SIF Infrastructure Specification 3.2.1: Utilities

www.A4L.org

Version 3.2.1, March 2017

1.   Introduction

A SIF 3 Utility Service is a Data Service that *provides* a data object related to the SIF infrastructure, rather than an object which is part of a particular locale-specific data model.

- They support some or all of the identical set of Consumer Requests (Query, Create, Update and Delete).

- Some Utility services may be subscribed to, and publish object Events when their internal data changes

- They are accessed by Consumers through the *requestsConnector* and return any delayed Responses and Events into the specified *Consumer Polling Queue*.

- They can be located by Consumers through a Provider entry in the Global Zone section of the Service Providers Registry.

- They are documented with the infrastructure rather than the set of educational Object Services which use that infrastructure, and are present in every release based upon that infrastructure version, independent of locale.

- They may be implemented independently of the core Infrastructure Services or any other Utility Service, and integrated into a site-wide solution in the same manner as implementations of other Object Services.

- They have their formats defined within the Infrastructure namespace (http://sifassociation.org/infrastructure/3.2.1), rather than in one of the Data Model namespaces which define the payload of the other Object and Functional Services.

1.1.   Required Reading

Please refer to the *Basic Architecture* document for an understanding of the terminology, concepts and global XML element definitions, Service types, operation descriptions and exchange choreographies that will be referred to here. Additionally, the legal notices contained in the Basic Architecture apply to this document.

Please refer to the *Infrastructure Services* document to gain an understanding of the "core" Infrastructure Service components, which these Utility Services supplement.

Some familiarity with both those documents is assumed as a prerequisite, as all Utility Services will employ the Service framework and leverage the specific Infrastructure Services described there.

1.2   Utility Service Types

As noted, all Utility Services are accessed through the *Requests Connector* Service and follow the standard Object Service API in that they support some or all of the defined Object Service Provider interface methods (Query, Create, Update and Delete).   There are two additional characteristics that are unique to this set of Services.

1.2.1   Administrative Level Authorization

Particularly in Brokered Architectures, there may be one or more administrative Consumer applications that are used to configure and maintain the Environments of other Consumers. These applications may be granted authorization rights to additional Provider Service CRUD Methods unavailable to non-administrative Consumers.

Where they exist, such extensions are site specific and except in a few cases, will not be explicitly defined in this document.

1.2.2   Scope

All Utility Services span the entire Consumer's Environment. They each have a single entry in the Service Providers Registry, located in the *environment-global* Zone.

This means that any in any Consumer Request, the combination of a *serviceType* value of "*UTILITY*" and the *serviceName*, provides enough information to allow the Connector in a Brokered Architiecture to first determine the identity of the Utility Service Provider, which should receive it, and then construct the exact Service URL to route it to.

As a result, any *zoneId* provided by the Consumer in a Utility Service Request does not affect the ultimate destination of the request, and can be treated differently than if the serviceType value was "*OBJECT*" or "*FUNCTIONAL*". There, a Consumer-provided zoneId may result in the Connector forwarding the Consumer's Request to an entirely different Service Provider URL (ex: an SIS for Special Ed students or one spanning only a specific middle school).

When a Service Consumer issues a Request to a Utility Service Registry however:

- If the Consumer explicitly specified the *environment-global* Zone, a non-qualified Query is assumed and the Utility Service returns all entries in its Registry. If the Consumer specified any other legal Zone (or its default *zoneId* was supplied by the Connector) only those entries applicable to that *zoneId* are

returned. Essentially the *zoneId* parameter serves as a query constraint on the entries in the Registry.

- For the Service Providers Registry, this means a Consumer gets only the Provider entries for its default Zone or, if it specified the *environment-global* Zone, it gets all Service Providers for all Zones.

- For the External Code List Registry (and others), things are somewhat different. The default assumption is that all External Code Lists are applicable to all Zones, and are therefore "Global" in scope, and their entry has a *zoneId* value of *environment-global*.

Whenever this is not true (for example grade enumerator codes may be different for different school Zones in the District Environment) a Zone-specific External Code List entry of the same name but with the different set of values is created, and tagged with the corresponding *zoneId* value.

When a Zone-specific query request is issued, each entry in the global list is then examined in turn

  o If there is no corresponding Code List with that name defined specific to the selected Zone, the Global Code List entry is returned.

  o If there is a specific entry for that Code List in the selected Zone, that entry is returned instead.

The end result is that when a Consumer issues a request to a Utility Service with a *zoneId* other than environment-global, only those entries (ex: Providers, External Code Lists) relevant to the specified Zone are returned. If the zoneId specified is *environment-global*, all entries across all the Zones in the Consumer Environment will be reported.

1.2.3 Functionality

The generic functionality provided by each Utility Service in supporting the Environments Provider API is shown in the following table. Unless otherwise noted, no Utility Service supports XQuery or Paged Query Requests.

They will all be described in further detail in the sections that follow.

Utility Registry Services	Functionality	Access
Zones	Contains the name and description of all Zones in the Consumer's Environment (not all	Only Query access is enabled for non-Administrative Consumers. Change Requests

	of which may be accessible by the Consumer).	are not supported.
Providers	Contains entries with the name, service type, vendor information and any supported Context of all Service Providers in a given Zone (not all of which may be accessible by the Consumer).	Only Service Providers operating in a Brokered Architecture may create and change the contents of their Registry Entries. All Consumers in both types of Architectures have Query functionality.
Namespaces	Used to retrieve namespace URIs and their corresponding schema location URLs, currently valid within the zone.	Only Query is enabled for non-Administrative Consumers. Change Requests are not supported.
Code Sets	Validates specific code values claimed to be part of an identified named code set, used to reduce breaking changes introduced via normative references to external sources like the NCES Handbook.	Only Query is enabled for non-Administrative Consumers. Change Requests are not supported.
Named XQuery	Accepts, validates (per-Consumer) and provides access to previously submitted or pre-defined XQuery Templates.	All approved Service Providers can use the supplied Token to retrieve the corresponding XQuery Script using Query by (XQuery Template Token) ID. This script is then used as the basis of expanding and executing a Consumer issued XQuery Request containing that Token and a set of values corresponding to the parameters in the template. Only Query is enabled for non-Administrative Consumers. Change Requests are not supported.
Other Utility Services	**Functionality**	**Access**
Alerts	Accepts and forwards reported alerts (notifications warnings and errors) in accordance with the pre-specified diagnostic and security policies defined by the site administrator.	All Service Consumers can create an Alert, and issue Query by (Alert) ID to access their own previously reported Alerts. Only administrative Consumers can access and change the Alerts issued by other Consumers.

2. Zones Registry

The Zones Registry Utility Service contains the name and description of all Zones visible to a registered Consumer within its Environment. Some Zones may contain services which are not accessible by the Consumer due to authorization restrictions.

2.1 Zones and Consumer Requests

A *Zone*[1] is basically a collection of Service Providers and associated Utility Registry information within the Consumer's Environment, pre-organized by the site Administrator to correspond to a discrete hierarchical grouping within the owning educational organization such as a school or district, or an alternative grouping such as the set of applications supporting Special Ed students. Zone identifiers are chosen by the administrator and can follow any convention that best meets the needs of the deploying organization.

Each Data Object and Functional Service "instance" accessible within the Consumer's Environment is scoped to a Zone, although a given Service Provider implementation may support the same Service Provider interface in several Zones. As noted above, there is one "special" Zone (*environment-global*) that is reserved for "globally available" (i.e. Architecture-wide) Utility Services.

The Zone in which the Service is to be found always qualifies every Consumer request for any Provider Service. Each Service Consumer is assigned a "*default*" Zone at Registration time, which is used whenever a specific Zone is not explicitly included in one of its Provider Service Requests. If there is no matching registered Service Provider for any Consumer Request, the request must fail.

2.2 Presence and Scope

The presence of a Zones Registry Utility Service is mandatory for all Environments that support Consumer self-provisioning, whether Direct or Brokered. In the simplest case (typically a Direct Architecture provided by an SIS or LMS application), the Zones Registry might consist solely of the Zone ID and Zone Description (with no additional parameters) for two Zones:

---

[1] *Please refer to the SIF 3.0 Base Architecture document for a more complete description of a SIF Zone.*

- *environment-global*
- XXX

In this case "XXX" would be the "default" Zone assigned to all registered Consumers. Its value might represent the name of the application providing the non-utility services (such as Assessment).

A Consumer does not need to access the Zones Registry if it interacts solely with the set of Service Providers contained in its assigned "default" Zone, and the available Utility Services in the environment-global Zone.

2.3   Supported Operations

The set of Zones is fixed for each Registered Consumer, and does not change while the Consumer is active. Therefore the Zones Registry Service is not required to support change requests or to publish change events.

The only guaranteed operation for non-Administrative Consumers is Bulk Query. There is no support for dynamic Query, and there are no defined Zones Registry Service Paths or XQuery Templates.

2.4   Zones Registry Data Elements

The following data elements comprise the contents of the Zone Service Registry.

Element	Char	Description	Type
zones		Top level element of the Zones Registry	Collection
zones/zone	RM	Individual entry defining a specific Zone within the Consumer's Environment	Zones Registry Entry
zones/zone@id	M	The unique id of the Zone, and the key to the Zone element in the Zones Registry. Typically this is the globally unique and recognizable "name" of the Zone, rather than a meaningless code.	xs:token
zones/zone/description	O	A short (possibly multi-sentence)	xs:normalizedString

		description of the Zone	
zones/zone/properties	O	An optional set of free-form name value properties further defining the Zone element	
zones/zone/properties/property	MR	The value of a specific property which defines the Zone.	xs:token
zones/zone/properties/property@name	M	The name of that property for which the value applies	xs:token

2.5   XML Example

The payload of the Response to Query Request issued to the Zones Registry Service in a minimal Direct Architecture is shown below.

```
<zones>
  <zone id="environment-global">
    <description>Contains all global Utility services</description>
    <properties>
      <property name="type">Utilities</property>
    </properties>
  </zone>
  <zone id="SuffolkMiddleSchoolSIS">
    <description>Scopes SIS Service Provider at School</description>
    <properties>
      <property name="Special Ed">Student</property>
      <property name="administrator">alexj@SuffolkMiddleSchool.va.edu</property>
    </properties>
  </zone>
</zones>
```

3. Providers Registry

There is one *Providers Registry Utility* Service per Consumer Environment. The Registry contains a list of Provider Entries, each of which includes:

- A Zone ID

- The type of the provided Service in that Zone (Utility, Data Object, Functional, Service Path or XQuery Template).

- The URL identification of the provided Service.)

- Any specific Context that Service supports.

- Any Service functionality extensions (such as support for Dynamic Queries or the ability to provide "Total Count")

All potentially accessible Services have an entry in the Providers Registry (including the Providers Registry Utility Service itself), although full or even partial Consumer access to that Service is determined by the access rights currently granted in the Consumer's Environment object[2], and is not guaranteed.

As noted in the Basic Architecture document sections on Service Types, there are 5 types of Services which could be represented in the Provider Service Registry.

Service Type	Defined	URL Service Identifier in Registry Entry (example)	Operations
Data Object	Data Model Schema	students	Query, Create, Update and Delete
Utility Object	Infrastructure 3.0 Schema	zones	Defined in this document (Query only in this case)
Functional Service	Data Model Schema and Functional Description	studentRecordExchanges	Specific to the Function
Service Path	Data Model Binding for Object Type. Schema identical to schema of object type name in final URL segment	*sections/{}/students* where "{}" indicates the section to report Students for	Read Only
XQuery Template	Data Model Binding defines Schema and XQuery script contents	StudentSnapshot	Read Only

---

[2] *Please refer to the Environments Service section in the Infrastructure Services document.*

3.1   Supported Operations

In many cases, a Service Consumer will be pre-provisioned to be able to access the set of Service Providers it must rely on to perform its functions. In that case, it does not need to utilize the Providers Registry Service.

However at sites where there the security policy is "authorization on demand", any Service Consumer with the proper authorization rights may be expected to dynamically utilize the Providers Registry to discover available Service Providers in its default Zone or elsewhere. At that point it can dynamically self-Provision itself to issue Requests and (where applicable) to subscribe to Events from one or more of these Providers

Such usage requires the ability to Query the Providers Registry.

If the Consumer has the proper authorization (and is deployed in a Brokered Architiecture), it can also register itself as a Service Provider by creating one or more of its own Provider Entries that will in turn be visible to other Consumers[3]. If the Service Provider implementation supports multiple service options (multiple Zones, multiple Contexts, multiple Data Model version numbers, etc.) it **must** create multiple corresponding entries in the Provider Registry.

The Providers Registry Service must publish change Events when Service Entries are added, updated or deleted. The Providers Registry Service itself does not support dynamic Query, and there are no defined Providers Registry Service Paths or XQuery Templates.

Operation	Description
Query	Provides the Consumer with a list of all accessible Service Providers within the specified Zone, or (if access was through the environment-global Zone), a list of all Service Providers of all types in all Zones.
Create (Brokered Architectures only)	Allows a Consumer to initially provision (or attempt to provision) itself as a Service Provider. As in any Create Request, multiple Provider Entries may be created with this call (as when the Consumer requests to be a Provider of a given object type for multiple Contexts).

Success results in the creation of one or more new Provider Entries. |

---

[3] *This is very similar to when a SIF US 2.6 application provisions itself as an Object Provider.*

Delete (Brokered Architectures only)	Removes one or more specified Entries in the Providers Registry. Except for Administrative Tools, this Request must be rejected for all Consumers other than the one which created the entry in the first place.
	Success results in the deletion of the specified entries from the Providers Registry. At this point these Services are unavailable to all Consumers[4].
Update	Prohibited

3.2   Providers Registry Data Elements

The following data elements comprise the contents of the Service Providers Registry. As noted, the entries returned in the response to a Consumer Query are scoped by the Zone in which the Providers Registry is accessed.

Element	Char	Description	Type
providers		Top level element in the Providers Registry	Collection
providers/provider	RM	Individual entry defining a specific provider	Providers Registry Entry
providers/provider@id	M	The unique id of the Provider entry. This is typically only used by the Service Provider to directly Query or Delete its own entry It may be a hashed combination of the next 5 entries in this table or it may be a standard UUID.	xs:token
Service "Scoping" Elements			
providers/provider/serviceType	M	The "generic" type of Service being provided. (Note: the namespace of Utility objects is fixed by the infrastructure version. The namespace of all object and functional Services **must** match the one the Consumer gave when it registered, and are not repeated in the Providers Registry)	One of: UTILITY OBJECT FUNCTIONAL SERVICEPATH XQUERYTEMPLATE

---

[4] *Such an Entry Deletion request is generally issued as part of an ordered "unregister" sequence for the Provider, prior to its deleting the Environment. It is not required that a Provider issue this however.*

providers/provider/serviceName	M	The part of the URL in the Consumer Request that, along with the Context and Zone **must** uniquely identify the recipient which will provide the response. This element is an XML Token, whose exact meaning is determined by the Service Type. In all cases this value determines exactly what the format of the message payloads will be. Examples: If the Service Type is: • **UTILITY:** then *zones* is the Zones Registry Utility Service (one of a fixed list of services) • **OBJECT:** then **sections** is the Sections object provider • **SERVICEPATH:** then **sections/{}/students** will return all Student objects in the Section who's RefId is supplied in place of the "{}" in the URL segment between *sections* and *students (ex: sections/1234/students returns all Students in Section 1234)* **XQUERYTEMPLATE:** then **StudentSnapshot** is an XQuery Template Token which is predefined to return a specific report. **FUNCTION**: then **StudentRecordExchange** is a Functional Service which creates a unique job object for the issuing Consumer which encapsulates the behavior of what can be a multi-step process.	xs:token
providers/provider/contextId	M	The value of the Service Context being provided (if any). If there is no context associated with this object type, the value of this element is "DEFAULT".	xs:token
providers/provider/zoneId	M	The Zone in which the Service Context is being provided. This must correspond to one of the entries in the Zones Registry. The combination of ZoneId, Service Type, Service Name, Context Name and (if present) Context value **must** be unique for every entry in the Service Providers Registry. .	xs:token
"Functionality Description" Elements			
providers/provider/providerName	M	The name of the Service Provider as it would be referred to by the administrator of the Zone (ex: RamseySIS).	xs:token
providers/provider/q	M	A set of elements corresponding to the extent of Service	

uerySupport		Provider support for Consumer Query Requests (above and beyond support for Query by Id, which is mandatory and assumed).	
providers/provider/querySupport/dynamicQuery	O	Present if this Service is capable of processing a Dynamic Query "where" argument when included in the URL of a Query Request.[5]	xs:boolean
providers/provider/querySupport/queryByExample	O	True if this service is capable of honoring query by example payloads.	xs:boolean
providers/provider/querySupport/changesSinceMarker	O	True if this service is capable of supporting Changes Since Queries.	xs:boolean
providers/provider/querySupport/paged	O	True if this service is capable of responding to Paged Query Requests.	xs:boolean
providers/provider/querySupport/maxPageSize	MC	If Paged Queries are supported, this is the maximum number of Objects that will be returned on a Page of Query results.	xs: unsignedInt
providers/provider/mimeTypes	MC	A complete list of supported Internet Media Types. If any Internet Media Type other than application/xml is supported, this list **must** be present. When present, this list **must** contain all supported Internet Media Types.	
providers/provider/mediaTypes	M	An Internet Media Type supported by this adaptor (for this service). **Examples:** • application/xml • application/json	xs:token
providers/provider/querySupport/totalCo	MC	If Paged Queries are supported, this indicates whether the Service Provider will return a "total count" of all objects satisfying the original Query, either in each Paged	xs:boolean

---

[5] *The actual level of Dynamic Query support may vary in terms of whether all comparators are supported or only "=", whether multiple "and / or" Boolean expressions are supported, and how deeply nested the identified elements in the "where" argument can be. This level of detail is specific to the Object type and may be also be defined in the Data Model "binding".*

unt		Response, or the Response to a Paged Request with a Page Size of zero.	
"Product Description" Elements **(these are taken from the Provider's Environment and are omitted from the Provider Entry Create but returned in the Query)**			
providers/provider/querySupport/applicationProduct	C	Application Vendor Identification	productIdentityType[6]
providers/provider/querySupport/adapterProduct	O	Adapter Vendor Identification	productIdentityType
"Hidden" Elements **(this is included in the Provider Entry create but not returned in the Query)**			
providers/provider/endPoint	MC	In a Brokered Architiecture, this element contains the URL where all requests for this Service Provider should be re-invoked by the Connector (which received them from the Consumer). **Note:** This URL is independent of Zone and Context information, which is supplied elsewhere in this entry.	protocolType[7]

3.3   XML Examples

1. The payload of a *create* Provider request issued to the Providers Registry Service is shown below. Note the end point specification.

```
<providers>
  <provider id="0940337c-27bb-11e6-b67b-9e71128cae77">
    <serviceType>OBJECT</serviceType>
    <serviceName>students</serviceName>
    <contextId>DEFAULT"</contextId>
    <zoneId>SuffolkMiddleSchoollSIS</zoneId>
    <providerName>RamseySIS</providerName>
    <querySupport>
      <dynamicQuery>true</dynamicQuery>
      <totalCount>false</totalCount>
```

---

[6] *See Appendix D of the Infrastructure Service Document for an expansion of this Data Type.*
[7] *See Appendix D of the Infrastructure Service Document for an expansion of this Data Type.*

```
      <paged>true</paged>
      <maxPageSize>3000</maxPageSize>
    </querySupport>
    <endPoint>https://www.suffolk-
jhs.pedmontdistrict.nc.edu/suffolkmiddleschool/sis</endPoint>
  </provider>
</providers>
```

2. The payload of the Response to a Query Request which corresponds to the RamseySIS entry created above. Note the endPoint element is not shown, and application identification information obtained from the RamseySIS Consumer Environment is included in this entry.

```
<providers>
  <provider id="c3d818e4-27bb-11e6-b67b-9e71128cae77">
    <serviceType>OBJECT</serviceType>
    <serviceName>student</serviceName>
    <contextId>DEFAULT"</contextId>
    <zoneId>SuffolkMiddleSchoollSIS</zoneId>
    <providerName>RamseySIS</providerName>
    <querySupport>
      <dynamicQuery>true</dynamicQuery>
      <totalCount>false</totalCount>
      <paged>true</paged>
      <pageSize>3000</pageSize>
    </querySupport>
    <applicationProduct>
      <vendorName>Raffles Software</vendorName>
      <productName>RafflesSIS</productName>
      <productVersion>1.5</productVersion>
    </applicationProduct>
    <adapterProduct>
      <vendorName>SifRus</vendorName>
      <productName>GlobalAdapt</productName>
      <productVersion>4.2</productVersion>
    </adapterProduct>
  </provider>
</providers>
```

4.   Namespaces Registry

The Namespaces Registry contains the set of XML namespace URIs and their corresponding schema location URLs that are currently valid within the Environment. Both Service Consumers and Providers **SHOULD** stay synchronized with this registry in order to prevent and detect namespace mismatches while exchanging messages.

In terms of guaranteeing interoperability and enforcing element-level privacy restrictions, controlling the Namespaces contained in exchanged messages is as important to Environment Administrators as controlling the XML elements themselves, because Consumer to Provider interoperability requires standardization of the name, value <u>and</u> scope of every element exchanged.

This imposes a Consumer to Provider co-dependency, in that each has to construct messages containing only the namespaces that the partner has the schemas for and can validate. In general, such schema agreement is static and predefined by either the version of the Infrastructure or Data Model supported within the Environment. In this case, neither the Consumer nor Provider need access the Namespaces Registry.

However in Environments where namespaces are subject to change, where customized profile extensions to the core SIF schemas have been made, and especially where the Environments Provider, Data Warehouse or Service Provider is bridging between versions, the Namespaces Registry allows for both synchronization and just in time retrieval of needed schemas, to minimize rejection of messages containing unexpected namespaces.

In particular, with the separation of infrastructure and data model in SIF 3, there is no longer one Namespace that all Servers and Providers within the Environment must conform to. The Namespaces Registry is provided as the only sure way for a Consumer determining what the Providers within the Environment will expect to see. It allows Service Consumers wanting to place XML on the wire or retrieve data over the wire to dynamically determine whether the namespaces (including version) they rely on are supported within the Environment before they begin operation.

4.1   Service Implementation Strategy

Each entry in the Namespaces Registry carries only the URL of a single schema file, which by convention leverages the XML Schema Namespaces themselves, other Namespaces present in

the Registry, and/or a replicable relative path structure and one or more file references from the targeted location.

The ability to Query the Namespaces Registry is required in all Environments where the Namespace Registry is supported. The Namespaces defining the infrastructure and Data Model versions supported in the Environment can be read, but entries can only be created and deleted by Administrators. There are no Namespaces Registry Events.

4.2   XML Schema Snippet

```xml
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="imports/xml/xml.xsd"/>
```

4.3   Supported Operations

Request	Direct Architecture	Brokered Architecture
Query	M	M
Create	P	P
Update	P	P
Delete	P	P
Events	P	P

There is no support for dynamic Query, and there are no defined Namespaces Registry Service Paths or XQuery Templates.

4.4   Namespaces Registry Data Elements

Element	Char	Description	Type
namespaces		Collection element used where multiple namespaces MAY be conveyed.	Collection

namespace	M		Namespaces Registry Entry
@id	M	Objects unique ID.	UUID
namespace/zone	M	If this namespace is tied to a specific zone it is specified here, otherwise "environment-global" indicates applicability to all zones.	xs:token
namespace/uri	MU	The namespace URI of the retrieved URL.	xs:anyURI minLength=0 maxLength=2048
namespace/url	MN	The URL of the specified namespace URI. If empty the namespace indicated by the URI is not valid within this Zone.	xs:anyURI minLength=0 maxLength=2048

4.5   XML Examples

```xml
<namespaces xmlns="http://sifassociation.org/infrastructure/3.2.1">
  <namespace id="97eea29f-013c-1000-007f-14109fdcaf83">
    <zone>environment-global</zone>
    <uri>http://sifassociation.org/infrastructure/3.0</uri>
    <url>http://sifassociation.org/infrastructure/3.0/core.xsd</url>
  </namespace>
  <namespace id="97eea29f-013c-1000-007f-14109fdcaf83">
    <zone>environment-global</zone>
    <uri>http://www.sifassociation.org/datamodel/us/3.0</uri>
    <url>http://www.sifassociation.org/datamodel/us/3.0/US3p00p04.xsd</url>
  </namespace>
  <namespace id="97eea29b-013c-1000-007f-14109fdcaf83">
    <zone>urn:edu:wwu</zone>
    <uri>urn:org:pesc:sector:AdmissionsRecord:v1.1.0</uri>
    <url>
http://www.pesc.org/library/docs/standards/Admissions%20Application/AdmissionsRecord_v1
.1.0.xsd
    </url>
  </namespace>
</namespaces>
```

5.   Code Sets Registry

The SIF standard includes multiple normative dependencies on external code sets such as the NCES Handbook. When one of these code sets is revised, the first application utilizing the updated codes will likely not interoperate with previously deployed SIF applications conforming to the earlier version of the code set.

Any such a code set revision then represents a potential breaking change which is asynchronous to the SIF standard release cycle. As a result, Consumer / Provider interoperability is impacted even between SIF applications conforming to the same SIF minor release.

Even in the case where all Code Set values are defined directly in the SIF standard, the addition of a single new value to an enumerated code set in a minor release can break interoperability in ways that the addition of a new element cannot. Reception of a new element can be simply ignored by an application confirming to an earlier release, because everything it expected is still present in the arriving message. But reception of a new code set value cannot be ignored, because it provides a value which cannot be determined to be legal for an existing element that the older application may need to store.

The Code Set Registry Service provides a way for <u>all</u> legal codes to be defined outside of the SIF Specification while allowing changes (additions and replacements) of external code set values to be easily verified by the recipient so as not to break existing Consumer / Provider interoperability.

Codes are arranged (and scoped) under their respective code sets within the Registry. They are specifically <u>not</u> namespace qualified so they can be built up from multiple sources and remain simple to employ.

5.1   Service Implementation Strategy

Consumers and Providers can use the Code Set Registry to verify and then accept codes that were not yet defined when they were written. Depending upon the implementation, all or part of the Code Sets Registry can be seeded either manually or by having the Registry automatically draw in codes from multiple sources and reconcile them with the manual entries.

Query is a mandatory Code Set Registry operation. All others (*create, update* and *delete* of both code lists and their individual entries) are optional, and may require manual administrative actions to achieve.

However entered, if an unexpected code value is received by a Consumer or Provider, the Code Set Registry **must** be capable of verifying whether the value is in fact legal.

5.2   Alternatives

If a Code Sets Registry Utility Service is not available in the Environment, Consumers and Providers **MUST** employ one or more of the following strategies for dealing with the arrival of a code which does not match the expected enumerated list of values defined when they were created.

- **Blind Trust**: What is received is accepted by the recipient and placed into its data store. This is the simplest approach and removes the need to support or reference the Code Sets Utility Service. However it may result in inconsistent or erroneous data, since it ignores the possibility of sender error.

- **SIF File Import**: The SIF Association intends to publish a series of Code Set files in the format shown below. Administrators at a given site can deploy a Code Sets Registry and feed in these files as a way of initializing the Service.

- **Other Import Techniques:** Additional code sets can be copied directly from their source on the web or hand entered.

If the code value is not found in the Registry, the remainder of the message can be accepted, but either way an Alert **SHOULD** be issued.

5.3   Supported Operations

Request	Direct Architecture	Brokered Architecture
Query (both Bulk and Paged)	M	M
Create	P	P
Update	P	P

Delete	P	P
Events	M	M

There is no support for dynamic Query, and there are no defined Code Sets Registry Service Paths or XQuery Templates. Individual Code Set entries may be returned if Query by ID is requested.

5.4   Code Sets Registry Data Elements

Element	Char	Description	Type
codeSets		Collection element used where multiple code sets **MAY** be conveyed.	Collection
codeSet	M		Member
@id	MIU	The unique name of the code set	xs:token minLength=0 maxLength=128
codeSet/zone	M	If this is tied to a specific zone it is specified here, otherwise "*environment-global*" indicates applicability to all zones in the Environment.	xs:token
codeSet/version	M	The official major, minor, and revision version of the code set. **MUST** be allowed to rev independent of (and more often then) both the data model and infrastructure used.	version
codeSet/timestamp	M	The date and time of the last change to this code set. So that only updates can be queried	xs:dateTime

		efficiently (where applicable).	
codeSet/source	CN	URL to an external code set whose values are not (yet) present in the system. Either source or codeItems **MUST** be present, however both **MUST NOT** be included.	xs:anyURI minLength=0 maxLength=2048
codeSet/codeItems	C	Either source or codeItems **MUST** be present, however both **MUST NOT** be included.	
codeSet/codeItems/codeItem	MR	A code in the set.	
codeSet/codeItems/code	MI	Official abbreviation for the given value.	xs:token minLength=1 maxLength=16
codeSet/codeItems/source	SN	URL where the related abbreviation comes from.	xs:anyURI minLength=0 maxLength=2048
codeSet/codeItems/namespace	SN	Namespace where the related abbreviation comes from.	xs:anyURI minLength=0 maxLength=2048
codeSet/codeItems/value	SN	What the code represent. Example: Morning Kindergarten	xs:token minLength=0 maxLength=128
codeSet/codeItems/description	ON	Human readable explanation of what the code represents.	xs:normalizedString minLength=0 maxLength=1024
codeSet/codeItems/definition	ON	Human readable explanation provided by the source!	xs:normalizedString minLength=0

			maxLength=4096
codeSet/codeItems/aliases	O		
codeSet/codeItems/aliases/alias	MR		
codeSet/codeItems/aliases/alias/code	M		
codeSet/codeItems/aliases/alias/code/old	M	Indicates the alias code has been deprecated or deleted, even if it no longer appears in the register. Example: false	xs:boolean
codeSet/codeItems/aliases/alias/code/official	M	Indicates the alias code is an official one; however the locale uses this one to represent that concept, at least in part. Example: true **MAY** be used when cleaning data to leave the Environment.	xs:boolean
codeSet/codeItems/aliases/alias/code/value	M	Alternate abbreviation that might need to be more specific or correct.	xs:token minLength=1 maxLength=16
codeSet/codeItems/aliases/alias/source	SN	URL where the related abbreviation comes from.	xs:anyURI minLength=0 maxLength=2048
codeSet/codeItems/aliases/alias /namespace	SN	Namespace where the related abbreviation comes from.	xs:anyURI minLength=0 maxLength=2048
codeSet/codeItems/action	M	The last action to be taken for this code item. Deleted code items **SHOULD** remain in the code set!	xs:enumeration ADD CHANGE DEPRECATED

			DELETE
codeSet/codeItems/timestamp	M	The date and time of the last action taken on this code item. So that only updates can be queried efficiently (where applicable).	xs:dateTime

5.5 XML Example

In the example below the SIF 2.x Data Model was used, and the appropriate Data Model Namespace is (counter-intuitively) "`infrastructure/2.x`".

```
<codeSets xmlns="http://sifassociation.org/infrastructure/3.2.1">
  <codeSet id="GradeLevels">
    <zone>environment-global</zone>
    <version>3.1</version>
    <timestamp>2012-09-01T07:00:13.000-07:00</timestamp>
    <codeItems>
      <codeItem>
        <code>K</code>
        <source>http://vocabulary.curriculum.edu.au/schoolLevel/</source>
        <namespace>http://www.sifinfo.org/infrastructure/2.x</namespace>
        <value>Kindergarten</value>
        <action>ADD</action>
        <timestamp>2012-01-01T07:00:13.000-07:00</timestamp>
      </codeItem>
      <codeItem>
        <code>1</code>
        <source>http://vocabulary.curriculum.edu.au/schoolLevel/</source>
        <namespace>http://www.sifinfo.org/infrastructure/2.x</namespace>
        <value>First Grade</value>
        <action>ADD</action>
        <timestamp>2012-01-01T07:00:13.000-07:00</timestamp>
      </codeItem>
    </codeItems>
    <!-- Ten more entries here (2nd through 11th). -->
    <codeItems>
      <codeItem>
        <code>12</code>
        <source>http://vocabulary.curriculum.edu.au/schoolLevel/</source>
        <namespace>http://www.sifinfo.org/infrastructure/2.x</namespace>
        <value>Senior</value>
        <action>ADD</action>
        <timestamp>2012-01-01T07:00:13.000-07:00</timestamp>
      </codeItem>
      <codeItem>
        <code>KA</code>
```

```xml
        <value>Morning Kindergarten</value>
        <aliases>
          <alias>
            <code>
              <old>false</old>
              <official>true</official>
              <value>K</value>
            </code>
            <source>http://vocabulary.curriculum.edu.au/schoolLevel/</source>
            <namespace>http://www.sifinfo.org/infrastructure/2.x</namespace>
          </alias>
        </aliases>
        <action>ADD</action>
        <timestamp>2012-09-01T07:00:13.000-07:00</timestamp>
      </codeItem>
    </codeItems>
  </codeSet>
</codeSets>
```

6.   Named XQuery Registry

The Named XQuery Registry contains the Environment-wide collection of registered XQuery Templates (XQuery scripts which incorporate externally set parameter values). This collection defines the entire set of XQuery scripts that Consumers may legally issue for execution by Service Providers. Each Named XQuery has an associated unique ID under which it may be referenced.

Many if not all of the templates in a typical Named XQuery Registry may have originally been specified as part of the Infrastructure "binding" of the Data Model release in the form of a profile. For example an Named XQuery called "StudentSnapshot" will, when requested, result in a Response with a predefined format.   Such templates are known "in advance" by developers of both Service Consumer and Service Provider applications, and serve primarily as a report documentation tool.   Exactly how the report is generated remains an implementation detail left to the Service Provider.

Other Named XQuerys may be installed specifically by the site administrator or dynamically created by administrative-level Consumer applications.

All are used identically. A Consumer specifies the ID of a Named XQuery in a Query Request to a specific Service Provider, along with a set of values for any associated script parameters[8].

The Provider may already either:

- Have its own copy of the script template corresponding to the supplied ID. In this case it proceeds to execute the script and return the results as the Response to the Query (either bulk or paged).

- Have its own logic ready to generate the expected results in the Response. When true, it allows Named XQuerys in a Request to be serviced by a Provider, which neither knows nor supports XQuery technology.

Otherwise the Named XQuery is unknown to the recipient Service Provider. In order to successfully process such a Consumer Query, the Provider must first issue its own Query by ID to this Named XQuery Registry to acquire the XQuery Template corresponding to the supplied ID. The Service Provider **should** then:

- Cache this Template (so it can be reused for later Requests)

- Execute it as an XQuery script by applying the associated parameters

---

[8] *Please refer to the XQuery section in the Base Architecture Document for further details.*

- Pack the results into the Response.

If the Provider is unable to support this logic, than a SIF Error Message **must** be returned with an HTTP Error Code of 405 (Method Not Allowed).

6.1   Security

Note: This entire section relates solely to Consumer-supplied XQuery Templates and can be ignored if all entries in the Named XQuery Registry are predefined by Site Administrators, guided in part by the collection of XQuery Templates contained in the binding of the SIF Data Model deployed at the site.

The Named XQuery Registry can optionally enforce administrator security over the formats of acceptable XQuery Scripts which may be utilized in the Environment. For example, any script in which element names are altered or element values combined and reported as the value of another element **SHOULD** be prohibited. This removes the possibility that the presence of XPath-based XML filters between Consumer and Provider in accordance with site security policies can be circumvented by the internals of an XQuery script.

6.1.1   Lazy Authorization

There may be a significant delay between the time a Consumer attempts to create a new XQuery Template in the Registry and the time when it is approved, since the XQuery Template contained in the *Create* Request may have to be reviewed by administrative personnel.

To determine whether or not the Template has been approved, the Consumer can issue a Query by ID for the Template it just created. This will contain a *status* element with one of three possible Responses:

- **Pending**. The Consumer should reissue the Query at timed intervals until one of the other statuses is returned.

- **Disallowed**. The Template was rejected. The *qualifier* element will contain the reason.

- **Approved.**   The Template has been accepted, and can be referenced in subsequent XQueries to other Service Providers

If the status is "*Pending*" the Consumer can either re-query at a fixed interval until the status changes, or alternatively (since the Status change represents a modification to the Template), the Consumer can wait for the corresponding Event for that XQuery

Template, and determine whether it reports a change from "Pending" to "Allowed" or Disallowed".

6.1.2 Security Limitations

Named XQuery are guaranteed to be secure only in their script limitations, and **NOT** in the particular instance in which they may be executed. There may be external administrator-imposed limitations on which Consumers can utilize a specific template, and limitations on the Zones in which the recipient Provider can reside.

For those dynamic aspects, the same authorization limitations that apply to a specific Consumer issuing a Paged Query to a specified Service Provider (which attempts to retrieve every element of every object) apply to any XQuery script the Consumer may issue to that Provider as well.

As a result, there is no information associated with a Named XQuery Registry entry that relates to a specific Consumer, Zone or Context.

An authorized Consumer **MAY** attempt to create Named XQuerys for later use by itself or other Consumers. Once approved, the XQuery may be utilized in Queries by any authorized Consumer with the Template ID.

Upon receiving a Create Request the Named XQuery Registry Provider sets the following items:

- **id**: **MUST** use the provided value. If it is a duplicate of an existing XQuery Template, the *create* Request **MUST** be rejected.

- **type**:   If the contained XQuery Template is limited to requesting all or a subset of defined object's elements in all or a subset of qualifying objects of a single type, the Template type **MUST** be set to SINGULAR.

- If the XQuery Template does not fit the Singular format, but is still limited to a single object type (as when it also includes requesting "calculated" values based upon existing elements (ex: "the sum of the number of students who …") the Template type **MUST** be set to FORMULA.

- Everything else (XQuery Templates containing references to objects of more than one type) **MUST** be set to EXTENDED.

- **status**: Object Providers **MUST** only execute XQueries with a status of APPROVED. This status **MAY** be assigned in a variety of ways (blind trust, manual administrator approval, or automated XQuery Template analysis).

As the Environments Provider encounters messages that reference XQuery Templates, it **MAY** enforce restrictions through ACLs. For example: allow constructs such as:

reference /sd:xquery[id="StudentsBySchoolName"]

When a Consumer invokes a Query which includes the ID of a registered XQuery Template, the Service Provider receiving it **MUST** accomplish the following.

- Resolve the Template ID to the contents of its Registry entry. For performance reasons, Providers **SHOULD** stay synchronized with the Named XQuery Registry, but **MAY** Query for the Template to resolve each arriving Query.

- Ensure the entry is of a *type* supported by this provider and its *status* is APPROVED.

- Ensure all parameter values contained in the Request have corresponding entries in the XQuery template and are valid for use.

- Substitute parameter values into the XQuery Template to make the XQuery Script

- Execute the script and verify that accurate results have been secured.

If an include statement is part of the registry entry, the provider **SHOULD** reduce elements of the resulting XML down to those specified, before sending results to the consumer. Note that especially when working with small objects; it is generally more efficient to receive the entire data object than to ask the Provider to return a partial object.

A Consumer can dynamically construct an "ad-hoc" XQuery in response to some user-selected form by creating, using, than deleting the need XQuery Template from this registry.

6.2   Restrictions on Use

6.2.1   Singular Form

This more limited single object Query operation only supports queries that align with the Singular XQuery Template capabilities defined below. Note that it is a superset of the dynamic Query "*where*" URL Query Parameter that is defined in the Basic Architecture.

- Each element in each data model reference must be restricted to the actual (qualified) name of the entity being referenced. In other words, no wildcards (//, ., .., *, @*, node()) may be used in the XPaths.

- XQuery and XPath functions **must not** be used in a Singular XQuery.

- Variables **must not** be used in a Singular XQuery.

- Singular XQueries **must** match the structure shown here (excluding whitespace).

- Transports may convey the Singular XQuery in the format most suitable for them; however that format **must** map directly to the structures defined in the following table.

  - Namespace_Declarations
  - Entity_Name[
    (Element Operator Value Inner_Boolean_Operator Element Operator Value) Outer_Boolean_Operator (...)]

Component	Examples	Comments
Namespace Declarations	declare namespace dm = "http://www.sifassociation.org/datamodel/na/3.3" declare default element namespace "http://namespaces.sifassociation.org/compliance/csq";	As individual Data Entities **may** contain multiple namespaces, multiple Namespace Declarations **must** be supported.
Entity Name	/dm:StudentPersonal /CombinedCSQ	This both indicates what Object to return as well as the root element of the XPaths located in the where clause (if any).
Where Clause	[...]	
Outer Boolean Operator	"and," "or," or omitted	May use as many as possible, they all **must** be the same.
Condition Group	(...)	
Inner Boolean Operator	"and," "or," or omitted	May use as many as possible, they all **must** be the same.
Condition	dm:Name/dm:LastName="Smith" ObjectMatrix/Object/@processesAdds = "true"	Logical expression bound by an operator.

Element	dm:Name/dm:LastName ObjectMatrix/Object/@processesAdds	XPath to the element (or attribute) being evaluated.
Operator	= < > <= >= !=	Transports **may** require these or other characters be escaped on the wire as is appropriate to the technologies employed.
Value	"Smith" "true"	XQuery requires non-numeric values to be surrounded with double quotes (and vice versa).

The Named XQuery Registry can, by confirming the exclusive presence of the following lines (in order), verify that the Singular XQuery form is being followed:

- One or more XQuery namespace declarations.

- One restricted XPath 2.0 expression + parameters.

Example:

The following example is of a Singular XQuery script, which searches for all students with a particular surname.

declare namespace dm = "http://www.sifassociation.org/datamodel/na/3.3";

/dm:Student [(dm:Name/dm:LastName="Smith") and (...)]

If the XQuery returned no qualifying objects, the Service Provider should return an HTTP error code of 404 (Not Found). Otherwise the XQuery returns one or more Student objects in the form.

<student> ... </student>

...

Before putting this result out on the wire as the Response, a Service Provider **must** "wrap" these results in a <students> element to conform to the formats returned by the other types of Query (Data Object and Service Path).

```
<students>

        <student> … </student>

        <student> … </student>

</students>
```

6.2.2 Formula Form

SIF intends a Formula to be an XQuery that does more than just retrieve data; it may do calculations and other useful transformations to greatly increase efficiency (over the wire) as long as namespace and other security rules are observed. Formally a Formula is any XQuery that does not follow the singular form, however still only targets a single data object. That said application specific function call are discouraged and any XQuery containing them **should** be rejected by the Providers registry.

6.2.3 Extended Form

Extended Queries are similar to Formulas however they may target and/or combine multiple objects.

6.2.4 Parameters

While named parameters provide the ability to pre-approve Templates where one or more values differ between Requests utilizing the same Template, Providers must be able to ensure these values specified are not more than a single value. To that end only alphanumeric values (including floating point numbers) with spaces or the empty string are allowed as parameter values. This restriction **MAY** require what is conceptually one value to be communicated through multiple parameters. For instance: "{:ns:}:{:object:}"

If parameters are provided the list **MUST** be comprehensive, such that the submitter **MUST NOT** need to also scan the script to ensure additional parameter are not available. This holds true, even if the list of parameters is empty.

6.2.5 Access and Ownership

Depending upon Named XQuery Service functionality and a Consumer's authorization rights, a Consumer may create a Template, query all Templates in the Registry by either

Paged Query or Query by ID (the template token) and delete only those Templates it has created.

6.3   Supported Operations

In most cases the contents of the XQuery Template Registry will be fixed, and only the Query operation will be supported.

Request	Direct Architecture	Brokered Architecture
Query	M	M
Create	O	O
Update	P	P
Delete	O	O
Events	O	O

There is no support for dynamic Query, and there are no defined Named XQuery Registry Service Paths or XQuery Templates which self-reference this Service.   Individual XQuery Templates may be returned if Query by ID is requested.

6.4   Named XQuery Registry Data Elements

Element	Char	Description	Type
xquerys		Collection element used where multiple XQuerys **MAY** be conveyed.	Collection
xquery	M		Member
xquery/@id	M	The unique ID of the XQuery Template, which is contained in the Query Request URL when this template is being utilized.	xs:token

		"Suggested" by the Consumer issuing the *create* Request, or pre-set in accordance with the binding requirements of a particular SIF release. This is NOT required to be a UUID, and must be provided as the Service Identifier in every Query URL which invokes this Template. Ex: "StudentSnapshot".	
xquery/type	M	Determines if the query may be sent to a data provider or if it requires the extended XQuery mechanism. Determined by the XQuery Templates Registry Service	enumeration SINGULAR FORMULA EXTENDED
xquery/status	M	Communicates if the query has been inspected for data hiding and the results of that assessment. Determined by the Named XQuery Registry Service Only APPROVED XQueries **MAY** be executed.	enumeration PENDING APPROVED DISALLOWED
xquery /qualifier	OC	If status is DISALLOWED this element optionally contains the reason. Determined by the XQuery Templates Registry Service	xs:normalizedString minLength=0 maxLength=1024
xquery /description	ON	Human readable description of the XQuery's intended behavior. Example: Query all the students in SIF University.	xs:normalizedString minLength=0 maxLength=1024
xquery/script	MN	The text of the XQuery which **MAY** accept parameters. By replacing fields surrounded by {: and :} with the value with the name of the field. XQueries are capable of hiding data. Therefore the Template **SHOULD** be scrutinized against privacy policies before being permitted to run.	xs:string minLength=0 CDATA
xquery /parameters /parameter /name	M	The name of the query parameter to be passed in. **Note:** Must not conflict with any other query parameter name.	xs:token

xquery /parameters /parameter /type	O	A well understood type that MAY be used for parameter validation. Example: xs:unsignedInt	xs:token
xquery /parameters /parameter /default	O	Value to be supplied for this parameter if the submitter omits it.	xs:token
xquery /parameters /parameter /description	O	An explanation of what this parameter represents.	xs:normalizedString minLength=0 maxLength=1024
xquery /parameters /parameter /@required	OC	Indicator as to if this parameter is required. If omitted, the parameter is assumed to be required!	xs: boolean
xquery /parameters /parameter /enumerations /enumeration /value	M	An acceptable parameter value for this query.	xs:token
xquery /parameters /parameter /minLength	O	The minimum length for values of this parameter. If omitted, zero. May be functionally different because of other restrictions.	xs:unsignedInt
xquery /parameters /parameter /maxLength	O	The maximum length for values of this parameter. If omitted, unbounded. May be functionally different because of other restrictions.	xs:unsignedInt
xquery /parameters /parameter /enumerations /enumeration	OC	If enumerated value has an equivalent code, this field may be used for conversion.	xs:token

/code			
xquery /parameters /parameter /range/start	M	If a numeric range is used instead of enumerations, this holds the inclusive starting value.	xs:long
xquery /parameters /parameter /range/end	M	If a numeric range is used instead of enumerations, this holds the inclusive ending value.	xs:long
/returnType	Q	The fully qualified name of the root element returned by this service. **Example:** http://www.sifassociation.org/datamodel/na/3.3:students	xs:anyURI

Example

In the example below the SIF 2.x Data Model was used, and the appropriate Data Model Namespace is (counter-intuitively) "infrastructure/2.x".

```xml
<xquerys xmlns="http://sifassociation.org/infrastructure/3.2.1">
  <xquery id="StudentsBySchool">
    <description>Query all the students in the specified school.</description>
    <script>
<![CDATA[
(:  Query all the students in the school (i.e. name=SIF University).  :)
declare namespace dm = "http://www.sifinfo.org/infrastructure/2.x";
for $school in /dm:SchoolInfo[dm:SchoolName = "{:name:}"]
for $enrollment in /dm:StudentSchoolEnrollment
for $student in /dm:StudentPersonal
where $school/@id = $enrollment/@SchoolInfoId and $enrollment/@StudentPersonalId =
$student/@id
return $student
]]>
    </script>
    <parameters>
      <parameter>
        <name>name</name>
        <type>xs:token</type>
        <required>yes</required>
      </parameter>
    </parameters>
    <returnType>http://www.sifinfo.org/infrastructure/2.x:studentPersonal</returnType>
  </xquery>
</xquerys>
```

7.   Alerts

There is a single Alerts Utility Service available in the Environment, located in the *environment-global* Zone. It replaces and extends the functionality provided by the LogEntry Event in SIF v2.x, providing authorized Service Consumers with the ability to:

- Log an "alert" (error, warning or status change) by creating an Alert object

- Retrieve specified collections of these Alert objects via standard query mechanisms

- Subscribe to Alert creation Events

The ability to detect when new Alerts are created could be used by an administrative Consumer to monitor the performance of newly installed applications, detect when another Consumer had indicated it was the cause of a problem, or track and flag Alerts above a pre-specified priority level. Such Alerts might include preventative maintenance warnings from a Queue Infrastructure Service that its Consumer has stopped polling for arriving messages, or that the number of messages in the Queue have crossed a predefined threshold.

Ideally an Alert should contain as much identifying information about the problem being reported as possible. However "nesting" an erroneous message inside the Alert can generate unanticipated problems if the error being reported is that the original message format was invalid. For this reason, the original message **MAY** be omitted or described and when it is included it **MUST** be properly escaped (included as CDATA).

7.1   Supported Operations

The required levels of supported Alerts Utility Service operations (both Requests and Events) available to properly authorized Consumers are indicated in the table below (where M = Mandatory, O = Optional, P = Prohibited).

Operation	Direct Architecture	Brokered Architecture
Query	O	O
Create (single object form only)[9]	O	M

---

[9] *Only one Alert object can be specified in each Alert Create Request.*

Update	P	P
Delete	P	P
Event	O	M

Support for Query is optional, and when provided for non-Administrative-level applications, returns only Alert objects created the same application that issued the Query. There is no support for dynamic Query, and there are no defined Alert Service Paths or XQuery Templates. Individual Alerts may be returned if Query by ID is requested.

7.2 Error Handling

Proper use of the Alerts Service is an essential part of the error logic handling of every Service Consumer and Provider. The following message exchange situations during which errors occur define how SIF 3 components **should** interact with the Alert Utility Service, and are applicable to all Service types (Object, Functional, and Utility).

7.2.1 A Service Provider receives a Consumer operation invocation which it rejects

Actors	Consumer which invokes a Service operation
	Functional or Data Entity Service Provider supporting that operation
	Alerts Utility Service
Preconditions	The Consumer invokes a Service operation. It is reaches the Service Provider, which rejects it. The reasons could range from:
	• Invalid XML in the Request payload
	• Omission of a mandatory element
	• Incorrect data value (ex: an Object ID which did not correlate to a stored object)
Actions	The Service Provider should then perform the following actions:
	• Return a *"NAK"* Response to the client, indicating at least the error category, code and description which identify the problem.
	• Optionally determine whether this error is a duplicate or deserves to

	generate a unique Alert (i.e. is it identical to a previous error about the same Consumer that has already been logged?). If not, ignore it (or file a different Alert).
	
	• If applicable, the reporting Service should issue a *Create Alert* Request to the Alerts Utility Service, identifying itself as the Reporter and the Consumer as the Cause, and providing as much information about the error as seems reasonable.
	
	This should indicate at least the description, error, category and code which document the problem in the Consumer Request. Along with the appropriate error category and code:
	
	   o For invalid XML, return the error reported from the parser
	
	   o For omission of a mandatory element, return the element tag name
	
	   o For an invalid data value, return the element tag name and the erroneous value
	
	• Take any other action (ex: logging the error to a local file) as seems appropriate

Post Conditions	When receiving the rejection of its Service method invocation, the Consumer should then perform only those actions which relate to its own internal logic (i.e. log the problem to a local file, report it to the user, back out of a transaction, adjust its internal data base). However the Consumer does not normally create an Alert object reporting the problem, as it can rely upon the Service Provider to do that.

7.2.2 The Consumer receives a Service Provider Response which it rejects

Actors	Consumer which invokes a Service Provider operation Functional or Object Service Provider supporting that operation Alerts Utility Service
Preconditions	The Service Provider Response to a previously invoked operation is seen and rejected by the Consumer. The reasons could range from: • Invalid XML in the Response payload • Omission of a mandatory element

	• Incorrect data value (ex: an enumerated value such as a Country Code which does not correspond to a valid entry as the Consumer understands it, in an External Code List)

Actions	The Consumer has no way to report this problem back to the Service Provider. It should then:
	
	 • Perform only those actions which relate to its own internal logic (i.e. log the problem to a local file, report it to the user, back out of a transaction, adjust its internal data base). This closely parallels the required actions in the case when the Service Provider has rejected the Consumer's Request.
	
	 • If the Consumer believes the cause of the disconnect rests with the Service Provider, it should also issue a *Create Alert* Request to the Alerts Service, identifying itself as the Reporter and the Service Provider which returned the Response as the Cause, providing as much information about the problem as seems reasonable.
	
	 This should indicate at least the description, error, category and code which documents the problem in the Response. Along with the appropriate error category and code:
	
	    o For invalid XML, return the error reported from the parser
	
	    o For omission of a mandatory element, return the element tag name
	
	    o For an invalid data value, return the element tag name and the erroneous value

Post Conditions	In rare cases, the Consumer may "re-request" the operation using alternative parameters if it has reason to anticipate that will produce better results.

7.2.3 A Service Provider posts an Event which a Subscribing Consumer rejects

Actors	Consumer Subscriber to Service Provider Events Service Provider which publishes a Change Event Alerts Utility Service
Preconditions	An Event is received and rejected by the Subscriber. The reasons could range from: • Invalid XML in the Event payload • Omission of a mandatory element (in a Create Event only)

	• Incorrect data value (ex: an enumerated value such as a Country Code which does not correspond to a valid entry as the client understands it, in an External Code List)
Actions	The Subscriber has no way to report the problem back to the Service. It must ignore the Event in terms of processing it or updating its Data Store. However it should log the problem to a local file and / or report it to its end user. There is also the possibility that an erroneous Provider is publishing a stream of faulty Events, which if left unchecked, would result in a flood of new Alert Objects from each Subscriber. As a result it is recommended that before a Subscriber creates an Alert to report an Event error, it first determines that it has never previously (or not in a pre-specified time) reported an error contained in an Event of this type. If the Subscriber believes the cause of the disconnect is a problem with the Publisher, and if it determines it is not redundant to do so, it should also issue a *Create Alert* Request to the Alert Service, identifying itself as the Reporter and the Service which published the Event as the Cause, providing as much information about the original Event as seems reasonable. This should indicate at least the description, error, category and code which documents the problem. Along with the appropriate error category and code: • For invalid XML, return the error reported from the parser • For omission of a mandatory element in a Create Event, return the element tag name • For an invalid data value, return the element tag name and the erroneous value
Post Conditions	The Subscriber should record that it has created an Alert because of an erroneous Event received for this Object type. This will help minimize the number of Alerts reporting the same problem should the cause reside with the Event Publisher.

7.2.4   A Service Provider detects an inactive Consumer

Actors	Stateful Service Provider dependent on additional Consumer operations Alerts Utility Service
Preconditions	The Service Provider detects a Consumer error because an expected Service operation has not been invoked. An example would be when a Polling Queue Instance determines that the preset

	maximum time limit to wait for the arrival of a GetNextMessage invocation has expired, indicating that the Consumer may be offline.
Actions	The Service has no opportunity to report the problem back to the Consumer (and in fact the Consumer may not even be active). The Provider should then perform the following actions: • Determine whether this error is a duplicate or deserves to generate a unique Alert (i.e. is it identical to a previous error about the same Consumer that has already been logged?). In the example given, if the error is deemed duplicative, it should be ignored. • If applicable, the reporting Service should issue a *Create Alert* Request to the Alerts Utility Service, identifying itself as the Reporter and the Consumer as the Cause, and providing as much information about the error as seems reasonable. This should indicate at least the description, error, category and code which document the "state" problem the Service is having with the Consumer When reporting "*Consumer inactivity*", along with the appropriate error category and code, the Alert object created should contain Alert Level, Error and Description element values which provide indications of: o Time of last client activity o Size (where known) of current Message Queue o Severity Level of Alert (warning that limit exceeded or application offline error) • Take any other action (ex: logging the error to a local file) as seems appropriate

7.3   Alert Object Data Elements

The following data elements comprise the payload of an Alert Object "create" request. Note that there is no "Id" attribute as that is supplied by the Alerts Utility Service Provider when the Alert Object is actually created.

Element or @attribute	Char	Description	Type
alert		Top level element of an Alert	Alert Object Root
alert@id	M	Identifier for the *alert* object. Defined by the Alerts Service when a create Alert is issued.	UUID
alert/reporter	M	External identification of the Application (Consumer or Provider) reporting this Alert. This is typically the Consumer Name of the reporting application.	xs:token
alert/cause	O	External identification of the cause of the Alert. This is typically the *sourceName* of the Partner as contained in the HTTP Header Field of the message which provoked the creation of the *alert*.	xs:token
alert/exchange	M	The exchange (or lack of exchange) responsible for generating the Alert.	xs:enumeration One of: REQUEST RESPONSE EVENT TIMEOUT
alert/level	M	The level of the Alert.	xs:enumeration One of: INFO STATECHANGE WARNING ERROR
alert/description	O	A description of the reason for the Alert	xs:normalizedString
alert/messageId	C	If available, the ID of the Message causing the problem	xs:token

alert/body	O	The internals of the offending message or a more complete description of the information or state change.	CDATA
alert/error	O	Detailed error results such as a stack trace	xs:string
alert/xpath	O	An indicator of the specific element that was in error (or contributed to the problem).	xs:selector
alert/category	MC	If the Alert Level is an error, this value must be the SIF 3.0 Error Category corresponding to the type of Error being reported. Otherwise it should be mapped to a corresponding category type where possible. If this message was persisted and processed later, the corresponding Category MAY be included here, even though the message was originally acknowledged normally.	xs:unsignedInt
alert/code	MC	If the Alert Level is an error, this value must be the SIF 3.0 Error Code[10] corresponding to the type of Error being reported. Otherwise it should be mapped to a corresponding category type where possible. If this message was persisted and processed later, the corresponding Code MAY be included here, even though the message was originally acknowledged normally.	xs:unsignedInt
alert/internal	O	Code internal to the reporter.	xs:token

---

[10] *Please refer to Section 4.5.2 of the Base Architecture document.*

7.4   XML Examples

The payload of an Alert issued by a Data Miner regarding data returned from and SIS system.

```xml
<alert id="890d7606-27bf-11e6-b67b-9e71128cae77">
  <reporter>49erDataMiner</reporter>
  <cause>RamseySIS</cause>
  <exchange>REQUEST</exchange>
  <level>ERROR</level>
  <description>Date format not understood.</description>
  <body>
<![CDATA[
<StudentSchoolEnrollment      Id="A8C3D3E34B359D75101D00AA001A1652"
StudentPersonalId="D3E34B359D75101A8C3D00AA001A1652"
SchoolInfoId="D3E34B359D75101A8C3D00AA001A1651" MembershipType="Home"
TimeFrame="Current" SchoolYear="2004">
  <EntryDate>2004-01-29</EntryDate>
  <EntryType>
    <Code>1838</Code>
  </EntryType>
  <GradeLevel>
    <Code>10</Code>
  </GradeLevel>
  <Homeroom SIF_RefObject="RoomInfo">D7510D3E34B3591A8C3D00AA001A1651</Homeroom>
  <Advisor SIF_RefObject="StaffPersonal">B359D3E34D75101A8C3D00AA001A1652</Advisor>
  <FTE>1.00</FTE>
  <FTPTStatus>FullTime</FTPTStatus>
  <ResidencyStatus>
    <Code>1653</Code>
  </ResidencyStatus>
  <NonResidentAttendReason>1658</NonResidentAttendReason>
</StudentSchoolEnrollment>
]]>
  </body>
  <error>Missing tag "ExitDate."</error>
  <xpath>/StudentSchoolEnrollment/ExitDate</xpath>
  <code>400</code>
</alert>
```