPv6</SoftwareModuleName>
  <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
</SoftwareModuleSpec>
```

7.3.4 Response Parameters

On success GetSoftwareModuleOperation returns a SoftwareModule which includes a binary representation of the module suitable for installation into the requesting device. If unsuccessful, a SuphiceError construct is returned. The table below shows the response parameters for this operation.

Parameter	Type	Description
SoftwareModule	SoftwareModuleType	The requested software module, including the specification used in the request and a hex encoded binary representation of the module suitable for installation into the target device.

7.3.5 Sample Response

A sample response document is shown below. HexBinaryData element contains a hexadecimal encoded representation of the binary software module. The xmllmime:contentType attribute on the HexBinaryData element indicates the MIME media type and will always be application/octet-stream.

```
<?xml version="1.0" encoding="UTF-8"?>
<SoftwareModule
  xmlns="http://suphice.com/schemas/suphice/2005_12_31"
  xmlns:xmllmime="http://www.w3.org/2005/05/xmllmime"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://suphice.com/schemas/suphice/2005_12_31 Suphice.xsd">
  <SoftwareModuleSpec>
    <SoftwareModuleName>IPv6</SoftwareModuleName>
    <DeviceModel>Thales e-Security Datacryptor 2000</DeviceModel>
  </SoftwareModuleSpec>
  <HexBinaryData xmllmime:contentType="application/octet-stream">
    FEEDFACE
  </HexBinaryData>
</SoftwareModule>
```

7.3.6 Errors

A list of possible errors that may be returned by this operation is show below. For a full description of SUPHICE error codes refer to Section 8.

SoftwareModuleNotFound

8 Error Codes

All SUPHICE web services operations may return an error using the defined XML document `SuphiceErrorType`. This type conveys information about the error to the client, including an error type code and a message describing the error. All defined error types are documented in the table below.

Error Name	Description
<code>CouldNotProcessAuthRequest</code>	Indicates that the authorisation request could not be processed.
<code>UnknownAuthRequest</code>	Indicates that the specified auth request was unknown.
<code>AlgorithmNotFound</code>	Indicates that the specified encryption algorithm could not be found.
<code>CANotFound</code>	Indicates that the specified CA could not be found.
<code>SoftwareModuleNotFound</code>	Indicates that the specified software module could not be found.
<code>CoundNotCreateSecurityAssociation</code>	Indicates that a device could not create a security association.
<code>CouldNotDeleteSecurityAssociation</code>	Indicates that a device could not delete a security association.

9 References

- [1] formal/03-03-01, OMG Unified Modelling Language Specification, Version 1.5, OMG, March 2003, <http://www.omg.org/cgi-bin/doc?formal/03-03-01>
- [2] D31-O-CO-002-TRT Architecture Definition, Version 2.0, TRT, December 2005
- [3] W3C Recommendation - XML Schema Part 0: Primer, Second Edition, W3C, 28 Oct 2004, <http://www.w3.org/TR/xmlschema-0/>
- [4] W3C Recommendation - XML Schema Part 1: Structures, Second Edition, W3C, 28 Oct 2004, <http://www.w3.org/TR/xmlschema-1/>
- [5] W3C Recommendation - XML Schema Part 2: Datatypes, Second Edition, W3C, 28 Oct 2004, <http://www.w3.org/TR/xmlschema-2/>
- [6] Web Services Description Language (WSDL), 1.1, W3C, 15 Mar 2001, <http://www.w3.org/TR/wsdl>
- [7] WS-I Basic Profile, Version 1.1 (Final), WS-I, 24 Aug 2004, <http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html>
- [8] W3C Working Group Note - Describing Media Content of Binary Data in XML, W3C, 4 May 2005, <http://www.w3.org/TR/xml-media-types/>
- [9] Spring Java/J2EE Application Framework, <http://www.springframework.org/>
- [10] UDDI Version 3.0.2, OASIS, 19 Oct 2004, <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>
- [11] JSR 105: XML Digital Signature APIs, <http://www.jcp.org/en/jsr/detail?id=105>
- [12] JSR 106: XML Digital Encryption APIs, <http://www.jcp.org/en/jsr/detail?id=106>