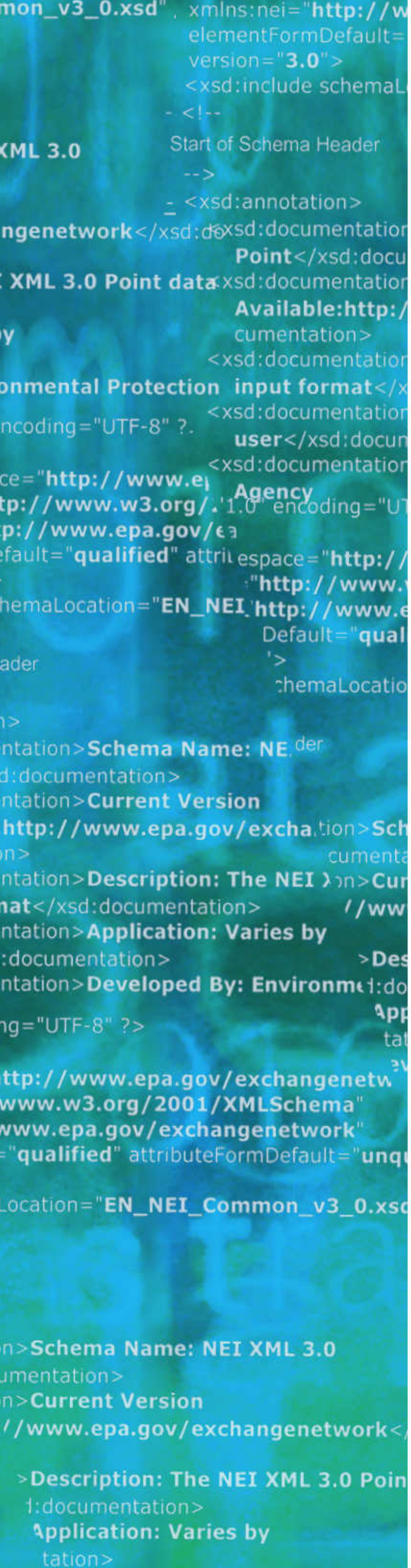

Exchange Network Discovery Services

Flow Configuration Document

Version: 2.0

Revision Date: April 6, 2009

Environmental Information eXchange Network

Table of Contents

2.1 Background 4
2.2 How to use this FCD 5
3.1 Flow Component Version History 6
3.2 Flow Component Versions Currently Supported 6
3.3 Flow Summary Information 7
3.3 Flow Summary Information 7
3.4 Flow Identification 7
3.5 Data Flow Overview 7
3.6 Flow Access and Security 7
3.7 Flow Administration 8
3.8 Additional Flow Tools and Resources 8
3.8.1 Web Interface 8
4.1 Schema Structure 8
4.1.1 Node Service Description Language 8
4.1.2 Data Element Description Language 8
4.2 Schema Components 9
4.2.1 Node Service Description Language 9
4.3 Retrieving Granular Data from NSDL and DEDL Schema 12
5.1 Data Services 14
5.2 Node Services 14
5.2.1 GetServices 14
5.2.2 GetServiceDetails 14
5.3 ENDS Services 15
5.3.1 GetNodeServiceList 15
5.3.2 Service Refresh 16

1 Acknowledgements

This document was prepared with input and support from the following individuals:

Glen Carr	OR DEQ
Yunhao Zhang	EPA CDX
Doug Timms	Enfotech
Chris Clark	EPA CDX
Mike Reich	Ross & Associates

2 Introduction

2.1 Background

This document describes the purpose and functionality of the Exchange Network Discovery Services. ENDS is composed of two main components: first, a set of services allowing EN partners to submit and query the service descriptions stored in the ENDS repository; second, a Node Service Description Language that describes the service descriptions in a machine readable format.

The Node Functional Specification version 2.0 lays out a powerful and generic framework for publishing different Exchange Network services. However, there is the need for a common service description repository because the Web Service Description Language (WSDL) file alone is not sufficient to describe Exchange Network services as defined in Flow Configuration Documents.

The services described below all accept or return a common XML schema: the Node Service Description Language (NSDL). This XML schema provides a structured, standard way to represent EN services across all EN Nodes. This document also defines a Data Element Description Language (DEDL) that can be used by individual Exchange Network nodes for describing acceptable parameters and valid allowable values, enhancing the ability for EN partners to build rich, client-ready applications that consumer EN services.

Functionally, this document describes two sets of services: the first are two services that are implemented on individual EN nodes; the second are services implemented and hosted centrally by a special Exchange Network node – the Exchange Network Discovery Services (ENDS) v2.0.

ENDS Design Overview
With the adoption of the Node Functional Specification version 2.0, the Exchange Network has a new opportunity to create an automated system of service description, cataloging and discovery. The Node Functional Specification defines a core primitive method called GetServices that all nodes must implement. This service is designed to return an XML schema describing every Query, Solicit, Execute, and/or Submit service that the node offers. The ENDS 2.0 Node has been architected to 'poll' the GetServices method on every Exchange Network node to collect and catalog the responses in a central service repository available to all EN Partners. This central repository will allow true browsing of the Exchange Network, and is a vital part of data publishing as it will enable a single Node Client to consume all services published by network nodes and dynamically bind to service requests at run-time.

The process for a client to use the new ENDS is described below:

1. The client retrieves all relevant node descriptions from the ENDS node in one call and caches it locally. The returned document contains complete descriptions about nodes, services and parameters.
2. The client can use very simple XPath to extract service names, node names, and other information from the local node description file.

3. Based on the information stored in the cached result, the client calls the requested service on the specified node.
4. The cached descriptions may be used for a longer period time or be refreshed as needed.

The “get-once and use multiple times approach” simplifies the client business logic and dramatically cuts down development and maintenance costs of ENDS 2.0.

2.2 How to use this FCD

Section 4 describes the implementation and use of the Node Service Description Language, as well as additional information on XPath queries for isolating granular information from the general ENDS return schema. Section 5.2 describes the mandatory GetServices method that all Exchange Network nodes must implement (Section 5.2.1), as well as an optional service for describing service parameters and allowable values (Section 5.2.2). Section 5.3 describes the services that can be used to query the ENDS registry (Section 5.3.1), and that can be used by Node administrators to update or replace records in the ENDS registry (Sections 5.3.2).

3 Component Alignment and Change History

This FCD requires the Node Service Description Language (NSDL) version 2.0 schema and the Data Element Description Language (DEDL) version 2.0 schema.

3.1 Flow Component Version History

Component	Version	Date	Changed By	Description of Change
FCD	1.3	January 30, 2006	Glen Carr	Final 1.0 version
FCD	2.0	July 30, 2008	Yunhao Zhang	Initial 2.0 version

3.2 Flow Component Versions Currently Supported

Component	Version(s) Supported	Explanation (optional)
FCD	2.0	
NSDL Schema	2.0	Node Service Description Language schema

3.3

Flow Summary Information

3.4 Flow Identification

Flow Name: ENDS_v20, DEDL_v20

Flow Description: The ENDS v20 services provide basic discovery capabilities for Exchange Network Nodes. This document outlines the services implemented at the ENDS repository, in addition the schema used to return ENDS records and to return Node GetServices response documents.

The DEDL v20 services provide a mechanism for publishing meta-information associated with data elements, such as data type, length and format. The data element description can be retrieved and used by applications to create feature rich user interfaces or validating user inputs.

Flow Steward: Exchange Network Operations Board

Flow Steward Contact Information: kurtr@sso.org

3.5 Data Flow Overview

This is the dataflow for publishing network node services to the central service repository. There are two ways node services can be published:

Publishing through the GetServices method
Publishing in the Exchange Network Discovery Services v2.0 (ENDS 2.0)

The GetServices method is one of the web methods defined in the Node Functional Specification v2.0. It is basic service publishing mechanism that should be supported by all network nodes.

The ENDS 2.0 is a network-wide service repository which contains service descriptions for all nodes. ENDS 2.0 offer not only a set of service publishing services, but also provide service management capabilities.

For applications that interact with only a single network node at a time, the GetServices method should be sufficient and it is more efficient. On the other hand, for applications that interact with many network nodes, the only way to get the complete service list is through the ENDS 2.0.

3.6 Flow Access and Security

ENDS 2.0 uses the Network Authentication and Authorization Services (NAAS) to authenticate and authorize users. A user must have a valid NAAS account in order to access ENDS 2.0 services.

Only node administrators are allowed to submit node service description documents to the dataflow, and the administrator must be the owner of the node the document describes. Services published to ENDS 2.0 are visible to all Exchange Network users by default. Since the purpose of ENDS is discovery, services which must not be discoverable by any network partner should not be included in the node's NSDL file.

It should be emphasized that being able to see services offered by a node doesn't means that the services are accessible. The service owner can still control who can access what by imposing additional policies.

3.7 Flow Administration

Because the ENDS 2.0 node will automatically populate the ENDS registry with entries from the GetServices return of Network Nodes, there is a need to ensure that all service definitions loaded into the registry are valid and consistent with the definitions provided in the relevant Flow Configuration Documents. To accomplish this, the ENDS node will use Schematron validations to ensure that all service definitions are valid and match the FCD definitions before they are loaded into the database. If a service definition is found to be invalid, an email will be sent to the Exchange Network coordinator and the Node administrator informing them of the invalid entry. The Schematron rules for each flow service will be developed by the Exchange Network Governance during the Schema Conformance review process.

3.8 Additional Flow Tools and Resources

3.8.1 Web Interface

This is a web application that allows a node administrator to add, update, and delete services using a web browser. The application can also generate node service description documents, so that it can be used by the GetServices method.

4 Schema Information

4.1 Schema Structure

The schema for ENDS 2.0 is separated into two parts: 1) the Node Service Description Language (NSDL) and 2) the Data Element Description Language (DEDL).

4.1.1 Node Service Description Language

Generally, there are three levels of crucial information for describing node services:

Node Information: This includes the node name, node address, node version, and deployment environment.
Service Request Definition: This defines the service name, service type, dataflow name, and parameters.
Parameter Definition: This describes parameter name, type and other restrictions.

4.1.2 Data Element Description Language

Data Element Description Language (DEDL) is a schema used for describing various aspects of a data or data type, for use in the optional GetServiceDetails Query. This query is designed to complement the information stored in ENDS by allowing for granular description of specific data elements, including allowable values which may change on a node by node basis. The schema includes meta information such as data type, data range, maximum values, default value and format.

The DEDL schema will be released in a companion document that formally specifies the functionality of the GetServiceDetails query.

4.2 Schema Components

4.2.1 Node Service Description Language

4.2.1.1 Network Node Type

The network node type is a XML data type that is defined as:

```
<complexType name="NetworkNodeType">
        <sequence>
                <element name="NodeIdentifier" type="string"/>
                <element name="NodeName" type="string"/>
                <element name="NodeAddress" type="string"/>
                <element name="OrganizationIdentifier" type="string"/>
                <element name="NodeContact" type="string"/>
                <element name="NodeVersionIdentifier" type="typens:NodeVersionCode"/>
                <element name="NodeDeploymentTypeCode" type="typens:NodeStageCode"/>
                <element name="NodeStatus" type="typens:NodeStatusCode"/>
                <element name="NodeProperty" type="typens:ObjectPropertyType"
minOccurs="0" maxOccurs="unbounded"/>
                <element name="BoundingBoxDetails" type="typens:NodeBoundingBoxType"
minOccurs="0"/>
                <element name="NodeServiceList" type="typens:ServiceDescriptionListType">
                </element>
        </sequence>
</complexType>
```

The 7 elements have the following descriptions:

NodeIdentifier: The name of the node. Although this can be any string, it is recommended that the name is the same as the node name defined in the Network Authentication and Authorization Services (NAAS).
NodeName: A short name of the node.
NodeAddress: The SOAP address of the node. It should be a fully qualified URL.
OrganizationIdentifier: The name of the service provider.
NodeContact: Email address and/or phone number of the technical contact person.
NodeVersionIdentifier: The version number of the node. Valid version numbers are 1.1 and 2.0.
NodeDeploymentTypeCode: The deployment environment of the node. It should be *development*, *test* or *production*. This determines which security service to be used.

NodeStatus: The current status of the node. It should be the same status string returned by the NodePing method.
NodeProperty: Other properties associated with the node, such as a string description. This is a name-value pair that could occur multiple times.
BoundingBoxDetail: A set of four lat/long coordinates that defines the geographic boundary of a Node's data. The latitude and longitude measures should be strings separated by a space, with latitude being the first value and longitude the second. See below for an example:

```
	<BoundingCoordinateEast>43,11 -71.11</BoundingCoordinateEast>
	<BoundingCoordinateNorth>43,1 -71.1</BoundingCoordinateNorth>
	<BoundingCoordinateSouth>43,1 -71.1</BoundingCoordinateSouth>
	<BoundingCoordinateWest>43,11 -71.1</BoundingCoordinateWest>
```

The node information is relatively stable and can be entered into the central discovery service manually by a node administrator.

4.2.1.2 Service Description List Type

The Service Description List Type is an XML data type that contains a list of service descriptions. The XML schema segment below defines its structure:

```
<complexType name="ServiceDescriptionListType">
	<sequence>
		<element name="Service" minOccurs="0" maxOccurs="unbounded">
			<complexType>
				<sequence>
					<element name="MethodName" 
type="typens:NodeMethodTypeCode"/>
					<element name="Dataflow" type="string"/>
					<element name="ServiceIdentifier" type="string"/>
					<element name="ServiceDescription" type="string"/>
					<element name="ServiceProperty" 
type="typens:ObjectPropertyType" minOccurs="0" maxOccurs="unbounded"/>
					<element name="StyleSheetURL" 
type="typens:StyleSheetType" minOccurs="0" maxOccurs="unbounded"/>
					<element name="Parameter" 
type="typens:RequestParameterType" minOccurs="0" maxOccurs="unbounded"/>
				</sequence>
			</complexType>
		</element>
	</sequence>
</complexType>
```

The service description list may contain unlimited number of services; each service has the following children:

MethodName: The type of the service request, it must be one of the web methods defined in the Node Function Specification 2.0, such as Query, Solicit, Execute or Submit.
Dataflow: The name of a dataflow for the service. This is required for Query and Solicit method.
ServiceIdentifer: The name of the service request. This should be a unique identifier under a particular dataflow.
ServiceDescription: A text description of the service.
ServiceDocumentURL: A URL pointing to additional documentation of the service.
ServiceProperty: A name-value pair for additional service properties. The element is optional and may occur multiple times.
StyleSheetURL: A default style-sheet which could be used to transform the results to a displayable format. The element is optional. The StyleSheetType includes a string URL and an associated type identifier. The value of the type identifier is defined in the relevant Flow Configuration Document, but should generally indicate the output format of the instance document after the style sheet transformation.
Parameter: This is a list of parameters defined in the next section. The value of the element could contain a list of allowable values for the parameter separated by comma.

The service description can be used for describing node services offered through Query, Solicit, Execute or Submit. For data submission services, the description contains data flows supported by the node. The serviceIdentifer may be used to indicate operations provided within a dataflow.

4.2.1.3 Request Parameter Type

Request parameter type is an XML type for describing service parameters. There was a major change in the parameter binding between Node 1.1 and Node 2.0. In Node 2.0, parameters are named XML elements which may occur multiple times while, in Node 1.1, parameters are an unnamed array of values. The order of the parameters is thus significant in the Node 1.1 binding, but insignificant for Node 2.0.

The following parameter type definition attempts to cover both binding requirements:

```
<complexType name="RequestParameterType">
        <simpleContent>
                <extension base="string">
                        <attribute name="ParameterName" type="string" use="required"/>
                        <attribute name="ParameterType" type="string" use="optional"/>
                        <attribute name="ParameterTypeDescriptor" type="string"
use="optional"/>
                        <attribute name="ParameterRequiredIndicator" type="boolean"
use="optional" default="true"/>
                        <attribute name="ParameterEncoding" type="typens:EncodingType"
use="optional" default="None"/>
                        <attribute name="ParameterSortIndex" type="integer" use="optional"/>
```

```
                              <attribute name="ParameterOccurrenceNumber" type="typens:allNNI"
use="optional" default="1"/>
                    </extension>
          </simpleContent>
</complexType>
```
As can be seen, the RequestParameterType is a simple extension of string type with additional attributes:

ParameterName: The name of the parameter. The attribute is required.
ParameterType: The valid XSD data type of the parameter. The default parameter type is 'string'.
ParameterTypeDesciptor: The data element description type (see the Data Element Description Language) the parameter uses.
ParameterRequiredIndicator: A Boolean flag that indicates whether the parameter is required or not. For node 1.1, all parameters are required although a parameter value may be empty. The parameter is considered required if the attribute is absent.
ParameterEncoding: the encoding type of the parameter. Parameters typically are not encoded, but they may be encoded XML strings or base64 encoded binary values in some applications.
ParameterSortIndex: The sequential order of the parameters, only used for Node 1.1 service parameters.
ParameterOccurrenceNumber: The number of occurrences of the parameter. This attribute applies to Node 2.0 parameters only.

All attributes are optional. The value of RequestParameterType is the name of the parameter as shown in the following example:

```
     <Parameter ParameterName="State" ParameterType="String"/>      <Parameter
ParameterName="ZipCode"  ParameterType="String"/>
```

4.2.1.4 Network Node List Type

The Node List Type is an XML type that contains a list of network nodes. It is defined as:
```
<complexType name="NetworkNodeListType">
          <sequence>
                    <element name="NetworkNode" type="typens:NetworkNodeType"
minOccurs="0" maxOccurs="unbounded"/>
          </sequence>
</complexType>
```
The data type is primarily used by the Exchange Network Discovery Services v2.0 (ENDS 2.0) for rendering service information for multiple nodes.

4.3 Retrieving Granular Data from NSDL

The ENDS 2.0 model returns a large service description file that is intended to be cached on the local machine for fast retrieval in the future. Below are some common XPath example queries to retrieve granular information from a NSDL document. These were formerly individual queries on the ENDS node are provided to ease transition to the new ENDS model.

1. Retrieve all nodes that support a service request. This can be used for invoking a service at multiple Nodes:
   //NetworkNode/NodeName[../NodeServiceList/ServiceIdentifier='GetFacilityByZipcode']

2. Get all services offered by a node (given the node name - MyNode)
   //NetworkNode[NodeName='MyNode']/NodeServiceList

3. Get all v2.0 nodes in production
   //NetworkNode[NodeVersionIdentifier='2.0' and NodeDeploymentTypeCode='production']

4. Get a list of node names that support a specific dataflow (FRS):
   //NetworkNode[.//Dataflow='FRS']/NodeIdentifier

5. Get a list of service requests that have more than 4 parameters:
   //Service[count(Parameter)> 4]/ServiceIdentifier

6. Get the bounding box of a node (MyNode):
   //NetworkNode[NodeName='MyNodeName']/BoundingBoxDetails

7. Get a list of parameters of a service given the service identifier (MyRequest):
   //Service[ServiceIdentifier='MyRequest']/parameter

8. Get Style sheets associated with a service request (MyQuery):
   //Service[ServiceIdentifier='MyQuery']/StyleSheetURL

9. Get all unique services:
   //Services[not(ServiceName=../../preceding-sibling::NodeServiceList/Service/ServiceIdentifier)]/ServiceIdentifier

The expression above selects a service identifier if it is not defined in preceding nodes.

5 Data Service Information

5.1 Data Services

Service Name or Description	Service Type
Node Level Service Description	GetServices
GetServiceDetails	Query
GetNodeServiceList	Query
GetDataElementList	Query
Service Refresh	Submit
Element Set	Submit
Element Refresh	Submit

5.2 Node Services

5.2.1 GetServices

Description: Service publishing through the GetServices method is relatively easy to implement because the XML schema is very simple. A node could put services and parameters into database tables and retrieve them when the GetServices method is called. It could also store all the service information in XML documents and simply return a service description document when invoked.

The returned XML document should return an XML document adhering to the GetServices_v2.xsd schema that contains all of the registered services on the target node. Depending on the requestor's provided value for "serviceCategory", different subsets of the Node's services may be returned…etc etc

Data Service Type: GetServices

Return Method: N/A

Payload Format: XML document containing one and only one element of NetworkNodeType.

Data Service-level Business Rules: N/A

Error Conditions and Fault Follow-up Actions: This method should always return a result.

XML Header Usage: N/A

5.2.2 GetServiceDetails

Description: The GetServiceDetails service allows users to query a specific node and get a specific list of allowable values for that service. This query is optional for nodes to implement, but it is strongly encouraged for nodes that host many queries.

Data Service Type: Query

Query Parameters:

Name	Data Type	Required?	Max Length	Notes and Examples
Dataflow	String	Yes	N/A	Dataflow name as defined in the relevant FCD
Service	String	Yes	N/A	Service name as defined in the relevant FCD

Node 2.0 Dataflow parameter: ENDS_v20

Return Method: N/A

Payload Format: DEDL v2 Schema

Data Service-level Business Rules: This method may be implemented for some or all services supported by a node (solicit and query only). The service should return a full result set of the allowable values for a particular service. If a full enumerated values list is unavailable, allowable values must be empty.

Error Conditions and Fault Follow-up Actions: N/A

XML Header Usage: N/A

5.3 ENDS Services

5.3.1 GetNodeServiceList

Description: The GetNodeServiceList service allows users to filter services by NodeName, Dataflow, Service and Version. It is possible to get individual service information as needed at runtime. However, to improve system efficiency and performance, it is recommended that all relevant services information be retrieved at once and cached locally, based on the fact that service information is not updated frequently.

Data Service Type: Query

Query Parameters:

Name	Data Type	Required?	Max Length	Notes and Examples
NodeName	String	Yes	N/A	May be an empty value to search all Nodes
Dataflow	String	Yes	N/A	May be an empty value to search all dataflows
Service	String	Yes	N/A	May be an empty value to search all services
Version	String	Yes	N/A	Can only be ‘1.1’ or ‘2.0’. If the value is omitted, all versions will be returned.

Node 2.0 Dataflow parameter: ENDS_v20

Return Method: N/A

Payload Format: The return of the query contains an XML element of NetworkNodeListType (see sample document in section 4.2.1.4).

Data Service-level Business Rules: N/A

Error Conditions and Fault Follow-up Actions: N/A

XML Header Usage: N/A

5.3.2 Service Refresh

Data Service Type: Submit

Data Service Parameters:

Parameter Name	Parameter Value	Comments
securityToken	Obtained from previous Authenticate call.	NAAS will be used for validating the token.
transactionId	Empty	
dataflow	ENDS_v20	
flowOperation	Refresh	ENDS 2.0 always perform a refresh operation.
recipients	Empty	
notificationURI	An optional email address for receiving status notification.	ENDS 2.0 will send status information if the parameter is a valid email address
Documents	Contains a single document in either XML or ZIP format.	

Node 2.0 Dataflow Parameter: ENDS_v20

Return Method: As defined in the Network Node Specification 2.0, ENDS 2.0 will return a transaction ID if the submission is received successfully. The submission will be processed asynchronously. A submitter may use the GetStatus method to retrieve additional information associated with the transaction.

Payload Format: N/A

Data Service-level Business Rules: A node administrator may submit a Node Service Description Language document to the ENDS 2.0. The submitted document will be validated and loaded into ENDS database if successful.

To simplify the operations, ENDS 2.0 performs a refresh operation when loading the data. This is means that the submitted document must contain complete service description of the node.

The refresh operation can be described as follows:

If a service or parameter doesn’t exist in the ENDS 2.0, it will be inserted.
If a service or parameter already exists in the ENDS 2.0, it will be updated.
If a service or parameter exists in ENDS 2.0 but not in the submitted document, it will be deleted.

The submitted NSDL document must contain one and only one node due to constrains of ENDS governance and authorizations.

Error Conditions and Fault Follow-up Actions: N/A

XML Header Usage: N/A