Reference Architecture for the National Forest Information System

January 7, 2002

Submitted to: The NFIS Project Office, Pacific Forestry Centre, Victoria, BC

Submitted by: Ron Lake, Galdos Systems Inc.
Mike Power, FirstMark Technologies Ltd.
Henry Kucera, Swiftsure Spatial Systems Inc.
-- Contractor team for the development of the Technical Vision and Reference Architecture for the NFIS.

Contractor Team Contact: Mike Power, FirstMark Technologies Ltd.
Suite 300, 16 Concourse Gate,
Ottawa, Ontario K2E 7S8
Tel: (613) 723-8020
Email: mpower@firstmark.ca

Table of Contents

GLOSSARY

API	Application Programming Interface
ASP	Active Server Page (Microsoft)
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance.
CCFM	Canadian Council of Forest Ministers
CGDI	Canadian Geo-spatial Data Infrastructure
CGI	Common Gateway Interface (HTTP Server mechanism to invoke scripts or other executable programs).
CORBA	Common Object Request Broker Architecture (OMG Distributed Computing Platform).
COM	Component Object Model (Microsoft)
COVERAGE	Distribution function for a property over a portion of the earth's surface (OGC term)
CSA	Canada's National Sustainable Forest Management System Standard (CSA)CAN/CSAZ908
DCOM	Distributed Component Object Model. Microsoft Distributed Computing Platform.
DNA	Distributed Network Architecture (Microsoft)
EJB	Enterprise Java Bean. Transactional component in J2EE System. Equivalent to COM object under MTS in Microsoft DNA.
Feature	OGC Term denoting a concrete or abstract geographic object.
FSC	Forest Stewardship Council
GML	OGC Geography Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over SSL (secure HTTP)
IIOP	Internet Inter-Orb Protocol (a CORBA wire protocol)
ISO	International Standards Organization
J2EE	Java 2 Enterprise Edition (Java Web Services Platform)
JSP	Java Server Page – J2EE Equivalent of ASP in DNA

MTS	Microsoft Transaction Server (provides transaction control of COM objects within DNA environment).
.NET	Microsoft Web Services platform
NFIS	National Forest Information System
NGO	Non-Government Organization
OGC	OpenGIS Consortium
ORPC	Object Remote Procedure Call (a DCOM Wire Protocol)
RMI	Remote Method Invocation (part of J2EE Distributed Computing Platform)
SFI	Sustainable Forestry Initiative Program
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
UDDI	Universal Description Discovery and Integration
VB	Visual Basic (Microsoft)
WCS	Web Coverage Service (OGC)
WFS	Web Feature Service (OGC)
WMS	Web Map Service (OGC)
WSDL	Web Services Description Language (W3C)
W3C	World Web Wide Consortium
XML	eXtensible Markup Language (W3C)

1. INTRODUCTION

The team of FirstMark Technologies Ltd., Galdos Systems Inc., and Swiftsure Spatial Systems Inc. is pleased to submit this report covering the Technical Reference Architecture for the Canadian Council of Forest Ministers' National Forest Information System, a national infrastructure for forest information.

The main objective of the Technical Reference Architecture is to describe the logical components of NFIS and the standards on which these components and their interfaces are based.

The Reference Architecture was developed on the results of NFIS workshops held on the NFIS in British Columbia, Ontario and Newfoundland under the auspices of the Canadian Forest Service, and with the participation of forestry, environment and other staff from the Provinces of British Columbia, Manitoba, Ontario and Newfoundland.

This Reference Architecture is consistent with the NFIS vision as described in the document, A Technical Vision for the National Forest Information System [1]. The notion of a technical architecture is defined in the C4ISR (Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance) Architecture Framework [2] that was adopted as the global architecture framework for the NFIS project. The C4ISR Architecture Framework breaks down the architecture into operational, technical and system viewpoints with a conceptual data model (Figure 1.1).

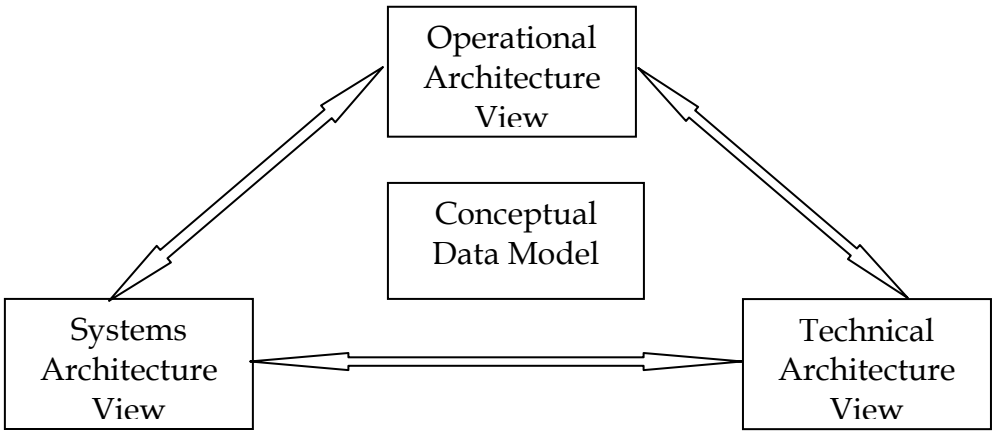


Figure 1.1 C4ISR Architecture Framework

Section 2 of this document provides a brief overview of the NFIS requirements on which this Reference Architecture is based. These requirements are also elaborated in the NFIS target vision document [1].

Section 3 provides a high level view of the Operational Architecture and contains some high-level use cases that motivated the design material presented in the Technical Reference Architecture of Section 5.

Section 4 provides an overview of the concept of service based architecture and is included as background technical material for the Technical Reference Architecture presented in Section 5.

The Technical Reference Architecture material is organized into several sections. Section 5.1 provides an architecture overview while Sections 5.2 through 5.6 provides additional details on the key architecture components including service type registry, service instance registry and services. Sections 5.4 through 5.6 are written from a user centric viewpoint and describe the functions of service connection (supplying a service), service discovery and service invocation. This section also provides references to the key technical standards on which the Technical Reference Architecture is based.

Section 6 provides a brief discussion of possible implementations of the Technical Reference Architecture and discusses related systems architectural issues. This section provides realizations of the concepts presented in Section 5 to depict the implementation and use of an NFIS constructed on this architecture.

2. REVIEW OF REQUIREMENTS

The fundamental motivation for the development of the National Forest Information System (NFIS) is to provide a system that offers an ongoing and integrated view of the state of forested lands in Canada. Such a system is viewed as a strategic tool for reporting on sustainable forest management and for the positioning of the Canadian forest industry in relation to its global competitors.

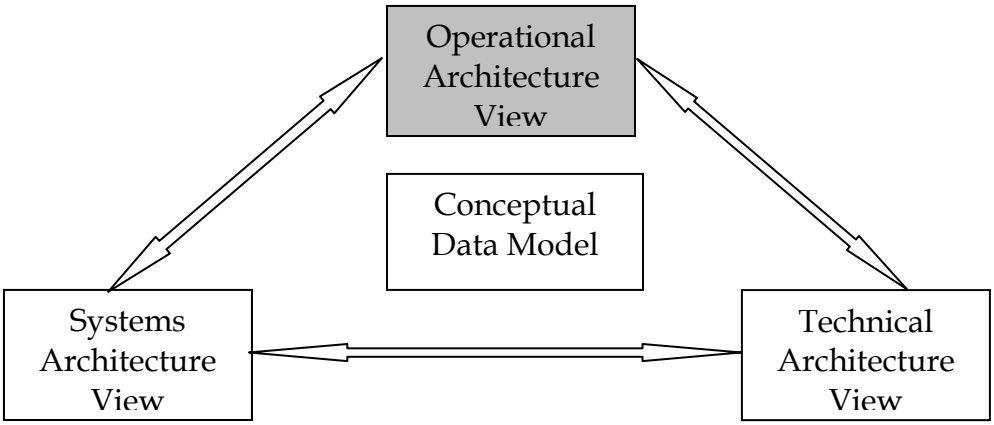
The fulfilment of this objective requires that NFIS offer a high degree of flexibility and an ability to integrate services from a wide variety of disciplines and jurisdictions. This objective also requires NFIS to:

- Provide services to NFIS users. NFIS users include members of provincial, territorial, federal and municipal governments, forest industry and non-government agencies (NGO), and members of the general public who are concerned with sustainable forest management (SFM).
- Provide services by NFIS participants. Participants are members of the same communities as the NFIS users. Services can be developed and deployed independent of the NFIS, and then made available to NFIS users through adherence to NFIS standards for service interfaces.
- Provide services that are jurisdiction-neutral. Jurisdictional data and service branding must be maintained and displayed where required.
- Provide services as "close to the source" as possible. The provinces, territories or similar local or regional agencies will provide most NFIS services. NFIS may provide integration capabilities and other national services as required for performance or other reasons.
- Provide the SFM component of the Canadian Geospatial Data Infrastructure (CGDI). NFIS will not re-invent or provide services that can already be obtained from other parts of CGDI. At the same time, NFIS can be expected to provide services that are used by non-NFIS users within the CGDI. Users can originate from the Canadian and international resource management communities.
- Provide services in a non-intrusive manner. NFIS service providers can continue to provide normal, non-NFIS access to their services and data while supporting standard NFIS interfaces for NFIS-based users.
- Provide one or more national services through the NFIS Project Office for service discovery. These services will be integrated into CGDI.
- Use the security provisions of the service provider. These security provisions will never be overridden by NFIS.

- Support the integration of services in both of Canada's official languages. This requirement means that service requests, input parameters, and service responses can be in either English or French.
- Provide levels of performance that are acceptable to its various user communities. For example, generation of a rolled-up national forest inventory will be feasible within 24 hours whereas generation of a particular map based on the rolled-up data should be feasible in less than a minute.

3. OPERATIONAL ARCHITECTURE

The operational architecture view is a description of the tasks and activities, operational elements, and information flows required to accomplish or support a given task or operation.



3.1 Overview

This section describes the operational architecture for the National Forest Information System.

The operational architecture view is a "**description of the tasks and activities, operational elements, and information flows required to accomplish or support a given task or operation"**. With this definition, the description of the operational begins with the identification of the users of NFIS and some of the key tasks or operations these users are expected to perform.

As noted in Section 2, the users of NFIS include members from a broad range of organizations in the private and public sectors. These members can be grouped into two main roles or user categories:

1. Service Consumers
2. Service Providers

Service Consumers
Service Consumers are users that make use of or consume the services accessible through NFIS. These services include those for data access, data processing and data presentation. Service Consumers include the public, environmentalists,

industrial foresters, park managers, water managers, land developers, engineering contractors, silviculturists, and wildlife biologists.

Service Providers
Service Providers are users that support data access by developing, connecting and maintaining the services provided through NFIS. Service Providers include database administrators, software developers, and systems analysts in municipal, federal, provincial and territorial governments, resource management and other agencies, including those in the private sector.

Using these two broad user categories one can define a number of representative use cases as described in the following subsections.

3.2 Use Case #1 – Data Request

The Data Request Use Case is shown in Figure 3.2-1.

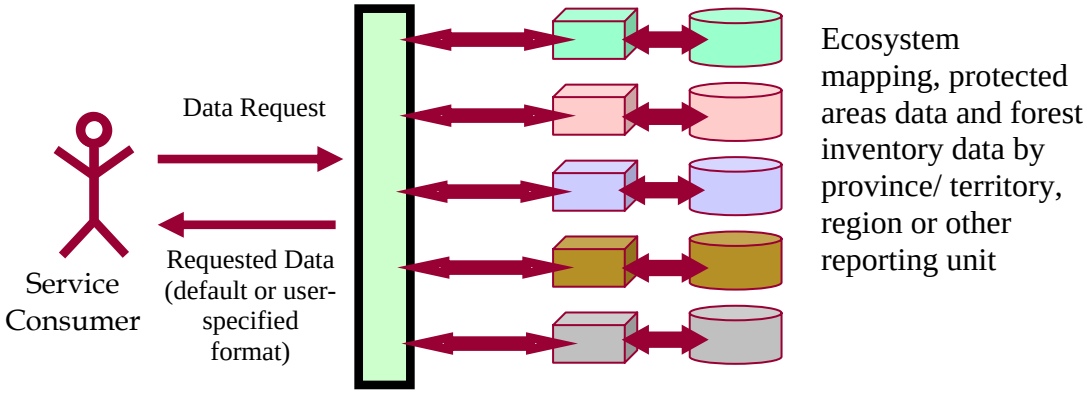


Figure 3.2-1. Data Request Use Case Example

The Data Request Use Case involves a request issued by the Service Consumer (actor) to one or more data access Service Providers, the access to the Service Providers being mediated by the NFIS.

Use Case Description	
Name	Data Request
Priority	High
Description	This use case allows a Service Consumer to request data from a data access Service.
Precondition	N/A

Flow of Events – Basic Path	
Step 1.	Service Consumer sends a data access request. The request contains the area of interest, and other parameters that describe the desired data.
Step 2.	NFIS sends an acknowledgment to the Service Consumer that the request has been received.
Step 3.	NFIS verifies that the Service Consumer is authorized to make the request. NFIS validates the request via an access control service operated by the targeted Service Provider. Details are documented in the Distributed Access Control System design documentation.
Step 4.	NFIS sends a response to the Service Consumer. The response contains the request status and the requested data.
Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a data access request. The request contains the area of interest, and other parameters that describe the desired data.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3.	The Service Consumer may be unauthorized to perform this operation. If so, the NFIS sends a response with status "failed" and an explanatory message.
Postcondition	N/A.

3.3 Use Case #2 – Data Presentation Request (a).

This use case is illustrated in Figure 3.3-1. In this use case, the Service Consumer provides presentation data to the presentation service. These data may have been locally generated or obtained from a Data Access Service.

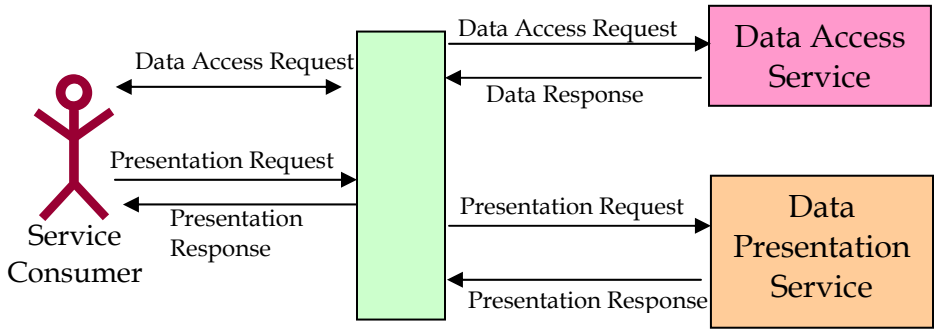


Figure 3.3-1. Data Presentation Request

Use Case Description	
Name	Data Presentation Request
Priority	High
Description	This use case allows a Service Consumer to request the presentation of data supplied by the Service Consumer. As indicated in Figure 3.3-1 the data to be presented is supplied directly by the Service Consumer.
Precondition	N/A

Flow of Events – Basic Path	
Step 1.	Service Consumer sends a data presentation request. The request contains the data to be presented together with parameters that describe the character of the desired presentation.
Step 2.	NFIS sends an acknowledgment to the Service Consumer that the request has been received.
Step 3.	NFIS verifies that the Service Consumer is authorized to request the data presentation. This process may involve NFIS sending a request to the targeted Service Provider to determine if the Service Consumer is authorized to make the request.
Step 4.	NFIS sends the requested presentation to the Service Consumer.

Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a data presentation request. The request contains the data to be presented together with parameters that describe the character of the desired presentation.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3	The Service Consumer may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and an explanatory message.
Postcondition	N/A.

3.4 Use Case #3 – Data Presentation Request (b).

This use case is shown in Figure 3.4-1. In this use case, the Data Presentation Service obtains the presentation data from a Data Access Service using a reference provided by the Service Consumer.

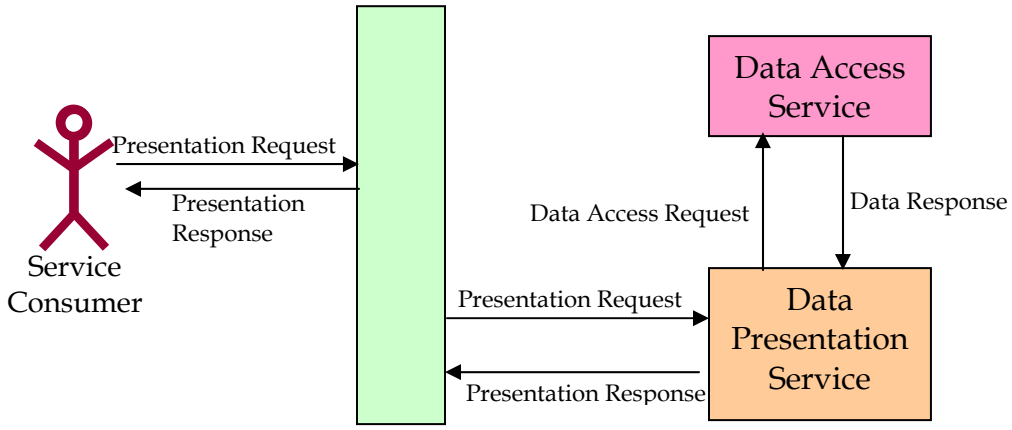


Figure 3.4-1. Data Presentation Request

Use Case Description	
Name	Data Presentation Request
Priority	High
Description	This use case allows a Service Consumer to request the presentation of data supplied by a Data Access Service. As indicated in Figure 3.4-1, the presentation data are obtained from a Data Access Service by the Data Presentation Service using a reference supplied by the Service Consumer.
Precondition	N/A

Flow of Events – Basic Path	
Step 5.	Service Consumer sends a data presentation request. The request contains a reference to the data to be presented together with parameters that describe the character of the desired presentation.
Step 6.	NFIS sends an acknowledgment to the Service Consumer that the request has been received.
Step 7.	NFIS verifies that the Service Consumer is authorized to request the data presentation. This process may involve the NFIS sending a request to the targeted Service Provider to determine if the Service Consumer is authorized to make the request.
Step 8.	NFIS sends the requested presentation to the Service Consumer.
Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a data presentation request. The request contains a reference to the data to be presented together with parameters that describe the character of the desired presentation.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3	The Service Consumer may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and an explanatory message.
Postcondition	N/A.

3.5 Use Case #4 – Data Processing Request (a)

This section describes the Data Processing Request Use Case (a). An overview of the use case can be seen in Figure 3.5-1. In this use case, the data to be processed is locally generated by the Service Consumer and obtained from a Data Access Service.

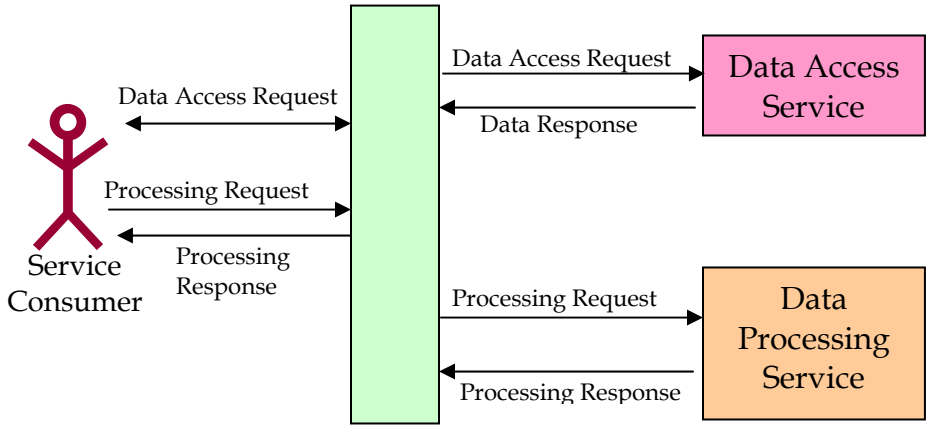


Figure 3.5-1 Data Processing Request (a) – User Supplied Data

The use case details are presented in the following table.

Use Case Description	
Name	Data Processing Request
Priority	High
Description	This use case allows a Service Consumer to request the processing of data supplied by the Service Consumer. The data to be processed are supplied directly by the Service Consumer
Precondition	N/A
Flow of Events – Basic Path	
Step 1.	Service Consumer sends a data processing request. The request contains the data to be processed together with parameters that describe the data processing characteristics.
Step 2.	NFIS sends an acknowledgment to the Service Consumer that the request has been received.

Use Case Description	
Step 3.	NFIS verifies that the Service Consumer is authorized to request the data processing operation. This process may involve the NFIS sending a request to the targeted Service Provider to determine if the Service Consumer is authorized to make the request.
Step 4.	NFIS sends the requested presentation to the Service Consumer.
Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a data processing request. The request contains the data to be processed together with parameters that describe the character of the desired data processing.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3.	The Service Consumer may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and a descriptive message.
Postcondition	N/A.

3.6 Use Case #5 – Data Processing Request (b)

This section describes the Data Processing Request Use Case (b). An overview of the use case can be seen in Figure 3.6-1. In this use case, the data to be processed are obtained from a Data Access Service by the Data Processing Service.

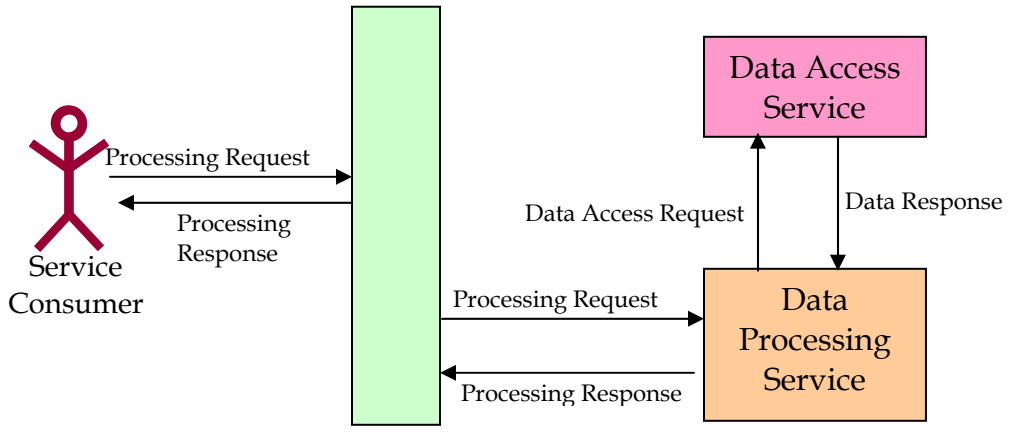


Figure 3.6-1 Data Processing Request

The use case details are presented in the following table.

Use Case Description	
Name	Data Processing Request
Priority	High
Description	This use case allows a Service Consumer to request the processing of data supplied by the Service Consumer. The data to be processed are obtained by the Data Processing Service from a Data Access Service using a reference supplied by the Service Consumer.
Precondition	N/A
Flow of Events – Basic Path	
Step 5.	Service Consumer sends a data processing request. The request contains a reference to the data to be processed together with parameters that describe the data processing characteristics.
Step 6.	NFIS sends an acknowledgment to the Service Consumer that the request has been received.
Step 7.	NFIS verifies that the Service Consumer is authorized to request the data processing operation. This process may involve the NFIS sending a request to the targeted Service Provider to determine if the Service Consumer is authorized to make the request.
Step 8.	NFIS sends the requested presentation to the Service Consumer.
Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a data processing request. The request contains a reference to the data to be processed together with parameters that describe the character of he desired data processing.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3.	The Service Consumer may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and a descriptive message.
Postcondition	N/A.

3.7 Use Case #6 – Connecting a Service

This use case concerns the connection of a Service by a Service Provider to NFIS. The use case is shown in Figure 3.7-1.

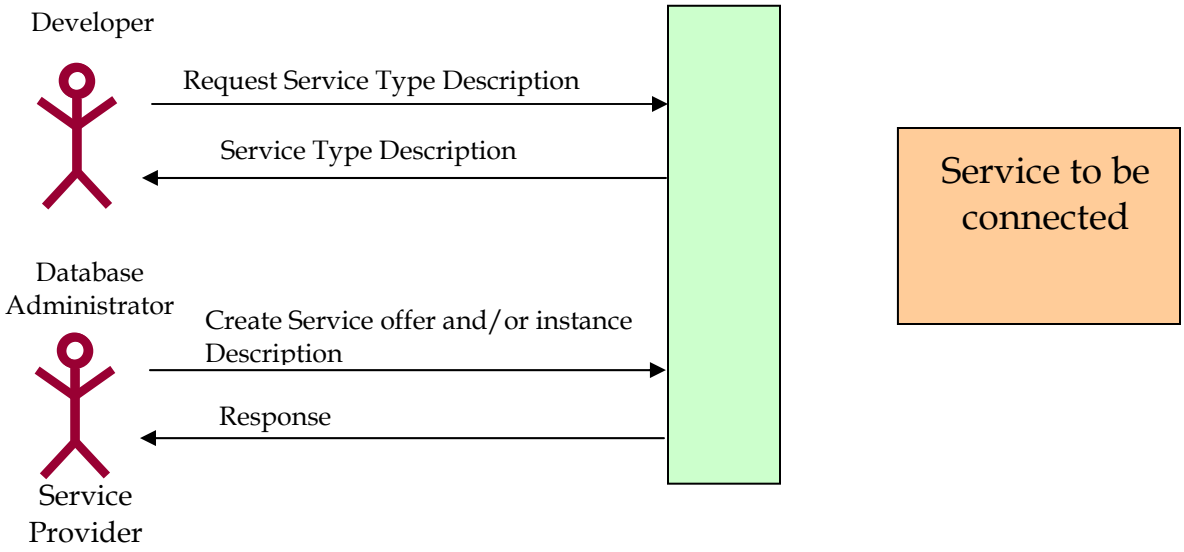


Figure 3.7-1. Connecting a Service to the NFIS

The use case details are presented in the following table.

Use Case Description	
Name	Connecting a Service to the NFIS
Priority	High
Description	This use cases allows a Service Provider to request the connection of a Service to the NFIS.
Precondition	N/A
Flow of Events – Basic Path	
Step 1.	Service Provider sends a request to obtain the description for a Service Type. The request contains the name of the desired Service Type.
Step 2.	NFIS sends an acknowledgment to the Service Provider that the request has been received.
Step 3.	NFIS verifies that the Service Provider is authorized to request the Service connection.
Step 4.	NFIS sends the requested services type description to the Service Provider.
Flow of Events – Alternative Paths	

Use Case Description	
Step 1.	Service Provider sends a request to obtain the description for a service type. The request contains the name of the desired service type.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3	The Service Provider may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and a descriptive message.
Postcondition	N/A.

3.8 Use Case #7 – Discovering a Service

This use case concerns the discovery of a service instance by a Service Consumer to the NFIS. The use case is shown in Figure 3.8-1.

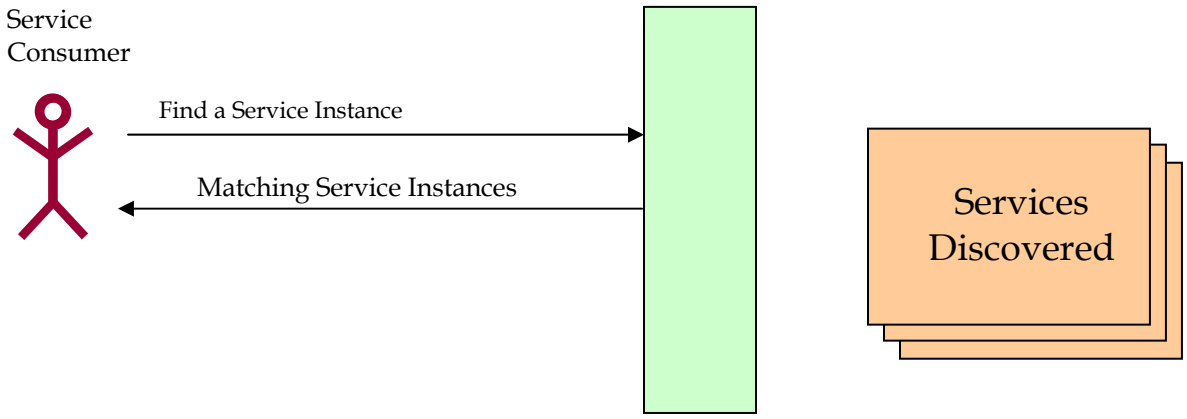


Figure 3.8-1. Connecting a Service to the NFIS

The use case details are presented in the following table.

Use Case Description	
Name	Finding a Service on the NFIS
Priority	High
Description	A Service Consumer finds a Service on the NFIS

Use Case Description	
Precondition	N/A
Flow of Events – Basic Path	
Step 1.	Service Consumer sends a request to obtain a list of Service Instances. The request contains parameters that define the services being sought.
Step 2.	NFIS sends an acknowledgment to the Service Provider that the request has been received.
Step 3.	NFIS verifies that the Service Provider is authorized to request the service instance information.
Step 4.	NFIS sends the requested services instance descriptions to the Service Consumer.

Flow of Events – Alternative Paths	
Step 1.	Service Consumer sends a request to obtain a list of service instances. The request contains parameters that define the services being sought.
Step 2.	The request may be invalid. If so, then the acknowledgment contains a rejection rather than an approval.
Step 3	The Service Consumer may be unauthorized to perform this operation. If so, NFIS sends a response with status "failed" and a descriptive message.
Post condition	N/A.

3.9 NFIS Network Administration

The previous sections have described use case scenarios for two broad classes of user: the Service Consumer and the Service Provider. These scenarios address the core functions to be delivered through the NFIS.

A third class of activity relates to administration and connectivity of NFIS at the network level. The details of these activities are dependent on network business practices yet to be defined and adopted by NFIS participants. As these business practices are identified, the operational architecture must be adapted in support. For now it is possible to identify issues that will arise in the distributed, service-based architecture being developed in this reference architecture, as follows:

- Creation and management of "trusted server" network among participating jurisdictions. This "trusted server" network would support core NFIS services such as Service Registries.
- Management of the Service Registries that are considered part of the NFIS core services. This issue includes provision of personnel and decision processes to authorize the registration of proposed service offerings in a particular Service Registry. This process includes the registration of non-core NFIS Service Registries and the registration of services offered by NFIS participants. Similar procedures are required for deletion or modification of Service Registry entries.
- General management of the Service Registry framework to support NFIS policies.
- Propagation of local changes to network peers.
- Maintenance of user and user class accounts with associated privileges for access to NFIS Service Registries and Services. The Service Provider controls access to specific Services, but the user class may filter users prior to this access control point.
- Maintenance of system backups, disaster recovery plans/procedures, and related support Services to ensure the integrity of the core NFIS Services, and the ability of NFIS to provide connectivity to all registered Service offerings.

3.10 Division of Roles

As a network of organizations, NFIS operation will depend upon the collaboration of jurisdiction staff and NFIS Project Office staff. A possible division of responsibilities is outlined in the Table 1.

Data/Information Providers **(federal, provincial, territorial and other key information providers)**	**NFIS Project Office**
Operate custodial servers	Operate national servers
Customize NFIS Reference Services (analysis, synthesis and report generation, *etc.*) to work in local environments	Design, build and implement NFIS Reference Services and underlying data standards
Implement and maintain NFIS services on custodial servers • Information services from custodial sources • Respond to cascading service requests from peer servers	Implement and maintain NFIS services on national server(s) • Information services from federal sources • Data integration/fusion from multiple NFIS servers • Cascading service requests to custodial

• Custodial registries, catalogues	servers • National registries, catalogues
Manage custodial data and metadata	Manage national data and metadata
Collectively define national data models	Provide requirements for national data models
Map custodial data to national data models	Adopt and maintain national data models
Aggregate data/information to small scale representations	Produce national information products for third parties (e.g. National Atlas, NGO, etc.)
Implement NFIS user authentication and access control services using organizational security infrastructure	Implement NFIS user authentication and access control services for users not served by a "home organization"
Participation in NFIS Technology Working Group • Liaison with NFIS Project Office • Provide input for future development; identify organization constraints and requirements	NFIS technical communications and outreach • Chair NFIS Technical Working Group, liaison with custodial representatives • National liaison/connectivity with IUFRO GFIS

Table 1. NFIS Roles

4. SERVICE-BASED ARCHITECTURE CONCEPT

4.1 Introduction

This section outlines the concept of a Service-Based Architecture [3]. In a Service-Based Architecture all system functionality is provided in terms of services, that is, in terms of a logical set of abstract interfaces plus non-operational service properties. A service is thus a contract between the Service Provider and the Service Consumer, the contract details being carried in the services interface definitions. This concept is illustrated in Figure 4.1-1.

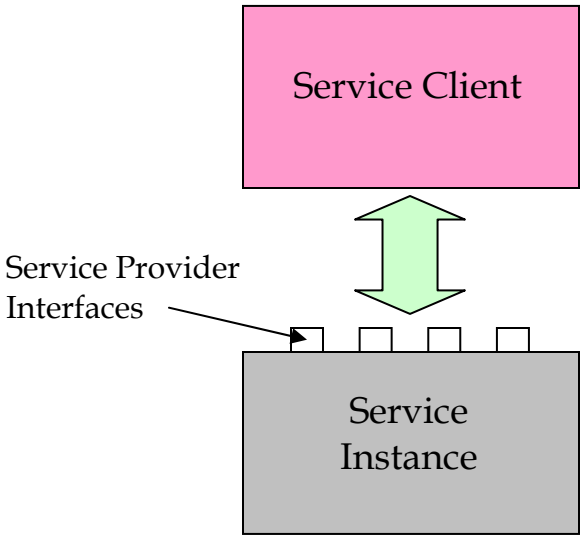


Figure 4.1-1. Service Concept

The Service Client shown in Figure 4.1-1 represents a program, a human user or some combination of the two.

The Service Instance is a software component such as an application process, Servlet, Active Server Page, JSP, Enterprise Java Bean, CORBA or DCOM object that delivers specific functionality to the client through the service's interfaces at some specified network endpoint. While the specific details of these interfaces depend on the selected implementation technology, the service functionality is determined by the abstract description of these interfaces. For example, a Service Instance may exist, which operates on a given forest stand in some region of the country and computes the value of the stand. Abstractly, this Valuation Service might be written as:

Value = f(ForestStand)

The ForestStand is an object with properties such as its geometric extent, species composition and merchantable wood volume/unit area. The Valuation Service then uses these data together with other data known to the service (e.g. value per unit volume by species) to compute the standing timber value.

The representation of these attributes and their routing between the Service Consumer and the Service Provider is implementation specific; however, these implementation details can be encapsulated within the particular implementation. The availability of technology bridges makes it possible to provide the abstract service functionality across a variety of implementation technologies.

Each service interface is composed of a logically related set of operations. Each operation in the service interface has one or more inputs, each of which has a specific data type. These characteristics define a unique "operation signature" for the service, which is discussed further in Section 4.2. In order to create a client that can interact with and utilize the service, a software developer can use knowledge of a service's operation signature, together with associated binding and protocol information.

Web services may be associated with web pages, but these two types of web components have important differences. A web page provides a visual or auditory presentation, and may provide "services" such as access to information or support for electronic shopping. A web service is more abstract and can be viewed as some piece of software that another piece of software communicates with in order to get something done. The web service, unlike a web page, is not associated with the presentation of visual information. In fact, a web page might access a web service and present the information provided by the web service in some form accessible to a human user. Web services support software access to functions or capabilities that may in turn be accessed by human users.

The distinction between web services and web pages is an important one from the point of view of the visibility of the supporting organization. When using a "Service" provided through a web page, users typically go to the providing organization's web page where they see the organization's name, logo and other information. When a piece of software uses a web service provided by an organization there may be no information presented to the human user about the service custodian. Means, other than direct visual presentation at the time of access, must be found to provide organizations with visibility and compensation for providing services. The issues of data branding and Service Provider compensation are outside of the scope of this document.

4.2 Operation Signature

The data types of the input and output of the operation define an Operation Signature. Given an operation F with n-inputs and m-outputs, the signature of F can be written as:

Signature(F) = (I1, .. In; O1, … Om)

Where I1, … In are the data types of the inputs of F and

O1, ... Om are the data types of the outputs of F.

Depending on the capabilities of the data-typing framework available for the signature expression, the semantics of the associated operations can be captured. With increased granularity of the typing framework, the operation semantics will have stronger representation through the operation signature.

The signature of the operation to compute the area of the forest polygon above can be used to illustrate these ideas. If the typing framework provides types for polygons and decimals (numbers), then the signature can be written as:

Signature(F) = (PolygonType; decimal)

One cannot tell from the signature that the output is an area, or that the input polygon is a forest polygon. If the typing framework provides polygon types and the area quantity (number and units of measure), then the signature more accurately represents the operation semantics as follows:

Signature(F) = (PolygonType; area)

Within a given typing framework the Operation Signature provides a basic means of service classification. Services that return collections of spatial features such as forest access roads, rivers and lakes, or that return distribution functions (coverages) for forest inventory or species habitat suitability can be considered as data oriented services. These data oriented services have operation signatures such as:

Signature(F) = (SpatialQueryExpression; SpatialFeatureCollection)

Signature(F) = (SpatialQueryExpression; SpatialCoverage)

Services that perform processing operations on geo-spatial data include the above area function example among many others. These services contain operations that have signatures such as:

Signature(F) = (LineGeometry; decimal)

Signature(F) = (LineGeometry; length)

Signature(F) = (AreaGeometry; area)

Signature(F) = (LineCoverage,a,b;
line integral of coverage function over [a,b])

4.3 Typing Framework

The previous section introduced the notion of an "operation signature" as a means of describing the operations of a service. This description depends on the existence of a set of data types that is shared across the infrastructure or across portions of the infrastructure community. This set of shared types is referred to as a "typing framework".

The Open GIS Consortium's (OGC) Geography Markup Language (GML) provides some of the key geo-spatial types required for the description of service interfaces. The OGC filter specification provides another key set of data types. A sample typing framework hierarchy is shown in Figure 4.3-1, with dependencies shown by dashed arrows. The figure shows both generic types and domain-specific types that might be developed by subject area interest groups (*e.g.*, forest inventory, ecosystem classification, *etc.)*

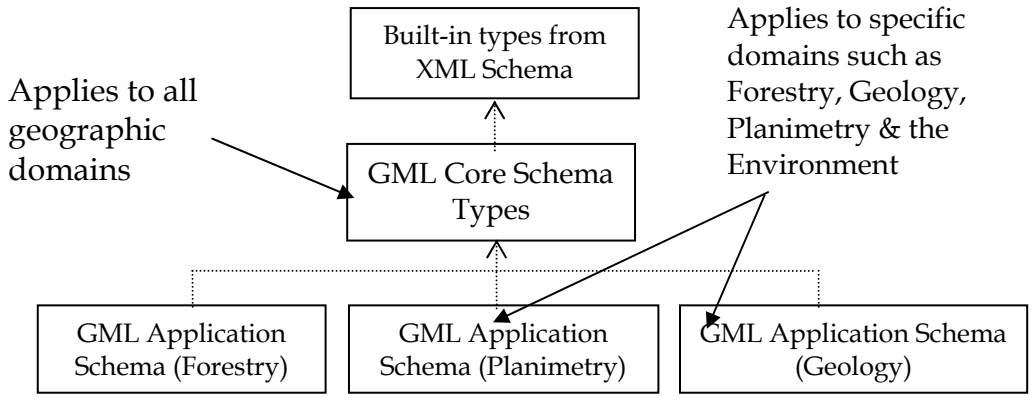


Figure 4.3-1. GML Typing Framework

Section 5.2 discusses domain-specific types that must be developed for SFM information.

To make this concept of typing framework clearer, a sample table of types upon which NFIS operations and services could rely is shown in the following table (See also section 5.2).

Data Type	Description (properties)
Forest stand	ForestStand(ID, extent, species, height, siteQuality)
Access Road	AccessRoad(ID, name, centreline, surfaceType)
Protected Area	ProtectedArea(ID, name, extent, classification)
Harvest Block	HarvestBlock(ID, extent, harvestMethod, ..)
Heritage Area	HeritageArea(ID, extent, classification)
Stream	Stream(ID, centreline, classification)
SeapageZone	SeapageZone(ID, extent)
Land Tenure	LandTenure(ID, extent, tenureType, holder)
Tree Farm License	TreeFarmLicense(ID, extent, …)
Range	Range(ID, extent,…)

4.4 Service-Based Architecture Logical Components

4.4.1 Overview

This section introduces the key logical components of a Service-Based Architecture. This is illustrated in Figure 4.4-1.

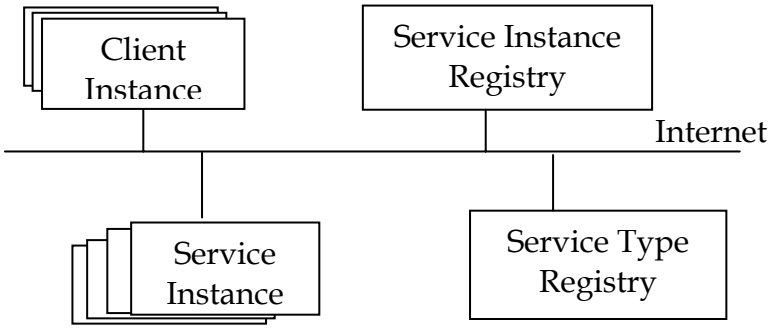


Figure 4.4-1. Service-Based Architecture Logical Components

These logical components, which are "connected" via the Internet in Figure 4.4-1 include:

- Service Instances
- Client Instances
- Service Instance Registry
- Service Type Registry

The architecture assumes that these components can communicate with another via the Internet. The specific means of communication is implementation dependent; this feature is discussed in greater detail in Section 6.

4.4.2 Client Instance(s)

In this document, the term "client" pertains to a component that facilitates interaction between a human user and a service provided by the infrastructure. The architecture assumes that services can communicate with one another and that these services can be considered peers of one another, depending on the implementation technology.

Clients discover Service Instances using discovery (search) mechanisms that exploit the metadata stored in the Service Type Registries and Service Instance Registries.

4.4.3 Service Type Registry

A service type registry is used to store the descriptions of services that are available within the infrastructure. Each service type (category of service) is described by an entry in the service type registry. This description provides a specification that can be used to implement a service instance or to invoke a service instance by a client through the service instance registry.

The Service Type Registry describes only service types and not service instances. The specifications in the Service Registry are thus defined in abstract terms only using the data types from a selected typing framework.

Some possible entries in a Service Registry for NFIS might include:

- OGC Feature Service (WFS).
- OGC Coverage Service (WCS).
- OGC Coordinate Dictionary Service.
- OGC Geo-coding Service.
- OGC Coordinate Transformation Service.
- Forest Inventory (to compute timber volume for a selected region).
- Distribution of Non-Harvestable Lands Service.
- Certification Service

The Service Type Registry contains metadata specific to each Service Instance.

4.4.4 Service Instance Registry

The Service Instance Registry describes service instances, whereby an instance of a Service Type is stored in one of the Service Type Registries. The Service Instance Registry with its stored specification provides sufficient information to invoke the referenced service. This service invocation may be carried out by either client or service instances. The Service Instance Registry thus provides information that describes the binding between the abstract interface types and the transport protocol supported in the service instance and provides addressing information.

The Service Instance Registry contains metadata specific to each Service Instance.

4.4.5 Service Instance(s)

A Service Instance provides an implementation of a Service Type. Each Service Instance is registered in a Service Instance Registry providing metadata that assists in Service Discovery.

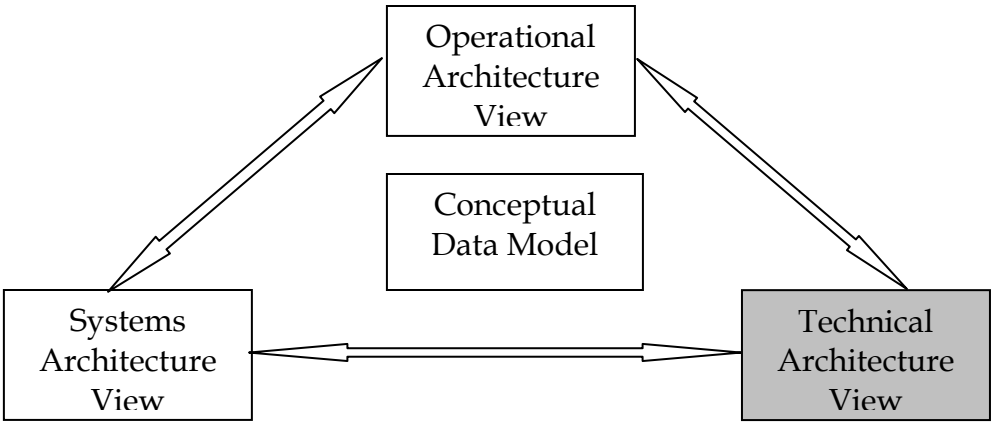
Client and Service Instances invoke other Service Instances using the information from the Service Type Registry.

All OGC Service Instances support a GetCapabilities operation, which returns a capabilities document that describes the capabilities and characteristics of the Service Instance. This document includes information respecting the service supplier, the service's operations, and information about the data content supported by the Service Instance. The Capabilities document provided by the Service Instance contains the same information as obtained for the Service Instance from the Service Instance Registry. The Capabilities document contains a description of the service's operations, the service availability, and information about the content (for data dependent services) provided by the service. The Capabilities Document is used to determine specific service capabilities to which it may respond, or present to a human user for further input or selection.

Note that many services instances will be data dependent either through providing direct access to data or through providing transparent processing services for data of a particular type and geographic region that is typically hidden from their clients. The Timber Values Service example discussed in Section 4.1 might only be available for the Province of British Columbia and that service may not provide access to the underlying forest inventory data on which the valuations are based. The reasons for this situation could be related to performance, security, ownership or commercial issues.

5. TECHNICAL REFERENCE ARCHITECTURE

The technical architecture provides a set of logical system components and their interconnections that is mapped to specific physical systems and software components in the System Architecture. The technical architecture is as implementation independent as possible.



5.1 Introduction

This section presents the Technical Reference Architecture of NFIS. The discussion of this section focuses on the logical structure of the NFIS and does not deal with implementation technologies. For a discussion of the key implementation issues see Section 6.0.

This section makes use of the technical vocabulary introduced in Section 4.0, specifically the logical components of a service-based architecture as shown in Figure 5.1-1.

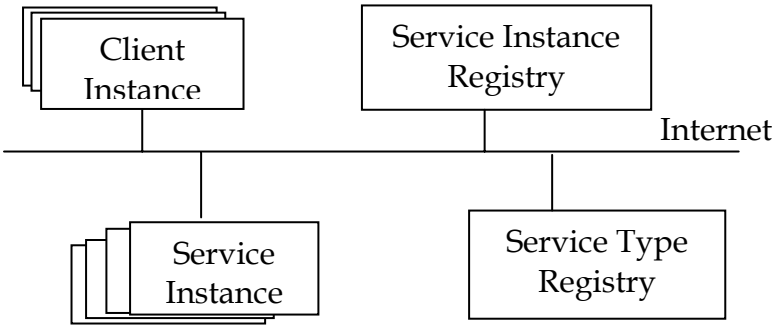


Figure 5.1-1. Logical Components of a Service Architecture

All of the capabilities/functionalities of NFIS reside in the NFIS Service Instances. These instances provide the services used by other service instances or by client instances that interact with people, systems or machines outside of NFIS. Service instances can provide services for:

- Access to forest information including forest inventories, ecological data, forest access roads, general topography and hydrography, and forest dependent species data, etc.
- Computational data processing services such as stand values, computation of habitat suitability, and computation of forest type representation for a given region.
- Reporting services, which may be combined with data processing and computation for certification, tenures, and reports on various types of forest management activities.
- Data presentation services such as generation of graphs, 3D visualization, and animations from user supplied or referenced data.

5.2 Typing Framework for Sustainable Forestry

A key component of any Service Architecture is a typing framework, which provides the conceptual data model. This framework provides the data types needed to define the operation signatures for the NFIS services. At minimum, this framework can use the core types provided by the OGC Geography Markup Language; however, a much richer set of types should be developed specific to SFM. These types can be created as GML Application Schemas.

A typing framework specific to SFM could be expected to include types such as the following:

- Forest Stand
- Forest Tree Species
- Forest Access Road
- Forest-Dependent Animal Species
- Protected Area
- Park (Provincial, National, Municipal)
- Timber Supply Area
- Wood Basket
- Ecological Zone

The development of this typing framework is a major undertaking but is essential to the development of NFIS. The following steps have been taken to initiate the development of the NFIS typing framework:

- Data dictionaries have been established or are being established for the forest and other natural resource sectors by key information holders and providers including governments, industry and other agencies and organizations in Canada.
- International standards have been developed for measuring criteria and indicators of SFM.
- Numerous domestic and international research projects have produced forest and other natural resources data models.

The key issue in the development of the NFIS Conceptual Data Model is to connect this model to the provision of related data and services in a timely manner. The NFIS Conceptual Data Model is essential to the expression of the web-services in support of SFM.

Appendix A provides a few starting points for the development of the NFIS Conceptual Data Model for reporting on SFM.

5.3 Type/Schema Registry

A Type/Schema Registry is necessary for several reasons, including:

- It provides a means of sharing typing information across multiple service instances and service registries.
- It provides a means of discovering type objects and schemas that are shared across information communities (e.g. forestry, environmental protection, etc.). A Data Access Service might, for example, map it's internal data schema to a public schema maintained by the Type/Schema Registry, in order to readily share its data with a broader community.

Creating a GML Application Schema file and hosting it on an NFIS web site could establish a simple typing framework for NFIS. This approach is adequate when the number of types is small. For a slightly larger number of types, a set of application schema files would suffice. For NFIS, it is anticipated that a large number of these types will be required. A type registry can support this requirement. This registry can be a searchable database of GML Application Schemas.

A Type Registry is a service that supports the following sort of operations:

- GetSchema(type_name) – returns the schema fragment for the type name.
- GetParent(type_name) – returns the parent type of the input type.

The World Wide Web Consortium (W3C) or the OGC has not yet defined a suitable Type Registry Service at present, although there has been considerable discussion on this topic over the past year. In the interim, simple Type Registry Services can be constructed using emerging frameworks such as ebXML. Prototype versions of such services are now being constructed as part of the OGC Open Web Services Initiative (OWS) Testbed. For further details on ebXML see (http://www.ebxml.org/).

5.4 Service Type Registry

NFIS Service Type Registries store descriptions of service types. A Service Type as described in Section 4 is a collection of related operations with non-operational service properties. Each operation consists of a set of messages determined by the operation kind. Each message is in turn composed of a set of message parts. The message parts reference element types that effectively determine the operation signature.

A Service Type Registry is a service and hence is also described by its set of operations. These operations are as follows:

- AddServiceType(ServiceTypeDescription)[1]
- GetServiceDescription(ServiceTypeName)
- FindServiceType(FilterExpression)

A ServiceTypeDescription is a document that describes the abstract operations of a Service Type. The GetServiceDescription() operation returns a ServiceTypeDescription for the particular named Service Type. NFIS allows multiple Service Type Registries. Each Service Type Registry has an associated namespace that serves to disambiguate service type names from one another.

5.5 Service Instance Registries

NFIS Service Instance Registries store descriptions of service instances. A Service Instance is a particular implementation of a Service Type, running on a particular physical machine and accessed through a particular message protocol and network address. A given software component may be registered as providing more than one Service Instance as it may support multiple Service Types. For example, the CubeWerx software component CubeServ implements both a Web Map Service and a Web Feature Service.

NFIS Service Instance Registries describe the Service Instances using the following information:

[1] These service operations have not yet been formally defined by the OGC and are part of the current OGC Web Services (OWS) testbed.

- Specific address information – the address of the service port(s).
- Protocol binding. This information describes the mechanism by which service transports information between the client and the service. Possible bindings include W3C Simple Object Access Protocol (SOAP), and Hypertext Transfer Protocol (HTTP) Get/Post.
- Service Type. This type is defined in terms of its operation signatures.
- Service Metadata. This information describes the service semantics and operational characteristics. Each NFIS Service Instance Registry may have its own metadata schema. Metadata schemas, with the operation signatures, describe the semantics of the service. These two items define most of the forestry specific aspects of NFIS services.

The definition of specific NFIS metadata schemas is outside the scope of this report; however, a few simple examples will illustrate the general idea.

A common metadata schema is likely to be defined for all CGDI Services and this schema will likely be a profile or a derivative of an international schema such as ISO 19115. This schema would likely contain the following entries:

- Service name
- Service text description
- Date initiated
- Contact names, addresses, phone numbers etc.
- Region of validity of the service (geographic region)
- Period or times of day when service is available.
- Service Type (based on ISO 19119 or similar service taxonomies)

More specific schemas related to the forest sector might include a Service Subtype (parameter, where the parameter is a refinement of the service taxonomy). This parameter would contain values that conform to an NFIS service type taxonomy. The following list contains examples of these values:

- Operational Forestry Services
    - Inventory
    - Valuation
    - Harvest Planning
    - Road Planning
    - Silvicultural Planning
    - Certification/Certified Areas Registration
    - Forest Fire History
    - License and Tenure
- Ecological Services

- Habitat Suitability
- Habitat Fragmentation
- Representation
- Protected Areas
- Forest Dependent Species

Specific schema would then need to be developed for these service subtypes.

As noted in Section 4, current OGC and other web-service instance provide capabilities or documents that describe the particular service instance. This information can be expected to duplicate that available from a Service Instance Registry, therefore, it is important that these two pieces of information are consistent. The Service Model being developed by the OGC address this consistency requirement by supporting a combined harvest and registration approach. In this approach, the Service Instance Registry is populated by harvesting (calling the GetCapabilities interface) the service instance descriptions from all registered instances. The registration operation is supported on the Service Instance Registry Service Type and all newly created service instances must register with some Service Instance Registry in order to be available for harvesting.

5.6 Supplying or Connecting a Service

Federal, provincial, territorial and municipal governments, private industries and NGOs can provide NFIS Services.

A Service is provided by constructing and deploying a software component that implements the operations for a selected Service Type.

For an existing Service Type, the developer retrieves the Service Type description from the Service Type Registry. This description provides the operation signatures that can be used as the basis for the service implementation. The developer uses this information to build an implementation in a selected implementation technology such as SOAP, HTTP/CGI, or J2EE (see Section 6.0 for discussion of implementation alternatives). This approach is similar to the development of a software subroutine or function, given the description of the subroutines input and output arguments. Software developers determine the internal functionality of the Service Instance and this functionality is transparent to the operation signature.

Once the Service Instance is completed, the service provider makes the Service Instance available by registering the Service Instance with a Service Instance Registry. This procedure involves completing a description document for the service that will include a reference to the Service Type's operation signature, and other parameters specified in the Service Type.

For a new service type, the developer must create the service description and register it with an NFIS Service Type Registry. This procedure involves invoking the AddServiceType(*TypeDescription*) operation on an NFIS Service Type Registry. Since the registration of a Service Type is essentially the definition of a standard, the Service Type Registration may not be an entirely automatic procedure. Some form of review by domain and systems experts may be required before a new Service Type can be registered. Business practices need to be defined and software components must accommodate and support these business practices. Detailed analysis of such business practices is a task under the Operational Architecture activity of NFIS.

Once the new Service Type is registered, the developer can deploy a Service Instance for the new type as for any other Service Type.

5.7 Service Discovery

To discover a service instance, a user invokes a search operation on a Service Instance Registry. Various types of search operations may be supported by different Service Instance Registries including those accepting queries expressed in the OGC Filter Expression (http://www.opengis.org/techno/rfc13info.htm) and the use of the metadata schema attributes of the Service Instance Registry. Such queries would return a list of Service Instances with their Service Types based on the user-specified criteria. The user would then select any service in the list and request a description of the Service Type from the Service Instance Registry. This description provides sufficient information for a software developer to create a client that can interact with the service instance. This case is shown in Figure 5.8-1.

A service provider may be interested in offering an instance of an existing service type or may wish to create a new type of service. In the case of a new service type, the developer would search a Service Type Registry to determine whether a definition already exists for such a service. If so, the Service Type Definition can be downloaded to develop an implementation of the service. This case is illustrated in Figure 5.8-2.

5.8 Service Invocation

To invoke a selected service operation, the service consumer or supporting software developer creates a client using the operation definitions from the service type description. This client is then able to call the operations of the selected service. Many generic clients are likely to be made available as commercial products that are capable of dynamically invoking a range of NFIS services. This concept is illustrated in Figure 5.8-1.

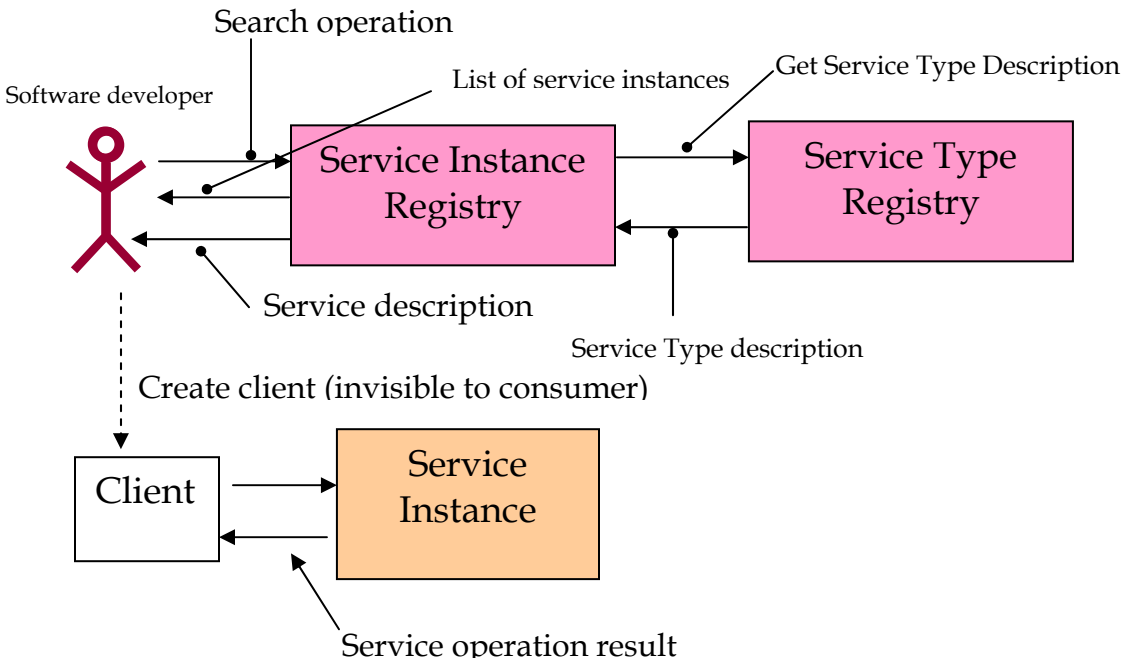


Figure 5.8-1. Invoking a service instance

Note that in the case of Service Invocation, any interaction between a Service Instance Registry and an associated Service Type Registry is invisible to the accessing client and the Service Instance.

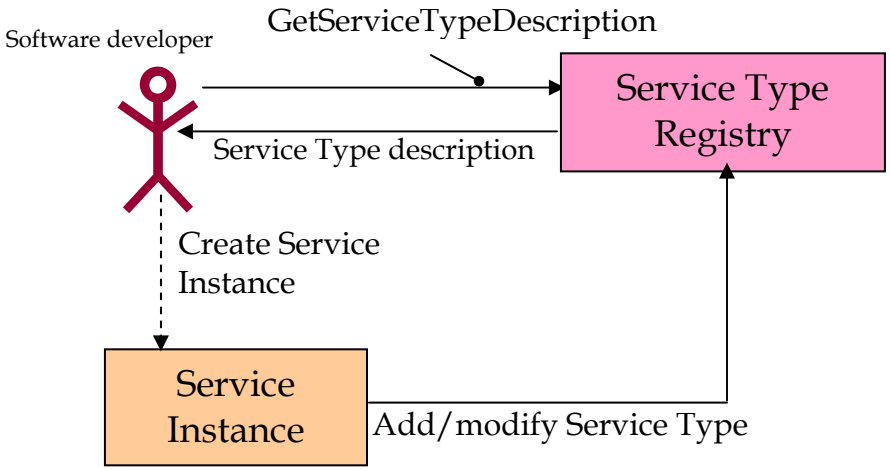


Figure 5.8-2. Creating and registering a new or modified service

5.9 Role of Standards in the Service-Based Architecture

This section summarizes the key standards involved in the Service-Based Architecture outlined in this report.

The overall service architecture is derived from the OGC Service Model, which has been under development over the past two years. Several formal Service Model documents have been issued by the OGC. Early documents centre on the creation of a Basic and General Service Model; however, this approach has been replaced by a single Service Model specification that is currently under development in part through the Open Web Services (OWS) initiative. This specification will provide formal definitions for the Service Instance and Service Type Registry that have been discussed in this section and in Section 4.

The OGC is also developing a number of general geo-spatial Service Types that will be of importance for NFIS. These Service Types include:

- Web Feature Service (See http://www.opengis.org/techno/rfc13info.htm)
- Web Coverage Service (http://www.opengis.org/techno/discussions.htm)
- Web Map Service (Portrayal Service) (http://www.opengis.org/techno/rfc13info.htm)

5.9.1 Web Feature Service

The Web Feature Services is an OGC data access service that provides clients with access to geographic feature information and concrete or abstract geographic objects that typically have geometric location and/or extent. Features in the NFIS community could consist of hydrographic features, topographic features (mountain peaks, valleys etc.), roads and railways, forest access roads, cut blocks, protected areas and parks, and forest cover type polygons. These feature types can have various geometries including points, lines, polygons and various types of multi-geometries.

A Web Feature Server is required to return features in the OGC Geography Markup Language (GML) (http://www.opengis.net/gml/01-029/GML2.html) although proprietary vendor formats can also be supported. All Web Feature Services must be able to provide data in the GML format.

5.9.2 Web Coverage Service

The Web Coverage Service (WCS) is an OGC data access service that provides clients with access to geographic coverage information. An OGC Coverage is a function describing the distribution of some property or properties over a portion of the surface of the earth. Important coverages of interest to NFIS

would include digital elevation models (elevation points or TINS), distribution of soil types (polygons), distribution of forest tree species (aggregated/dissolved forest cover polygons), and climatic data (isolines, polygons). Current Web Coverage Services (WCS) provide only binary data files and support only gridded coverages. More general coverages using the GML 3.0 Coverage Schema are under development.

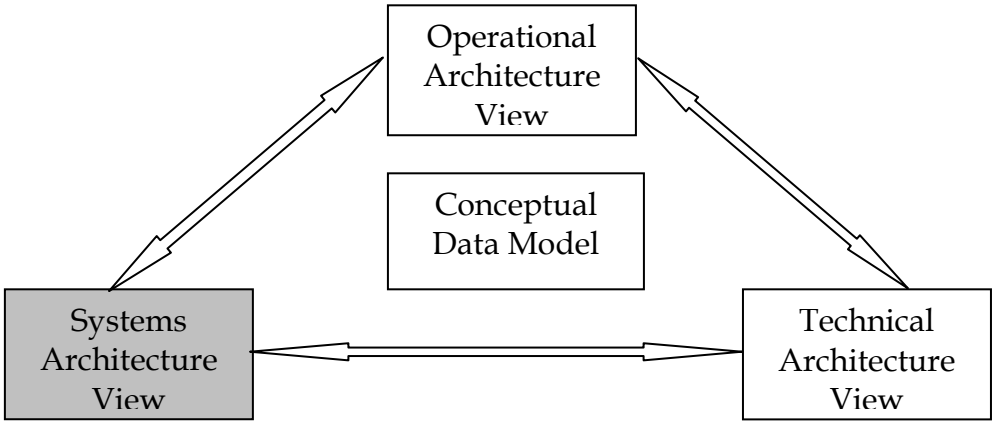
5.9.3 Web Map Service

The Web Map Service (WMS) is an OGC portrayal service that provides graphical presentations of geographic data including both coverages and features. The WMS does not provide data, but rather styled visualizations of data. WMS servers accept a GetMap() request that specifies a geographic area of interest and a list of "layers" that are to be portrayed. The GetMap() request returns an image representing the requested layers in the specified geographic area of interest. Simple feature information queries are supported through WMS. Some WMS implementations (servers), may also implement Web Feature and Web Coverage Service interfaces.

5.9.4 Geography Markup Language

GML is an XML encoding for geographic information including both geometric and non-geometric aspects of the geography. GML provides for the expression of user defined feature types (e.g. roads, rivers, bridges and forest stands) and feature-to-feature relationships. GML is based on W3C's XML Schema, XLink and XPointer and several of its key concepts derived from the W3C Resource Description Framework (RDF). The content of GML 2.x is based on the OGC Simple Feature Model. GML Versions 3.x geometry, topology and temporal content will be based on the newest version of the OGC Abstract Specification, which in turn is based on the appropriate sections of the ISO TC/211 19xxx specifications. GML 3.x will be backwards compatible with GML 2.

6. System Architecture (Implementation)



6.1 Overview

This section looks at the some of the system architecture issues associated with the implementation of the NFIS.

The NFIS Technical Architecture is implementation neutral and can be implemented in a variety of distributed computing platforms (DCP) including HTTP, Microsoft's Distributed Component Object Model (DCOM), the OMG's Common Object Request Broker Architecture (CORBA), Sun's Enterprise Java Beans (J2EE/EJB), Microsoft's .net initiative, and the W3C Simple Object Access Protocol (SOAP). Commercial implementations are likely to favour web-based technologies such as SOAP, .net, HTTP and J2EE.

To handle non-Web-based DCPs, protocol bridges will be required and many of these are currently available as commercial or open source products (See Section 6.3). This protocol bridging is shown graphically in Figure 6.1-1.

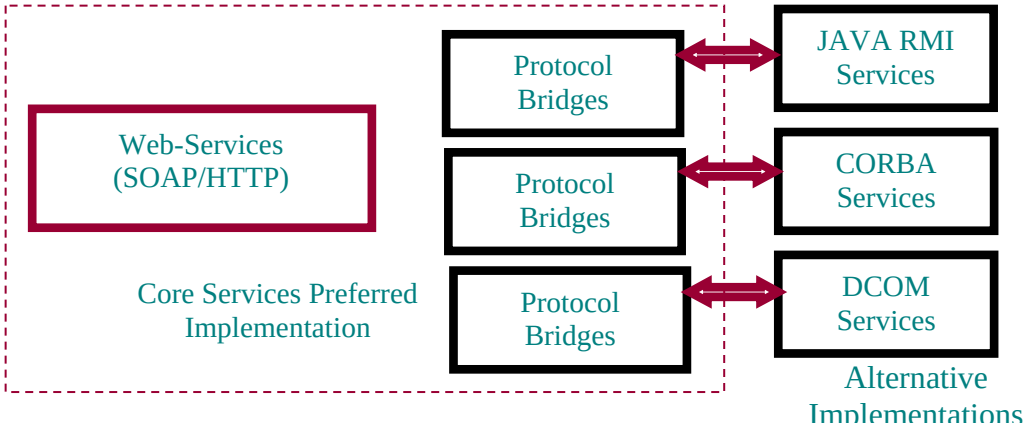


Figure 6.1-1. Commercial Implementations favour SOAP/HTTP with bridges to other protocols.

Commercial development platforms based on SOAP are emerging from a variety of leading software vendors including MicroSoft (considered part of the .net platform), IBM, Sun Microsystems, and Iona. All these vendors have introduced development platforms based on SOAP (and usually Web Services Description Language (WSDL)) in support of web-based services. ([4])

Within the OGC, vendors are developing web-based services such as WCS, WFS, WMS and these will all be based on either conventional HTTP (Servlet/Common Gateway Interface (CGI)) or SOAP implementations.

Given the current direction of the OGC vendors, and the wider world of commercial web-based services, it makes sense to develop the initial NFIS Components (Service Type Registries, Service Instance Registries and Typing Frameworks) in the SOAP/WSDL framework.

This next subsections discuss some of the System Architecture issues with respect to the NFIS Implementation, including:

- Implementation technology alternatives.
- Performance issues
- Authorization and access control
- Integration of legacy systems

6.2 Implementation Technology Alternatives

Implementation of NFIS requires the following general capabilities:

- The ability to discover a service based on service type description and other metadata associated with the service. Users want to find services that perform specific kinds of operations such as providing access to certain types of data, or performing specific kinds of computations while meeting specific quality of service and other criteria including response time, availability, and accuracy.
- The ability to get the formal definition of a service in a sufficient detail to enable developers to implement the service, or build a client to access it.
- The ability to advertise the existence of a service to potential user clients.
- The ability to invoke the service over the Internet.

The technologies to support these requirements are available from a number of distributed computing platforms, including:

- SOAP Web-Services. This DCP uses HTTP or other transport. MS .net, WSDL and Universal Description Discovery and Integration (UDDI) are related technologies.
- HTTP/CGI Web Services. This DCP uses HTTP transport and HTTP/GET, POST operations). WSDL and UDDI are related technologies.
- DCOM (Distributed Component Object Model from Microsoft).
- Common Object Request Broker Architecture (CORBA) from Object Management Group (OMG). Consideration is given to higher-level CORBA services like CORBA Trader and the CORBA Interface Repository.
- JAVA RMI & Java RMI/IIOP. This DCP includes other J2EE Services.

Each of these DCPs provides some or all of the capabilities required for NFIS, and any of them could be used to build the NFIS core infrastructure. While any of these DCPs may be used to build the NFIS Core Infrastructure, discussion will focus on SOAP Web-based services.

The adoption of SOAP-based Web Services also has some negative implications and these must be borne in mind in any future implementations. In general:

- SOAP will use more bandwidth than IIOP (CORBA), RMI/IIOP, RMI, or DCOM.
- SOAP marshalling (construction of data packets from input arguments) and de-marshalling will require more memory and CPU resources than CORBA, RMI/IIOP, RMI or DCOM.

- SOAP will be more complex to implement within any given non-SOAP environment than the native mechanisms (CORBA, J2EE, DCOM), although tools and bridges are emerging rapidly.

SOAP has a significant number of positive implications that provide a strong rationale for favouring SOAP over other DCPs. These reasons include:

- Ease of handling firewalls
- Ease of supporting thin clients.
- Broad industry support.

All of the OGC vendors (ESRI, Intergraph, Galdos, Oracle, Laser-Scan etc) are adopting either the traditional HTTP solution or SOAP over HTTP.

6.2.1 Dealing with Firewalls

As enterprise applications developed in J2EE, CORBA or DCOM moved onto the Internet, the problem of firewall handling became increasingly apparent, and was a significant factor in the subsequent development of SOAP.

In CORBA, Java/RMI, Java/RMI/IIOP, DCOM web-service implementations, it is necessary to get the wire protocol (e.g. IIOP) through at least two firewalls, as shown in Figure 6.2-1. In some cases as many as four firewalls controlled by different organizations would have to be coordinated, a situation that it is very likely to be unsuccessful. Furthermore, a service user must make changes to the security infrastructure in order to access a network service. Finally, in the current, and likely future security climate, it is highly unlikely that any security personnel will be predisposed to open additional ports in their firewalls in order to access a given web service. For all of these reasons SOAP becomes very attractive.

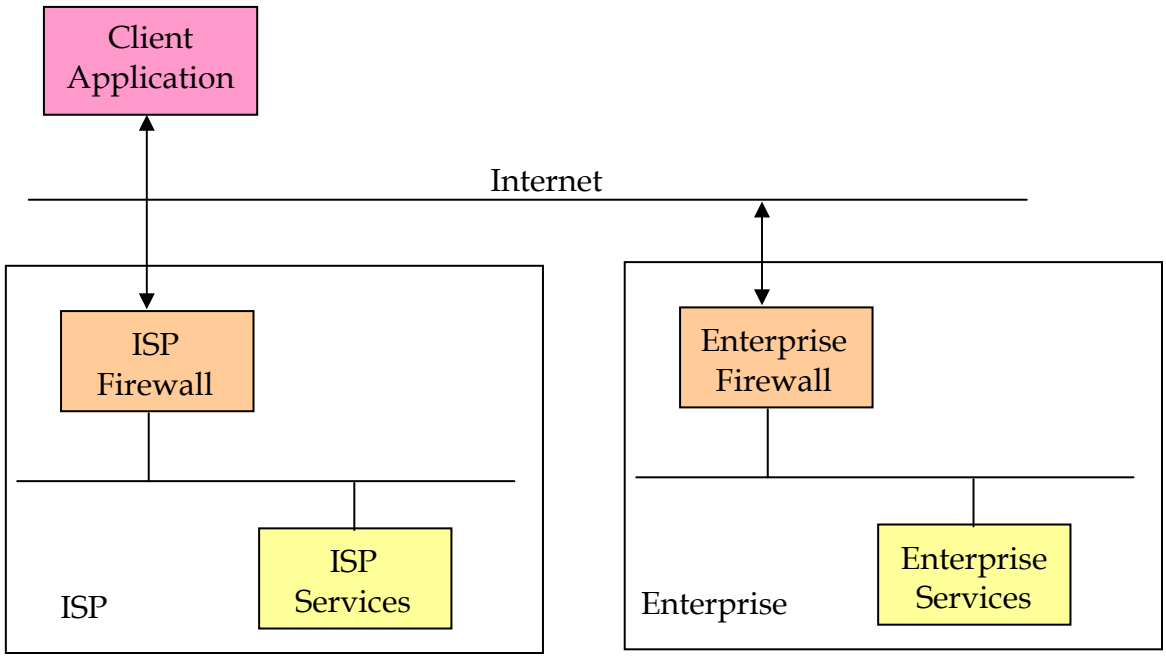


Figure 6.2-1. Accessing an Internet Service means negotiating firewalls

A SOAP implementation alleviates the firewall problem. SOAP can use a variety of transports, particularly HTTP, which is supported on nearly every firewall. This feature means that almost all development environments (e.g. Java JDK 1.0.x) provide all the required support out-of-the-box.

Standard versions of Internet Inter-Orb Protocol (IIOP) or Object Remote Procedure Call (ORPC, a DCOM wire Protocol) are not encrypted and only a few servers that use CORBA, Java 2 Enterprise Edition (J2EE) or DCOM support the encrypted version (e.g. IIOP/Secure Socket Layer (SSL)). Even fewer firewalls are expected to be configurable to enable passing encrypted protocols. On the other hand, HTTPS (HTTP over SSL) is supported by all J2EE servers, and HTTPS can be handled by nearly every firewall. Hence SOAP/HTTP works equally well in both encrypted and non-encrypted environments [7], [8].

6.2.2 Enabling Thin Clients

The support of a client (e.g. Java applet, Java application, Active-X control) that uses J2EE/RMI/IIOP, CORBA (IIOP) or DCOM typically requires a runtime library be bundled with the client side code stubs. These libraries are typically quite large and in some cases can only be pre-installed in the client, for instance, the user native code library for RMI/IIOP cannot be downloaded. A similar situation exists for other DCPs. Note that this situation is not only an issue

impacting the "size" of the client, but also the kinds of clients that can work with the Web service. While servers should support a broad variety of client types, providing this capability will be difficult if we select any of the traditional DCPs. However, SOAP/HTTP enables a wide variety of clients. The SOAP protocol is very simple making it easy to build a SOAP wrapper. Furthermore, there are many commercial application development environments that already provide SOAP support for JAVA and for COM. For example, Microsoft provides SOAP support for Visual Basic (VB) within the MS Office Suite. VB SOAP applications should then be able to access DCOM Servers, .net servers and J2EE servers via SOAP without any problem [13].

6.2.3 Broad Industry Support

SOAP is not a replacement for CORBA, Java/RMI, or even DCOM. These technologies are mature and provide much better performance when operated on a trusted network. These DCPs will continue to have a significant following in all forms of implementations that do not need to cross a firewall boundary. In Web-services, crossing a firewall is mandatory, therefore, we can expect SOAP (SOAP/HTTP in most cases) to become the dominant protocol. This role of SOAP is illustrated in Figure 6.2-2.

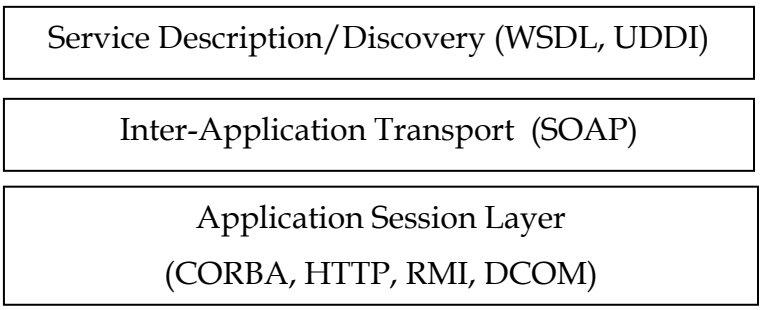


Figure 6.2-2. SOAP as an Inter-Application Transport

Viewed in the context of Figure 6.2-2, SOAP has almost universal vendor support, including Microsoft, Tivoli, IBM, Software AG, Sun Micro Systems, and Oracle.

In the context of OGC, vendors are now moving to deploy Web-based geo-spatial services such as the Web Feature Service, Web Coverage Service and the Web Mapping Service. All of the OGC vendors, who include ESRI, Intergraph, Galdos, Oracle, and Laser-Scan, are adopting either the traditional HTTP solution or SOAP over HTTP.

6.3 Performance Issues

As noted in Section 6.2, the use of SOAP/HTTP or custom HTTP solutions is likely to incur performance penalties relative to solutions using RMI, DCOM or CORBA (non RMI/IIOP) and for some applications these performance issues will be very significant. Unfortunately, there is no information on the comparative performance of SOAP and other DCPs.

While a complete analysis of SOAP performance vs. RMI, DCOM/ORPC is outside the scope of this report, a few remarks can be made that provide an order of magnitude comparison.

A series of tests comparing the performance of SOAP and RMI for the transport of scientific data were conducted at the University of Indiana [6]. These tests involved the round trips (send/response) for arrays and linked lists of various sizes over a network of SUN Sparc and Linux machines.

In general the performance of SOAP was roughly 10 times poorer than SUN's RMI implementation. Three graphs (Figs. 6.3-1, 6.3-2, 6.3-3) from the Indiana University Report show the performance of SOAP vs. SUN RMI as a function of the size of the objects transferred in the operation. Large objects transfer more slowly than do small ones. Where objects are serialized into XML for transport over the wire, there is an additional and significant cost for this serialization.

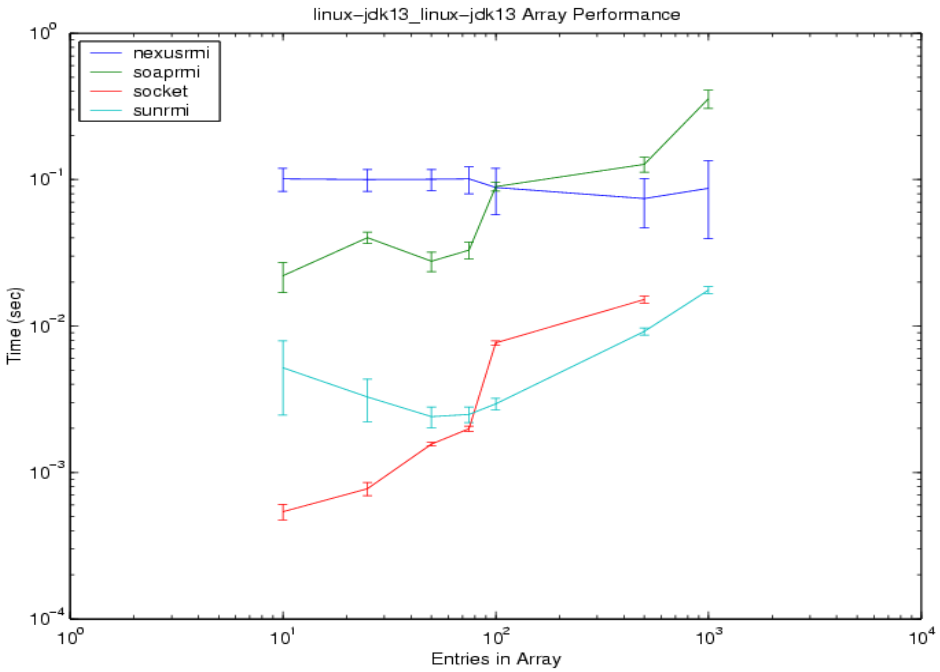


Figure 6.3-1. Round Trip Times for SOAP compared to RMI for various implementations

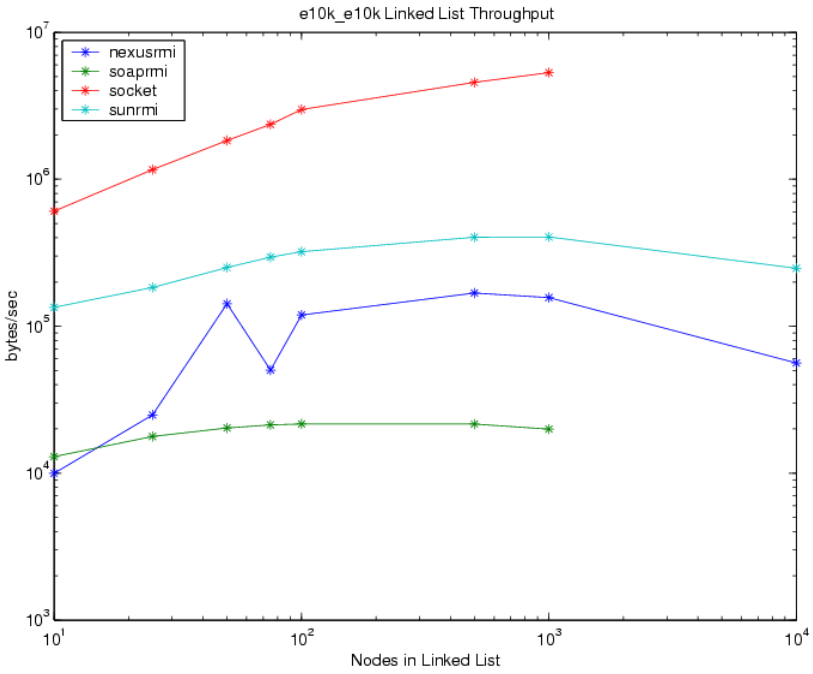


Figure 6.3-2. Throughput for SOAP compared to RMI for various implementations

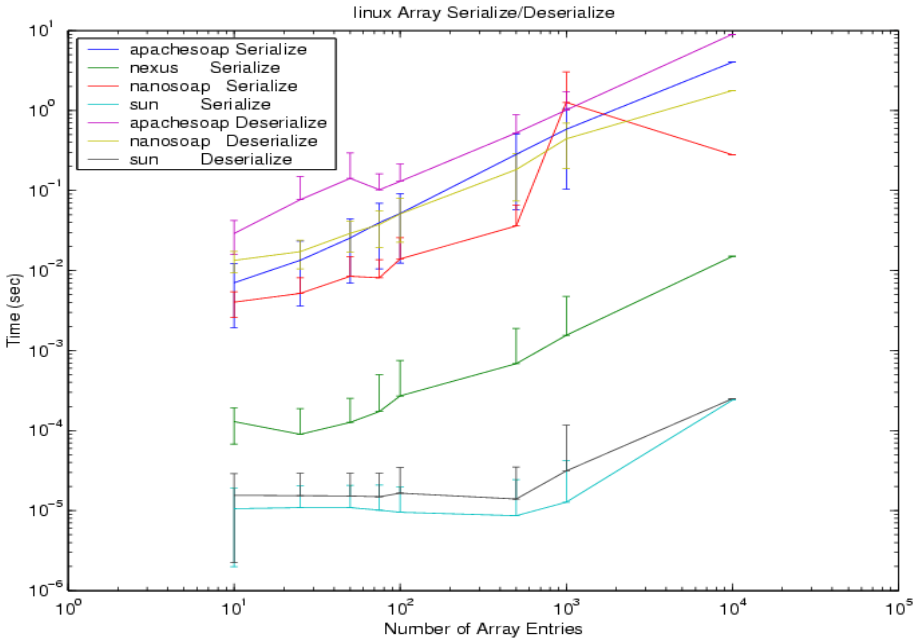


Figure 6.3-3. Serialization/De-serialization Times for SOAP compared to RMI for various implementations

6.4 Security Issues

This section provides a review of some security issues of importance for Web-based service implementations. The section assumes use of SOAP/HTTP as this is the most likely implementation for NFIS web-based services.

Using an HTTP transport, SOAP can carry messages over the Internet and invoke data access or data processing operations behind the firewall of a supporting organization. Often, these processes have been incorrectly viewed as bypassing firewalls altogether. SOAP is another payload that can be carried via HTTP, so the potential vulnerabilities are similar as with using HTTP alone. Moreover, since SOAP packets declare their "intent" by publishing interface and method names in the HTTP header, it is possible for firewalls to perform filtering based on this information.

The SOAP specification requires that implementations verify that the information in the HTTP header matches the corresponding headers and tags in the SOAP payload, otherwise the SOAP call should be rejected. SOAP enables the implementation of Message Layer Security. Formal proposals for Message Layer Security have been defined in IBM's Soap Security Extensions that make use of XML Signatures.

Since SOAP runs over HTTP, standard authentication mechanisms that are HTTP-friendly can be used with SOAP. These protocols can authenticate the server (and optionally the client), and can provide a confidential channel over which SOAP payloads can travel.

As has been shown by Hada [7], SOAP over SSL can be combined with XML Digital Signatures to enable non-repudiation of requests.

6.5 Integration of Legacy Systems

In this document, all system implementations that are not based on SOAP are viewed as "legacy systems". This view does not imply that SOAP will now or in the future replace any of these legacy systems.

For the purposes of this discussion, legacy systems are categorized as follows:

- Non-client server systems. These systems provide data access or processing functionality based on a local API that is called by the user's application program running on the same machine.
- Client-Server or peer-to-peer systems based on CORBA (non-Java).
- Systems based on Microsoft Distributed Network architecture (DNA) (DCOM, ASP, MTS)

- Systems based on J2EE
- Systems based on HTTP/CGI.

Each of these categories of systems can be readily integrated under SOAP as described in the following subsections.

6.5.1 Non-client server systems

To migrate a non-client server system, the developer expresses the existing API calls as SOAP messages. The SOAP layer then plays the role of message receiver and local dispatcher as shown in Figure 6.5-1.

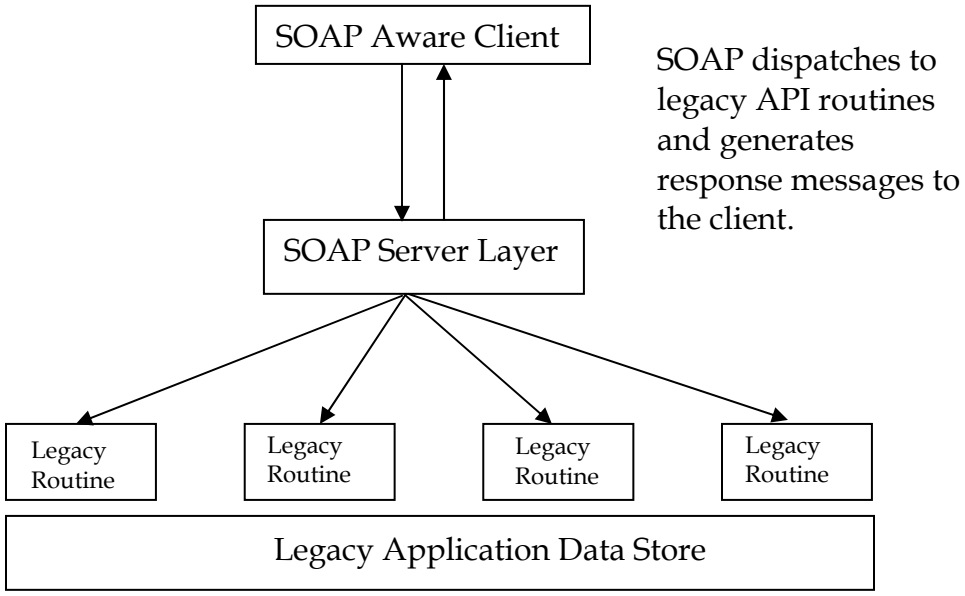


Figure 6.5-1. SOAP over legacy non-client-server application

A variety of SOAP Server tool kits can be used to implement a system as shown in Figure 6.5-1.

6.5.2 CORBA Based Systems

CORBA Systems provide complete location and language transparency. Using compiled interface definitions an object method on one machine can be called by an application or an object method on any other machine on a network. Objects in the CORBA system communicate with one another using a binary wire protocol called IIOP.

To introduce SOAP into the CORBA domain, one requires a means of calling CORBA object methods from SOAP and a means of generated SOAP messages from CORBA calls. This requirement can be accommodated in a general fashion by using a SOAP-CORBA bridge such as the one provided by the open source project from LifeLine Networks (http://soap2corba.sourceforge.net/html/). This mechanism is illustrated in Figure 6.5-2 for the case of calling a CORBA object method from a SOAP request.

The SOAP message is parsed and processed for comparison against the CORBA Interface Repository by the SOAP Message Processor, which is part of the SOAP to CORBA bridge. If a match is found, a dynamic CORBA request is constructed and passed to the CORBA ORB (Object Request Broker) that in turn dispatches to the actual implementation object.

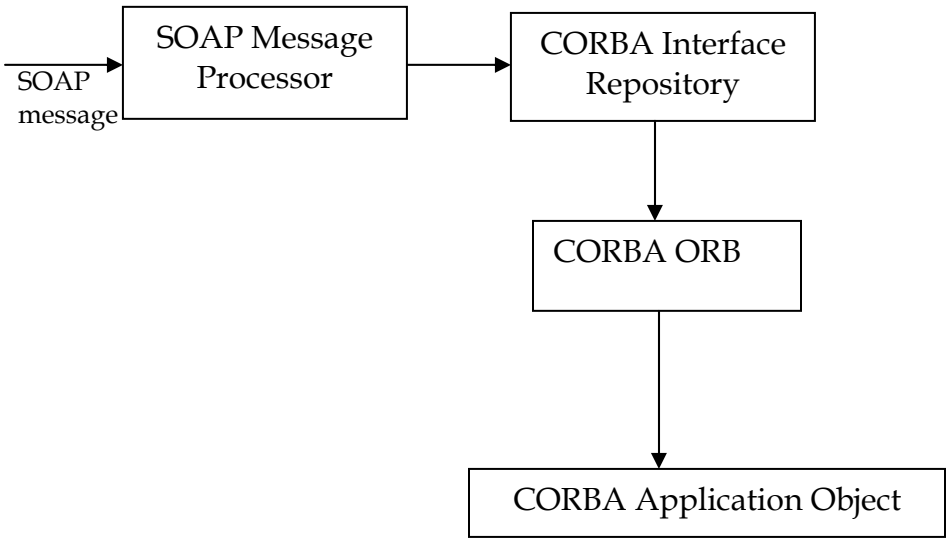


Figure 6.5-2 Calling CORBA Objects Methods from SOAP

For further details the reader is referred to the Lifeline web site [11]. A number of commercial products (e.g. Iona XML Bus (http://www.iona.com/products/xmlbus_home.htm)) also provide generic bridges between CORBA and SOAP [12].

Specific solutions can be obtained using a variant of Figure 6.5-2. In this case the CORBA System is assumed to be complete and no new methods are to be added after integration into NFIS. The problem then becomes one of calling CORBA object methods from SOAP messages where the SOAP messages can be statically mapped to the CORBA interfaces as illustrated in Figure 6.5-3.

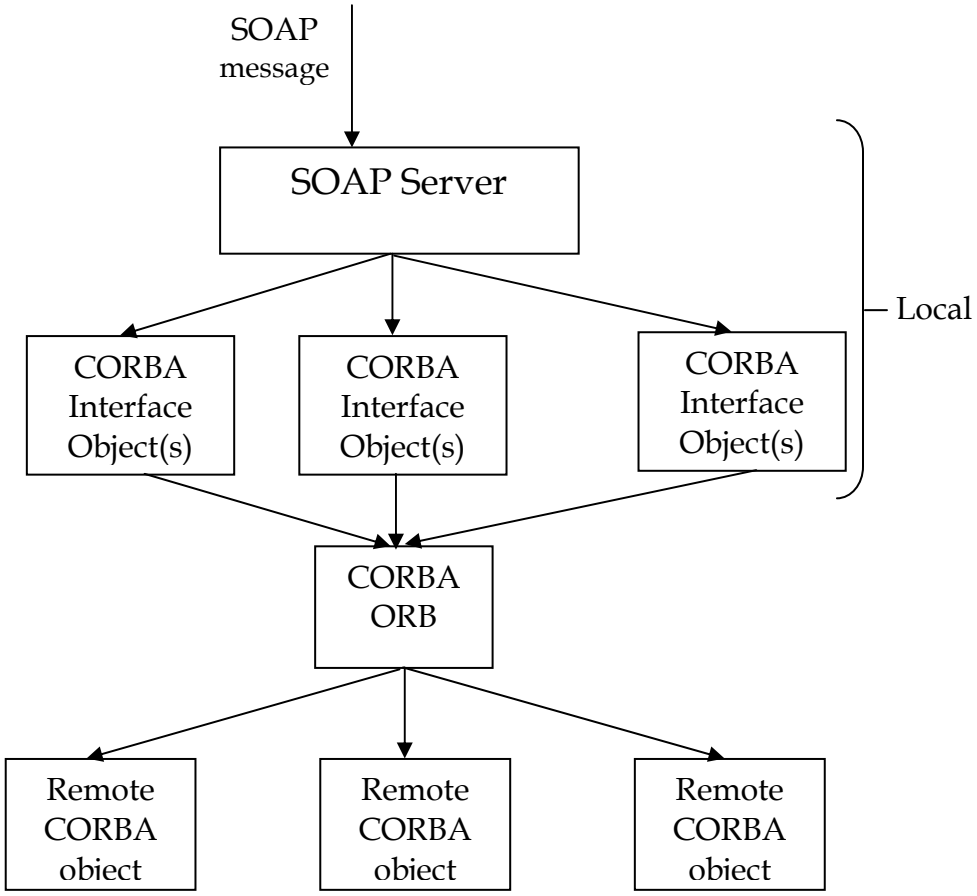


Figure 6.5-3. Static Interface Method – SOAP to CORBA

A CORBA object or set of objects acts as the client of the CORBA service to be integrated into NFIS. A SOAP Server then dispatches to the appropriate object methods based on the incoming SOAP message. The object method then calls the appropriate remote object methods using the CORBA mechanisms. The response is generated in a similar manner.

Note that this method is much simpler than the implementation of a general SOAP to CORBA bridge and in some cases, it will be simpler than installing and configuring the bridge; however, this method is not a general solution. If new CORBA services are added, then this second method will require that a new interface object be written and integrated.

6.5.3 Microsoft DNA Based Systems

With the introduction of the SOAP Toolkit V2.0, Microsoft has begun to provide serious tools in support of traditional DNA (Distributed Network architecture) (VB, COM) applications. In effect the SOAP toolkit provides a set of COM

objects to interact with SOAP. These objects provide interfaces that translate COM object method calls into SOAP requests and vice versa as shown in Figures 6.5-4 and 6.5-5 taken from the SOAP toolkit documentation [13].

For static legacy systems, one can also approach the COM integration problem in a manner similar to that shown for CORBA in Figure 6.5-3 with about the same advantages (likely better performance) and shortcomings (less flexible and requires new code to be written when COM service extended).

Note that unlike the SOAP2CORBA bridge described above, the Microsoft SOAP Toolkit V2.0 relies on WSDL for interface description.

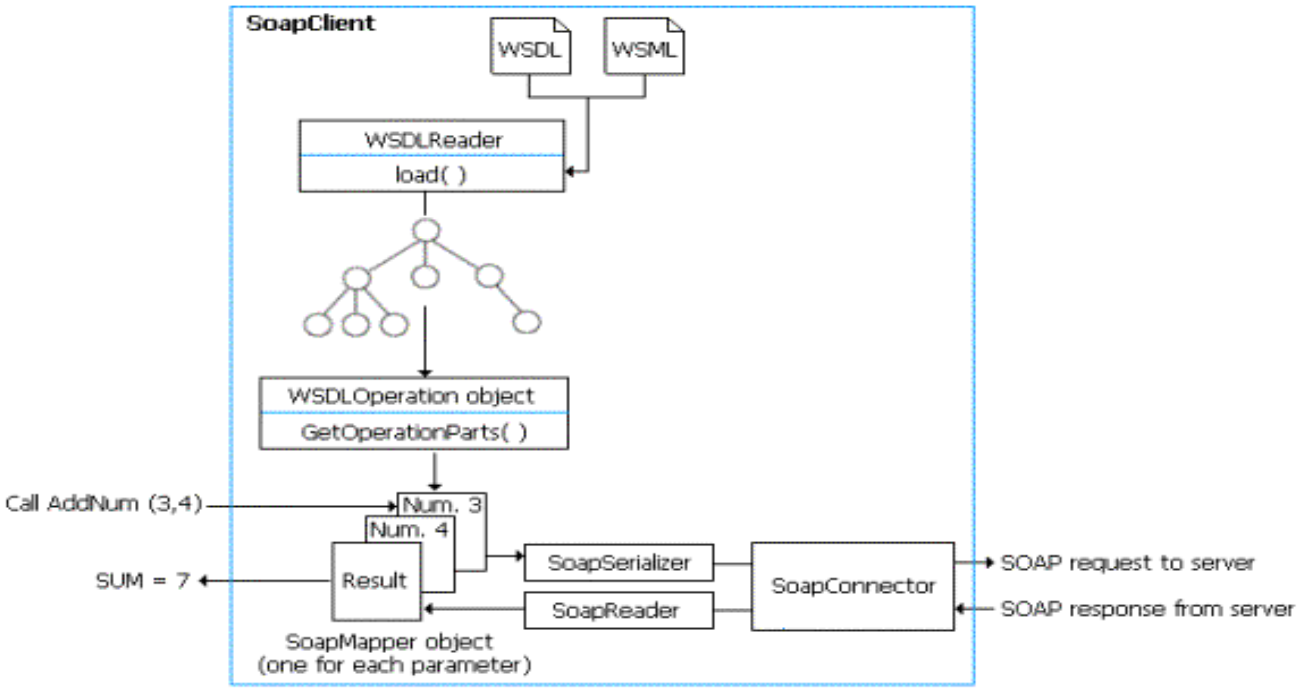


Figure 6.5-4. SOAP Toolkit client side model

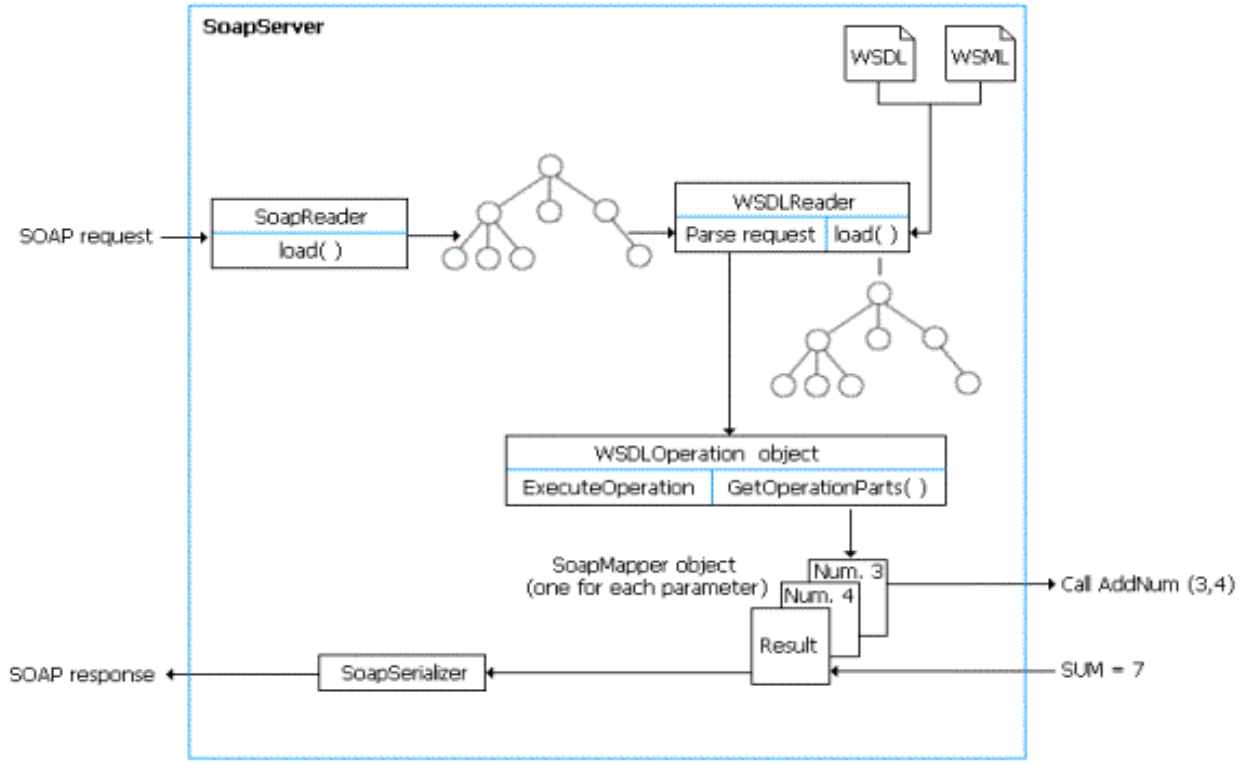


Figure 6.5-5 SOAP Toolkit server side model

6.5.4 J2EE Based Systems

Outside of the Microsoft DNA/.net domain, the dominant DCP is J2EE (Java 2 Enterprise Edition). An overview of the J2EE Architecture is shown in Figure 6.5-6 [5],[9],[10].

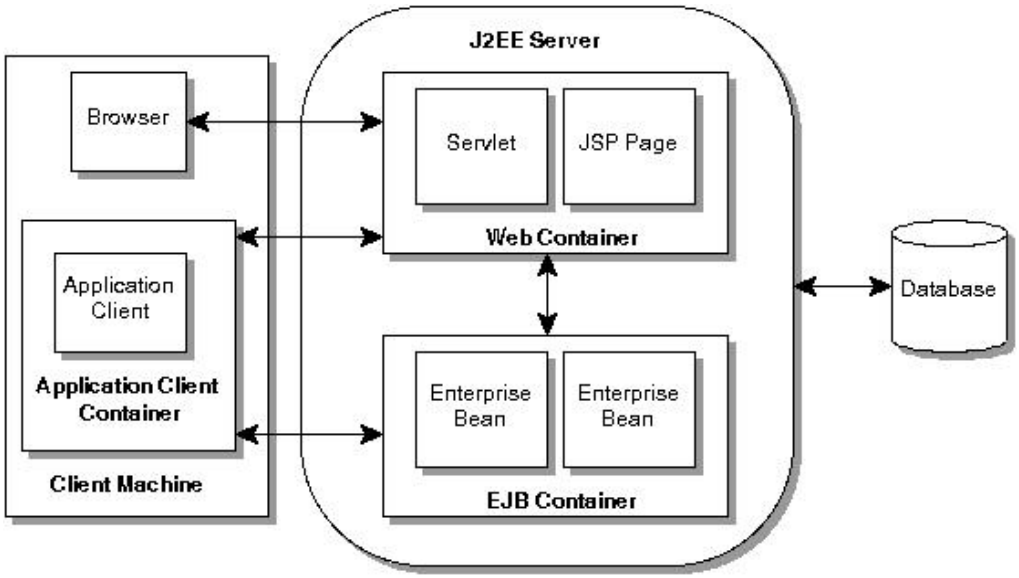


Figure 6.5-6. J2EE Architecture Overview – Container Viewpoint

J2EE provides an environment supporting distributed transactional access to a set of persistent Enterprise Java Beans in a wide area network. Like their DNA(.net) counterparts, J2EE platforms provide a complete set of "plumbing" components including services for authentication, access control, object access serialization, clustering, and dynamic connection pooling.

Recent versions of J2EE provide specific API support for XML (JAX*) and in particular for web-services. These include:

- JAXR – Jave API for Registries. This API will provide UDDI functionality on J2EE and will support the typical querying, editing and publishing tasks that registries require.
- JWSDL – This API is still in the offing but it will soon make it easier to create and edit WSDL service descriptions. Note that IBM has released its own version called WSDL4J.
- JAX/RPC – This API provides support for the sending and receiving of SOAP requests from JAVA applications and includes support for marshalling and un-marshalling. This standard Java API is expected to replace the several vendor specific API's for SOAP access that are currently on the market.

After a slow start, it is clear that SOAP support within J2EE is building quickly. SOAP in the context of J2EE is shown in Figure 6.5-7.

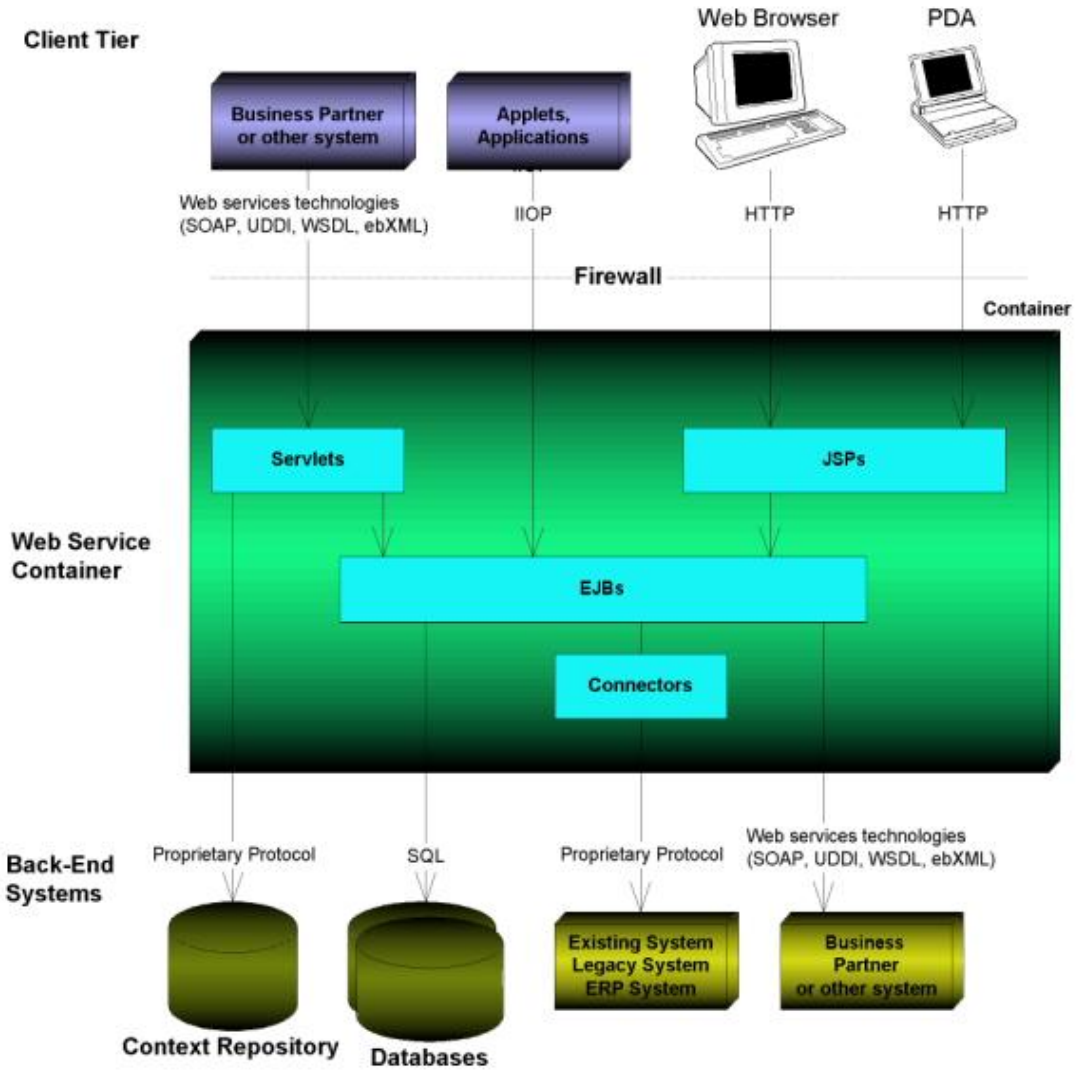


Figure 6.5-7. Integrating SOAP into the J2EE World

6.6 Workshop Use Cases and Potential NFIS Services

In the course of the three workshops held during the NFIS Technical Vision & Reference Architecture study, a number of use cases that touched on possible NFIS services were discussed. A selected subset of these use cases is shown here and interpreted as prototypical NFIS services.

6.6.1 Certification Reporting Service

The Certification Reporting Service (Figure 6.6.1) extracts certification information (geographic extent of each certified area, level of certification, tenure unit to which the certification applies, and organization name) that is then locally styled for presentation or sent to a separate Presentation Service. Output of the presentation component of the service may be a map showing the certified areas with a colour code for each certification type or level. A tabular presentation may be generated from the same data showing the certification body, certification achieved, date of certification, area certified, percentage of tenures or percentage of land area certified and the allowable annual cut. Such a service would need to be able to take into account land areas that are not candidates for certification, such as private lands, protected areas and land that is not in the forest management zone.

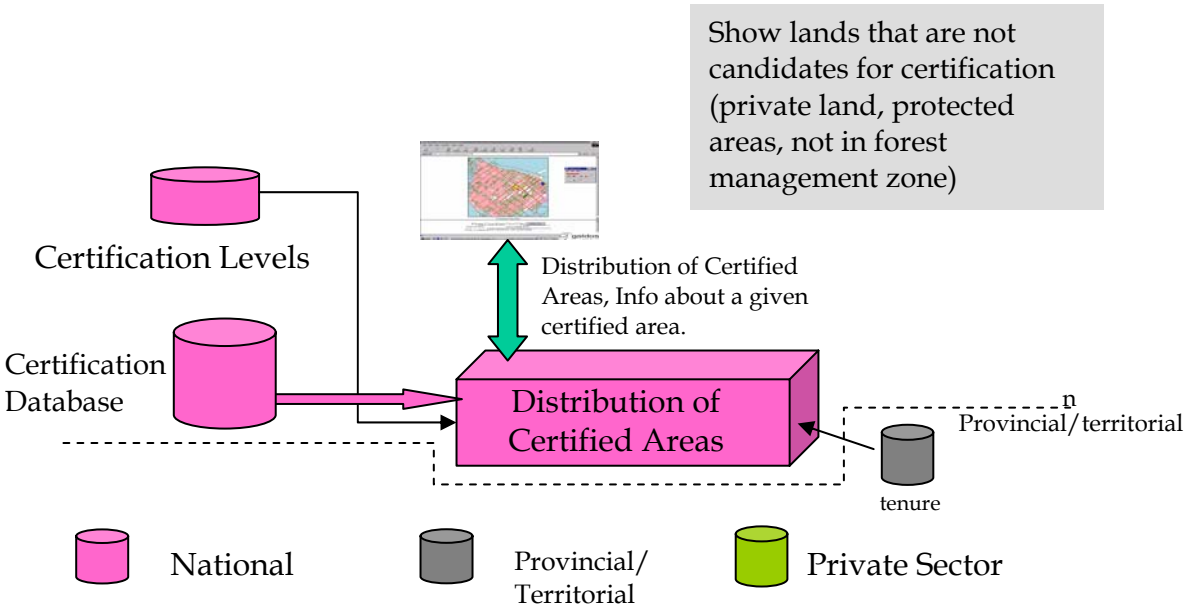


Figure 6.6-1. Certification Reporting Service

6.6.2 Reporting on Protected Areas

A Protected Area Reporting Service is a data access service that obtains information from various provincial/territorial and local databases and compiles a national picture of protected areas in accordance with an agreed to national Protected Areas Schema. This service is illustrated in Figure 6.6-2.

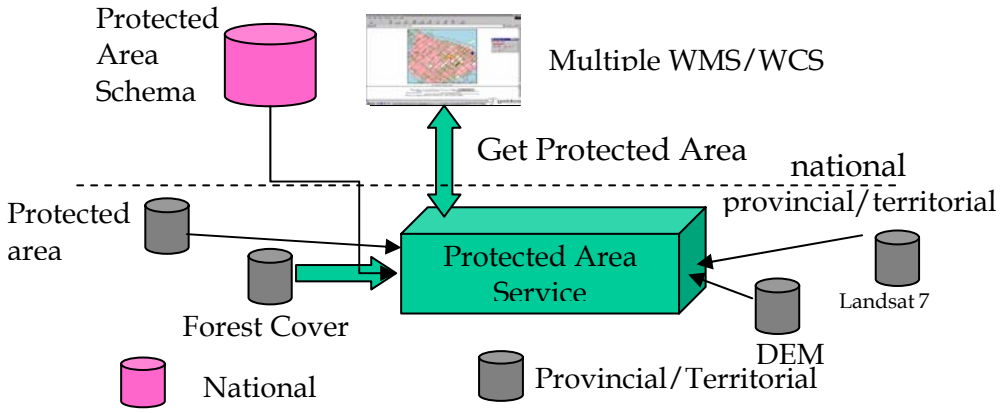


Figure 6.6-2. Protected Area Reporting Service

This service may incorporate or make use of a separate presentation service that could show a map of the distribution of protected areas. As in the Certification Reporting Service, tabular reports showing percentage of land base (by various measures) in protected areas could also be provided. A Protected Areas Registration Service may be used to register protected areas as shown in Figure 6.6-3.

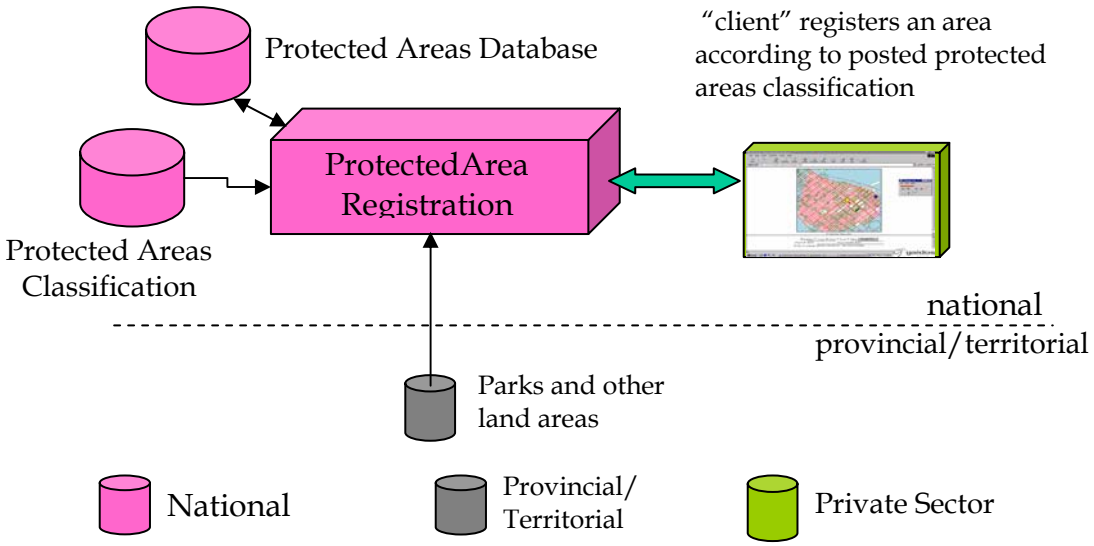


Figure 6.6.3. Protected Area Registration Service

6.6.3 Forest Dependent Species Services

These services provide information on the status of forest dependent species and their habitat and would likely be implemented as a group of services. The general idea of these services is illustrated in Figure 6.6-4.

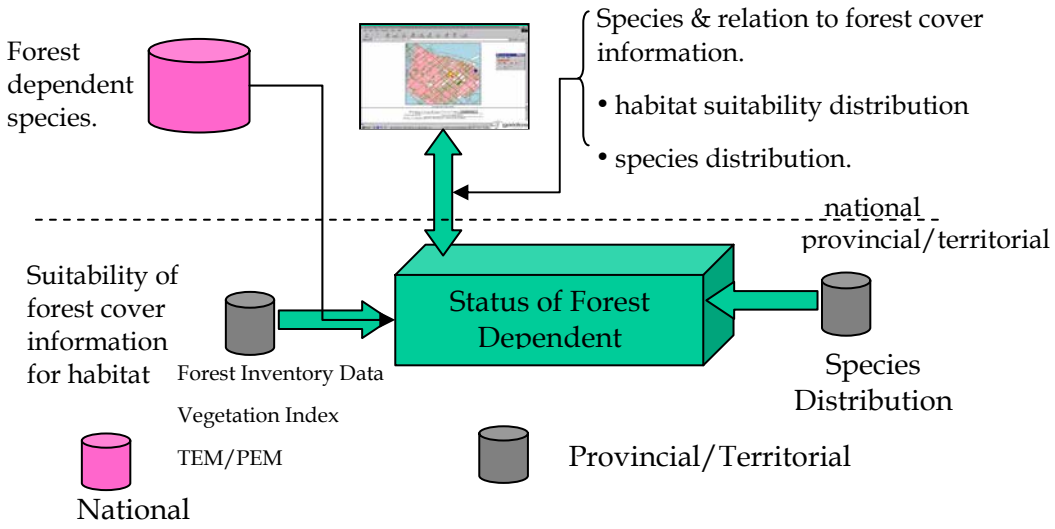


Figure 6.6-4. Status of Forest Dependent Species

This service would combine forest cover and other ecological and climatic information to determine habitat suitability, and likely requiring different measures in different parts of the country. The service would also enable the comparison of habitat information with population estimates by species.

This type of service could be used to determine the current status of forest dependent species, and to assess the possible impact of forest management activities.

6.6.4 Regeneration Status

This service is another querying/report generation service, which provides data on the regeneration status of tenures that are subject to active harvesting within the forest management zone. This service would integrate regeneration information provided at the provincial/territorial level to provide a national view of regeneration status for any selected area in the country. The service is shown graphically in Figure 6.6-5.

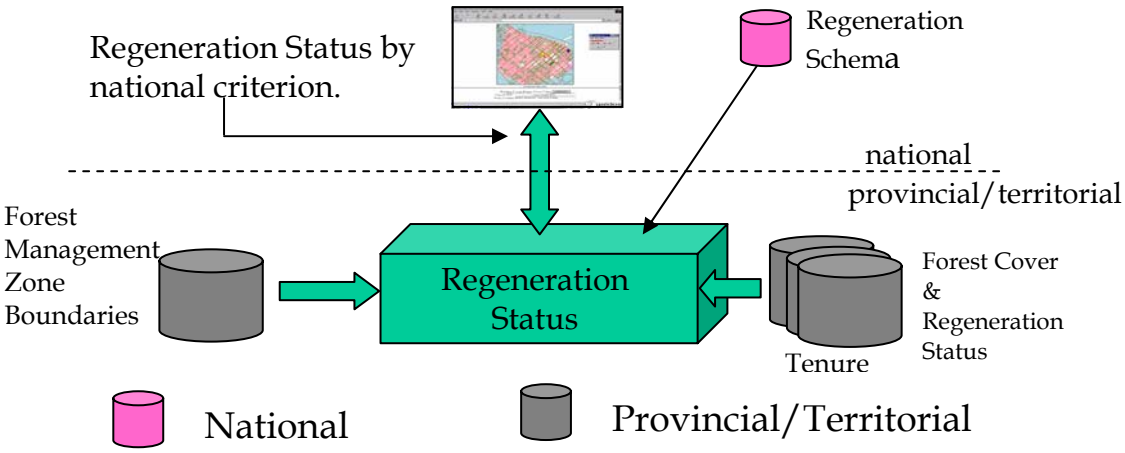


Figure 6.6-5. Regeneration Status Reporting Service

This service may incorporate or make use of a separate presentation service in order to provide a map showing the distribution of regeneration levels. In the same manner as for the Certification Reporting Service, tabular reports showing the percentage of the land base (by various measures) in various states of regeneration could be provided. A similar service to the Protected Areas Registration Service might be used to update the regeneration status of a forest tenure.

REFERENCES

[1] "A Technical Vision for the Canadian Council of Forest Ministers, National Forest Information System", First Mark/Swiftsure/Galdos NFIS TVRA Team. Draft, September 2001

[2] "C4ISR Framework" (http://www.fas.org/irp/program/core/c4isr.htm)

[3] "Service Model Concepts", Galdos Technical Report, August 2001 (Technical Report # TN2001-01)

[4] "SOAP Overview", Tom Clemens, Sun Micro Systems Best Practices, Web Services, Technical Overviews (http://dcb.sun.com/practices/webservices/overviews/overview_soap.jsp)

[5] "J2EE Overview" (http://java.sun.com/j2ee/overview.html)

[6] "Requirements for and Evaluation of RMI Protocols for Scientific Computing", Madhusudhan Govindaraju, Aleksander Slominski et al. (http://www.extreme.indiana.edu/soap/sc00/paper/index.html)

[7] "SOAP Security Extensions", Satoshi Hada, IBM Tokyo Research Laboratories (http://www.trl.ibm.com/projects/xml/soap/wp/wp.html)

[8] "SOAP Security Extensions: Digital Signature" (http://www.w3.org/TR/SOAP-dsig/)

[9] "Developer's Guide to Building XML-base Web Services with J2EE", James Kao, June 2001 (http://www.theserverside.com/resources/article.jsp?l=WebServices-Dev-Guide)

[10] J2EE Downloads and Specifications (http://java.sun.com/j2ee/download.html).

[11] SOAP2CORBA and CORBA2SOAP (http://soap2corba.sourceforge.net/)

[12] CORBA/COM & SOAP Interoperability at IONA (http://www.iona.com/pressroom/2000/20000605.htm)

[13] "Microsoft updates SOAP toolkit, further supports Web services standards", Tom Sullivan (http://www.infoworld.com/articles/hn/xml/01/04/11/010411hnsoaptoolkit2.xml)

APPENDIX A - CONCEPTUAL DATA MODEL

This appendix provides a few resources for the development of the NFIS conceptual data model(s).

- **Central European Environmental Data Request Facility** (http://www.cedar.at/terminology/env-term/datamodel_txt.html)
- **The Sustainability Report** (http://www.sustreport.org/resource/reports.html)
- **Sustainability Development Information (SDInfo)** (http://www.sdinfo.gc.ca/ENG/Default.cfm)
- **State of Canada's Environment Infobase** http://www.ec.gc.ca/soer-ree/)
- **Centre for International Forestry Research** (http://www.cifor.cgiar.org/acm/methods/toolbox1.html)
- **The International Institute for Sustainable Development** (http://iisd.ca/natres/forests/)
- **The State of Canada's Forests** **(**http://www.NRCan.gc.ca/cfs/proj/ppiab/sof/oldsof95/toceng.htm)
- **Integrated Application of Sustainable Forest Management Practices** (http://www.NRCan.gc.ca/cfs/kochi/papers/planning_e.html)
- **Criterion and Indicators of sustainable forest management in Canada** (http://www.NRCan.gc.ca/cfs/proj/ppiab/ci/cr3p_e.html)
- **Glossary of Forestry Terms (Canadian Forest Service)** (http://www.nrcan-rncan.gc.ca/cfs-scf/science/prodserv/glossary_e.html)
- **Fire Weather Definitions** (U.S. National Fire Weather Service) http://205.156.54.206/er/gyx/firewx_definitions.html)