



Intelligence Community and Department of Defense Technical Specification

REST Service Encoding Specification for Content Discovery and Retrieval: Deliver

Version 2

09 May 2014

Distribution Notice:

This document has been approved for Public Release and is available for use without restriction.

Table of Contents

Chapter 1 - Introduction	1
1.1 - Purpose	1
1.2 - Scope	1
1.3 - Background	1
1.4 - Enterprise Need	2
1.5 - Audience and Applicability	3
1.6 - Conventions	3
1.6.1 - Namespaces	4
1.7 - Conformance	4
1.8 - Security	4
Chapter 2 - Service Behavior	5
2.1 - Deliver	5
Chapter 3 - Service Interfaces	6
3.1 - Deliver Function	6
3.1.1 - Preconditions	6
3.1.2 - Input	6
3.1.2.1 - HTTP Method	6
3.1.2.2 - URI Template	6
3.1.2.3 - Message Header	7
3.1.2.4 - HTTP Message Body	7
3.1.2.5 - Deliver Request Example	7
3.1.3 - Output	7
3.1.3.1 - Header Status Code	7
3.1.3.2 - HTTP Message Header	7
3.1.3.3 - HTTP Message Body	7
3.1.4 - Post-condition	8
3.1.5 - Fault Conditions	8
Chapter 4 - CDR Conformance Validation	9
Chapter 5 - Technical Guidance and Examples	10
5.1 - Disconnected, Intermittent and Low-Bandwidth (DIL) Environments	10
Appendix A - Feature Summary	11
A.1 - DELIVER Feature Comparison	11
Appendix B - Change History	12
B.1 - V2 Change Summary	12
Appendix C - Mapping to CDR-SF	13
Appendix D - Service Features	14
D.1 - Data Compression	14
D.1.1 - Preconditions	14
D.1.2 - Input	14
D.1.2.1 - HTTP Method	14
D.1.2.2 - URL Template	14
D.1.2.3 - HTTP Message Header	14
D.1.2.4 - HTTP Body	15
D.1.2.5 - Input Message Example	15
D.1.3 - Output	15
D.1.3.1 - HTTP Status Code	16

D.1.3.2 - HTTP Message Header	16
D.1.3.3 - HTTP Body	16
D.1.3.4 - Output Example	16
D.1.4 - Post-Conditions	16
D.1.5 - HTTP Fault Conditions	16
D.1.5.1 - HTTP Fault Message Example	17
D.2 - Data Prioritization	17
D.2.1 - Preconditions	18
D.2.2 - Input	18
D.2.2.1 - HTTP Method	18
D.2.2.2 - URL Template	18
D.2.2.3 - HTTP Header	18
D.2.2.4 - Body	18
D.2.2.5 - Input Message Example	18
D.2.3 - Output	18
D.2.3.1 - HTTP Status Code	19
D.2.3.2 - HTTP Header	19
D.2.3.3 - Body	19
D.2.3.4 - Output Example	20
D.2.4 - Post-Conditions	20
D.2.5 - HTTP Fault Conditions	20
D.2.5.1 - HTTP Fault Message Example	21
D.3 - Adaptive Bandwidth	21
D.3.1 - Preconditions	21
D.3.2 - Input	22
D.3.2.1 - HTTP Method	22
D.3.2.2 - URL Template	22
D.3.2.3 - HTTP Header	22
D.3.2.4 - Body	22
D.3.2.5 - Input Message Example	22
D.3.3 - Output	22
D.3.3.1 - HTTP Status Code	22
D.3.3.2 - HTTP Header	23
D.3.3.3 - Body	23
D.3.3.4 - Output Example	23
D.3.4 - Post-Conditions	23
D.3.5 - HTTP Fault Conditions	23
D.4 - Dynamic Session Management	23
D.4.1 - Preconditions	24
D.4.2 - Input	24
D.4.2.1 - HTTP Method	24
D.4.2.2 - URL Template	24
D.4.2.3 - HTTP Header	24
D.4.2.4 - Body	25
D.4.2.5 - Input Message Example	25
D.4.3 - Output	25
D.4.3.1 - HTTP Status Code	25
D.4.3.2 - HTTP Header	26
D.4.3.3 - Body	26

D.4.3.4 - Output Example	26
D.4.4 - Post-Conditions	27
D.4.5 - HTTP Fault Conditions	27
D.4.5.1 - HTTP Fault Message Example	27
Appendix E - Glossary	28
Appendix F - Bibliography	30
Appendix G - Points of Contact	32
Appendix H - IC CIO Approval Memo	33

List of Figures

Figure 1 - CDR Architecture Documents	2
---	---

List of Tables

Table 1 - Namespaces	4
Table 2 - Deliver Component Faults	8
Table 3 - Feature Summary Legend	11
Table 4 - DELIVER Feature comparison	11
Table 5 - Identifier History	12
Table 6 - Encoding Specification V2 Change Summary	12
Table 7 - Specification Framework Create Input Variables Disposition	13
Table 8 - HTTP Message Header in HTTP Request for Data Compression	15
Table 9 - Output HTTP Message Header	16
Table 10 - Output HTTP Fault Conditions	17
Table 11 - HTTP Header Definition in HTTP Request for Data Prioritization	18
Table 12 - HTTP Header Definition in HTTP Response for Data Prioritization	19
Table 13 - Data Prioritization HTTP Fault Conditions	21
Table 14 - HTTP Header Definition in HTTP Request for Adaptive Bandwidth	22
Table 15 - HTTP Header Definition in HTTP Response for Adaptive Bandwidth	23
Table 16 - HTTP Header Definition in HTTP Request for Dynamic Session Management	25
Table 17 - HTTP Header in Response Message for Initial Range Request	26
Table 18 - HTTP Header in Response Message for Subsequent Range Request	26
Table 19 - Dynamic Session Management HTTP Fault Conditions	27

List of Examples

1.1 - Notation Convention	4
3.1 - General Deliver Request	7
D.1 - HTTP GET Request with Data Compression	15
D.2 - HTTP POST Request with Data Compression	15
D.3 - HTTP Response with Data Compression	16
D.4 - Data Compression HTTP Fault Message	17
D.5 - Data Prioritization Input Message	18
D.6 - Data Prioritization Output Message	20
D.7 - HTTP Response with Data Prioritization	21
D.8 - Adaptive Bandwidth Input Message	22
D.9 - Adaptive Bandwidth Output Message	23
D.10 - Dynamic Session Management Input Message - Initial Range Request	25
D.11 - Dynamic Session Management Input Message - Subsequent Range Request	25
D.12 - Dynamic Session Management Output Message – Initial Response	26
D.13 - Dynamic Session Management Output Message – Subsequent Response	27
D.14 - Dynamic Session Management HTTP Fault Message	27

Chapter 1 - Introduction

1.1 - Purpose

The Deliver Component, as defined by the Intelligence Community/Department of Defense (IC) / (DoD) Content Discovery and Retrieval (CDR) Reference Architecture (CDR-RA), serves as the primary mechanism for providing common transport to other CDR services. This component provides common service interfaces and a behavioral model for enabling CDR consumers to channel service request messages using Disconnected, Interrupted, and Low-bandwidth (DIL) enabled message components, and to receive the response messages on behalf of CDR service consumers.

This specification defines requirements and provides guidelines for the realization of the CDR Deliver Component as a web service using REST ^[12] web service, hereafter termed a Deliver Service in this document. This specification provides the necessary information for implementing CDR- conformant Deliver Services.

1.2 - Scope

This specification is limited to the interactions that occur between an Initiating Consumer and the Deliver Service as described in the CDR-RA ^[2] and CDR Specification Framework (CDR-SF). ^[4]

This specification provides the description of the Deliver Service Behavior in terms of the message exchange patterns necessary to enable service consumers to manage the transfer of a content resource to a specified destination.

The scope of this specification is limited as follows:

- The resource being processed by the Deliver Service is not defined.
- The Deliver Service may include additional (interim) processing, including, but not limited to, scheduling, encryption, or conversion. The specification for interim processing and any associated properties is not specified.

Successful delivery of a resource payload from the Initiating Consumer to the Receiving Server is accomplished through the use of the Deliver Service which sends/transfers an information payload on behalf of the sender. For the transaction to successfully complete, the Receiving Consumer REQUIRES the Deliver Service implementation to use protocols understood by the Receiving Consumer. At this time, this specification does not address delivery reliability beyond status codes. This specification is limited to the description of the interaction between the Initiating Consumer and the Deliver Service.

1.3 - Background

This specification is a part of the set of specifications that defines the concrete, implementation-specific guidance for the services defined under the auspices of the CDR Integrated Project Team (IPT). The CDR-RA ^[2] prescribes an abstract-to-concrete model for the development of architecture elements and guidance for content discovery and retrieval. Each layer or tier of the

model is intended to provide key aspects of the overall guidance to achieve the goals and objectives for joint IC / DoD content discovery and retrieval. [Figure 1](#) below, discussed in detail within the CDR-RA,^[2] illustrates this model.

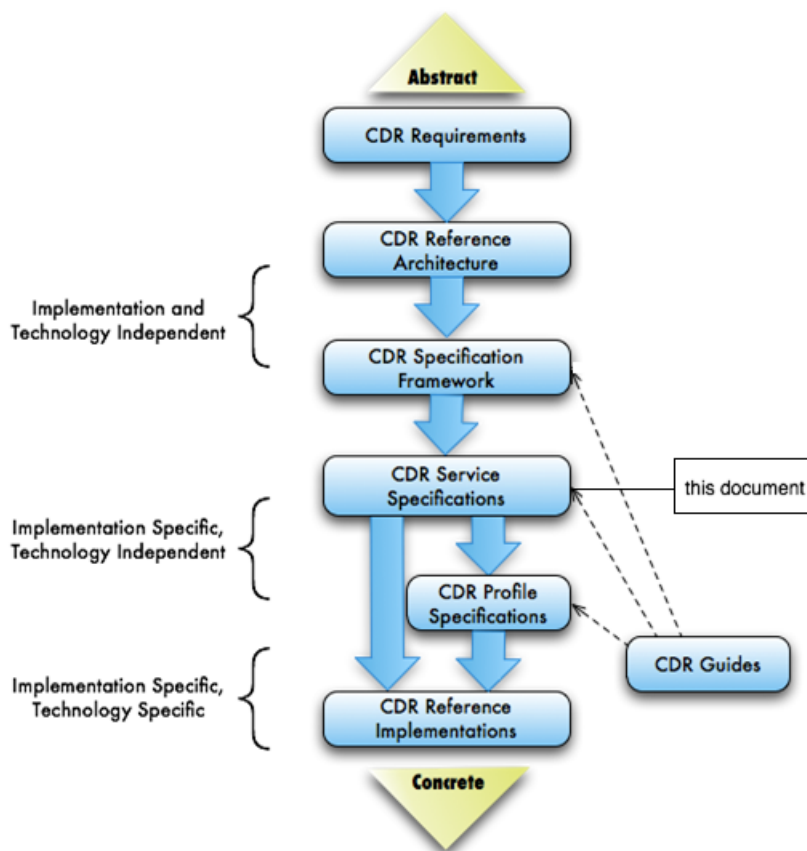


Figure 1 : CDR Architecture Documents

As illustrated, the CDR-SF derives from and describes behavior in terms of the capabilities, components, and usage patterns defined in the CDR-RA. Multiple CDR Service Specifications are derived from the CDR-SF, with separate specifications associated with the components of the architecture (e.g., Query Management) and, for each service, separate specifications to address Representational State Transfer (REST) and SOAP implementations.

This document is a specification for implementing the CDR Deliver Service as a RESTful Web Service and is intended to parallel the corresponding SOAP specification, the IC / DoD SOAP Interface Encoding Specification for CDR Deliver,^[3] as closely as possible, to minimize the difficulties in interoperating. Additional CDR Guides, Profile Specifications, or Reference Implementations may provide additional guidance on implementing this specification in a particular context.

1.4 - Enterprise Need

Enterprise needs and requirements for this specification can be found in the following policies and implementation guidance:

- IC Information Technology Enterprise (IC ITE)
 - Intelligence Community Information Technology Enterprise (IC ITE) Increment 1 Implementation Plan^[5]
- 500 Series:
 - Intelligence Community Directive (ICD) 500, Director Of National Intelligence Chief Information Officer^[6]
 - Joint IC/DoD Memorandum, IC and DoD Commitment to an Interoperable Service-Based Environment (13 Jul 07)^[11]

1.5 - Audience and Applicability

Within the IC, the conditions of use and applicability of this technical specification are defined outside of this technical specification. IC Standard (ICS) 500-20, Intelligence Community Enterprise Standards Compliance, defines the IC Enterprise Standards Baseline (IC ESB) and the applicability of such to an IC element. The IC ESB defines the compliance requirements associated with each version of a technical specification. Each version will be individually registered in the IC ESB. The IC ESB will define, among other things, the location(s) of the relevant artifacts, prescriptive status, and validity period, all of which characterize the version and its utility. Additional applicability and guidance may be defined in separate IC policy guidance.

Within the DoD , the conditions of use and applicability of this technical specification are defined outside of this technical specification, and can be located within the DoD Information Technology Standards Profile Registry (DISR).

1.6 - Conventions

The key words "MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this specification are to be interpreted as described in the IETF RFC 2119.^[9] When these words are not capitalized, they are meant in their natural-language sense.

When describing concrete eXtensible Markup Language (XML) schemas and example XML documents, this specification uses XPath as the notational convention. Each member of an XML schema is described using an XPath notation (e.g., /x:RootElement/x:ChildElement/@Attribute). The use of {any} indicates the presence of an element wildcard (<xs:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xs:anyAttribute>).

A parameter contained in curly brace, generally represented in the form {name}, is meant to be replaced with an actual value determined at run-time. An optional parameter in a URL template is one whose name is followed by ?, e.g., {name?} and these MAY be replaced by an empty string.

[Example 1.1, "Notation Convention"](#) illustrates syntax examples that in this text are distinguished by a blue border. These are meant to be illustrative and represent one way that the described syntax can be used.

Example 1.1. Notation Convention

```
<atom:entry>
  <atom:title>This is an example.</atom:title>
</atom:entry>
```

1.6.1 - Namespaces

Namespaces referenced in this document and the prefixes used to represent them are listed in [Table 1](#) . The namespace prefix of any XML Qualified Name (QName) used in any example in this document should be interpreted using the information in [Table 1](#) .

Table 1 - Namespaces

Prefix	URI	Description
cdrd	urn:cdr:deliver:2	CDR Deliver at the indicated version
xs	http://www.w3.org/2001/XMLSchema	XML Schema
atom	http://www.w3.org/2005/Atom	Atom Syndication Format [10]

Many of the examples will include an entry such as <atom:entry xmlns...> to indicate that the full XML would include the appropriate namespace declarations but the full declarations have not been included as part of the example for brevity and ease of maintaining this specification. Any use of namespaces should be interpreted as defined in [Table 1](#) . The use of elements from the atom namespace is consistent with the Atom Syndication Format.

1.7 - Conformance

This specification defines an interface to a Deliver Service to which an implementation and a subsequent deployment MUST conform. A deployment is an instance of an implementation. For an implementation to conform to the requirements contained in this specification, it MUST adhere to all mandatory aspects of the specification.

1.8 - Security

In this specification, it is assumed the originator is authorized to execute the Deliver Service, and the destination system is authorized to receive the content. It is understood that each resource MAY have associated policies for use, especially as applies to authentication and authorization and these policies MAY be asserted by the resource owner, resource destination and those responsible for governance and management of the enterprise.

This specification does not directly address those security concerns. It will be necessary for any implementation of this specification to address security concerns relevant to the systems with which the implementation interacts and to the corresponding governance bodies. Several aspects of Deliver should be addressed in the detailed security plan of an implementation, but are out of scope for this document.

Chapter 2 - Service Behavior

As defined in the CDR-SF, [\[4\]](#) Deliver behavior is realized through a single activity and is accessed through the use of the Deliver Function interface.

2.1 - Deliver

The Initiating Consumer supplies the information content via the payload parameter to be delivered to the Receiving Consumer, as specified via the destination parameter. The properties included as part of the Deliver Service request may be used to provide delivery specific information, including, but not limited to, interim processing, routing, and security. In the event that a particular implementation of the Deliver Service makes use of “default” values for message retrieval and/or delivery, the service implementer is responsible for publishing this information using an agreed upon mechanism. Cases where information is not supplied as part of the Deliver Service request and service defaults are not available will result in a fault condition.

Chapter 3 - Service Interfaces

The service interface contains the technical descriptions ¹ of functions through which the consumer will interact with the service. Support for input and output parameters for each function is described in the following tables in terms of what is expected of the Deliver Service and what is expected in terms of a consumer interacting with the service.

3.1 - Deliver Function

A Deliver Service **MUST** implement the Deliver Function as defined in this section.

3.1.1 - Preconditions

The following precondition **MUST** be satisfied if the Deliver Function is to correctly process input and generate results and post-conditions as described:

- The requester is authenticated and authorized according to applicable policy requirements for the Deliver Function implementation.
- The Recipient is authorized to accept delivery of the resource.
- A payload exists and is accessible for delivery.

3.1.2 - Input

The input **SHOULD** be directed to the endpoint address identified by the implementer.

3.1.2.1 - HTTP Method

The Deliver Function **MUST** use the HTTP POST method.

3.1.2.2 - URI Template

The URI used to access the Deliver function **MUST** conform to the following: ²

```
http://{anyAuthority}/{anyHierarchy}/deliver?  
deliverTo={DeliverTo}&{DeliverProperties}
```

where

{DeliverTo} **MUST** be one or more URLs defining the target of the Deliver resource payload.

{DeliverProperties} **MAY** include additional parameters that support configuring optional behavior specific to use for Deliver.

¹The Deliver Service is intended to conform as described by the Deliver Component section of the CDR-SF. [\[4\]](#)

²For example, <http://www.cdr.org/templates/examples/deliver?deliverTo=http%3A%2F%2Fagency.gov%2Freceive> is a conforming URL .

3.1.2.3 - Message Header

- The Header MUST include Content-Type and Content-Length.
- The Header MAY contain additional fields such as those specified below in the Service Features appendix.

3.1.2.4 - HTTP Message Body

The body of the Deliver request that MUST contain a resource payload that, if provided, corresponds to the Content-Type, as defined in Section 3.1.2.3.

3.1.2.5 - Deliver Request Example

[Example 3.1, "General Deliver Request"](#) shows an example with {Resource Payload} which can contain any payload. Specifications / profiles based on this format may replace the notation as appropriate.

Example 3.1. General Deliver Request

```
POST /deliver?deliverTo=http%3A%2F%2Fagency.gov%2Freceive HTTP/1.1
Host: CDR.org
Content-Type: application/xml
Content-Length: 9236
{ResourcePayload}
```

3.1.3 - Output

3.1.3.1 - Header Status Code

If the POST is successful, the service will respond with a '200 OK' status code. For requests that result in an error, an HTTP error code MUST be returned. Fault codes are discussed in Section 2.

3.1.3.2 - HTTP Message Header

If an HTTP Message Body is returned, the output SHOULD contain the following:

- The Header SHOULD include Content-Type and Content-Length.
- The Header MAY contain additional fields such as those specified below in the Service Features appendix.

3.1.3.3 - HTTP Message Body

Although there are no required outputs, a specific Deliver Service implementation may choose to return delivery-specific information (such as a reference to the Deliver status).

3.1.4 - Post-condition

The following condition **MUST** be met upon completion of Deliver:

1. The use of this function has been audited according to applicable policy.³

3.1.5 - Fault Conditions

Fault conditions as described in [Table 2](#) provide a brief description of faults that may occur during Deliver Component processing. Individual specifications **SHOULD** support these faults and **MAY** expand their description of fault conditions to address additional situations or provide additional detail on the fault.

Table 2 - Deliver Component Faults

Fault	Description	Error
Security Fault	A fault used if (1) the consumer is not authenticated or is not authorized to use the Deliver function or (2) the Recipient cannot be authenticated or is not authorized to accept the resource.	403
Deliver Component Unknown	A fault used if Recipient Address is not specified and there is no default defined.	400
Deliver Service Not Found	A fault used if the Recipient Address cannot be reached.	404
Incompatible Deliver Protocol	A fault used if there is a protocol mismatch.	400
Unsupported Properties Fault	A fault used if a property of the Deliver Properties is not supported. This fault will terminate all processing.	400
Properties Error Fault	A fault used if an element of the properties is supported but contains an error in its representation. The fault will terminate execution without proceeding.	400
Deliver Component Unavailable	A fault is used to reflect a general server side error	500

³The use of this function may be audited according to applicable policy and may include auditing of the success or failure of the function.

Chapter 4 - CDR Conformance Validation

Conformance is defined herein as “adherence to all relevant organizationally-mandated standards within a related family of specifications such that all implementations of web services based on these standards will achieve measurable levels of increased interoperability with others implementing the same standards.” Conformance does not necessarily imply compliance. Conformance addresses specific technical interoperability while compliance is associated with standards policy adherence in its entirety. While not sufficient by itself, technical conformance is a necessary condition that helps in validating compliance within specifications.

Conformance of an implemented service to this specification will be validated by using certified tests or other means that are approved by the relevant authority by which this standard is mandated. In the absence of a certified test to validate the technical conformance, documentation that enables unambiguous traceability from each specification requirement to satisfactory execution of that requirement in the implementation must be provided. This is often in the form of a mapping matrix called a Conformance Test Matrix (CTM).

Note: The ability to validate conformance is greatly enhanced by articulating individual specification requirements during specification authoring, and for developing the associated implementation tests.

Chapter 5 - Technical Guidance and Examples

5.1 - Disconnected, Intermittent and Low-Bandwidth (DIL) Environments

Disconnected, Intermittent and Low-Bandwidth (DIL) environments are those environments at the tactical edge that may not have the infrastructure necessary for full service transactions. To better perform in a DIL environment, a Deliver Service MAY make use of the following features described in [Appendix D - Service Features](#) :

- Data Compression
- Data Prioritization
- Adaptive Bandwidth

Appendix A Feature Summary

Section A.1, “[DELIVER Feature Comparison](#)” summarizes major features by version for DELIVER and all dependent specs. The “Required date” is the date when systems should support a feature based on the specified driver. Executive Orders, ISOO notices, ICDs and other policy documents have a variety of effective dates.

Table 3 - Feature Summary Legend

Key	Description
F	Full (able to comply and verified by spec to some degree)
P	Partial (Able to comply but not verifiable)
N	Non-compliance (Can’t comply)
N/A	Not Applicable. Feature is no longer required.
Cell Colors represent the same information as the Key value	

A.1. DELIVER Feature Comparison

Table 4 - DELIVER Feature comparison

DELIVER Feature Comparison			
Required date	Feature	V1	V2
	Data Compression	P	F
	Data Prioritization	P	F
	Adaptive Bandwidth	P	F
	Dynamic Session Management	P	F

Appendix B Change History

[Table 5](#) summarizes the version identifier history for this technical specification.

Table 5 - Identifier History

Version	Date	Purpose
0.1	13 September 2010	Initial draft for subgroup review.
0.2	12 October 2010	Revised following subgroup review, and again following the 5 October CDR - IPT Design Meeting.
0.3	18 October 2010	Revised following subgroup review.
1.0	20 October 2010	Revised following subgroup review.
2.0	14 March 2014	Added DIL features to specification. For details of changes, see Section B.1 - V2 Change Summary

B.1 - V2 Change Summary

Significant drivers for Version 2 include:

DIL Features(see Appendix D Service Features):

- Data Compression
- Data Prioritization
- Adaptive Bandwidth
- Dynamic Session Management

[Table 6](#) summarizes the change made to V1 in developing V2.

Table 6 - Encoding Specification V2 Change Summary

Change	Artifacts changed	Compatibility Notes
Add DIL features	REST Service Encoding Specification	

Appendix C Mapping to CDR-SF

This section explicitly ties the items in this specification to the requirements of the CDR-SF. [\[4\]](#) The CDR-SF identifies the requirements for creating specifications, while implementation details are outlined in this document.

The IC / DoD CDR-SF defines a number of inputs to the Search operation. [Table 7](#) maps the CDR-SF inputs and outputs, respectively, to parameters, elements, and attributes defined in this specification.

Table 7 - Specification Framework Create Input Variables Disposition

Specification Framework Variables	REST Deliver Specification
Deliver Properties	DeliverTo {DeliverProperties}
Resource Payload	{ResourcePayload}

Definition and use of Deliver Properties MAY be supported by an implementation; if supported, the properties MAY be included by the consumer in the input. An implementation SHOULD ignore properties it does not support. Additional values enabling more selective output or additional Deliver Properties may be defined in future versions of this specification.

Appendix D Service Features

The features defined in this section may be applied to a function to provide enhanced functionality used for specific implementation concerns such as mitigating DIL network constraints. This section describes each feature and how the behavior of the function to which the feature is applied is augmented.

D.1 - Data Compression

The Data Compression is used to exchange message compression parameters and compressed content between a client and a server in a request and response transaction.

When a client sends an HTTP GET request, the client MUST send the HTTP header Accept-Encoding to specify requested compression algorithm(s) (e.g., gzip). If the server supports the desired compression algorithm(s), the server MUST return the HTTP response with the Content-Encoding HTTP header specifying the actually used compression algorithm and the compressed HTTP body. If the server does not support the requested compression algorithm(s), the server MUST return an HTTP error message with the error code 415 to the client indicating unsupported format(s).

When a client sends an HTTP POST request, the client MUST send the HTTP header Content-Encoding to indicate the body of the request is compressed and specify the type of compression algorithm used. If the server does not support the compression algorithm used by the client to compress the request body, the server MUST return an HTTP error message with the error code 415 to the client indicating an unsupported format. The client MAY send the HTTP header Accept-Encoding to specify requested compression algorithm(s) (e.g., gzip). If the server supports the desired compression algorithm(s), the server MUST return the HTTP response with the Content-Encoding HTTP header specifying the used compression algorithm and the compressed HTTP body. If the server does not support the requested compression algorithm(s), the server MUST return an HTTP error message with the error code 415 to the client indicating unsupported format(s).

D.1.1 - Preconditions

No additional preconditions.

D.1.2 - Input

D.1.2.1 - HTTP Method

The Data Compression feature MAY be applied to the HTTP GET and POST methods.

D.1.2.2 - URL Template

No Change.

D.1.2.3 - HTTP Message Header

When used with the HTTP GET method, the HTTP request for Data Compression MUST contain the Accept-Encoding HTTP header as defined in Table 3. When used with the HTTP

POST method, the HTTP request for Data Compression MUST contain the Content_Encoding HTTP header and MAY contain the Accept_Encoding HTTP header as defined in [Table 8](#).

Table 8 - HTTP Message Header in HTTP Request for Data Compression

Header Element and Description	Support
Accept_Encoding A list of compression formats that the sender accepts. Each compression format is specified as String type. Two or more compression formats are separated by comma(s).	MUST be supported by service for HTTP GET method; MAY be supported by service for HTTP POST method.
Content_Encoding The Compression format specified as String type that is used in the message body.	MUST be supported by service for HTTP POST method.

D.1.2.4 - HTTP Body

The body of the HTTP input message contains the compressed service request if and only if the Content_Encoding header is included in the HTTP header of the input message.

D.1.2.5 - Input Message Example

Example D.1. HTTP GET Request with Data Compression

```
GET /search?q=WMD+Iraq&StartIndex=0&count=20 HTTP/1.1
Host: CDR.org
Content-Type: application/xml
Content-Length: 2516
Content-Encoding: gzip
{compressed payload}
```

Example D.2. HTTP POST Request with Data Compression

```
POST /deliver?deliverTo=http%3A%2F%2Fagency.gov%2Freceive HTTP/1.1
Host: CDR.org
Content-Type: application/xml
Content-Length: 2516
Content-Encoding: gzip
{compressed payload}
```

D.1.3 - Output

The body of the HTTP response message is compressed.

D.1.3.1 - HTTP Status Code

No change.

D.1.3.2 - HTTP Message Header

Table 9 - Output HTTP Message Header

Header Element and Description	Support
Content_Encoding The Compression format specified as String type that is used in the response message body. It MUST be one of the Compression formats that the sender accepts.	MUST be supported by service.

D.1.3.3 - HTTP Body

The body of the output message contains the compressed content of the body of the applied function.

D.1.3.4 - Output Example

Example D.3. HTTP Response with Data Compression

```

HTTP/1.1 200 OK
Content-Type: application/atom+xml
Content-Encoding: gzip
Content-Length: 4595
M86-T:79A=&EO;BTQ+C$N:F%R"F%N=&QR+3(N-RXW+FIA<@IA>&EO;2UA<&DM
M,2XR+CDN:F%R"F%X:6]M+61O;2TQ+C(N.2YJ87(*87AI;VTM:6UP;"TQ+C(N
M.2YJ87(*87AI<S(M861B+3$N-2XR+FIA<@IA>&ES,BUA9&(M8V]D96=E;BTQ
M+C4N,BYJ87(*87AI<S(M86YT+7!L=6=I;BTQ+C4N,BYJ87(*87AI<S(M8VQU
M87(*87AI<S(M;65T861A=&$M,2XU+C(N:F%R"F%X:7,R+6UT;VUP;VQI8WDM
M,2XU+C(N:F%R"F%X:7,R+7-A86HM,2XU+C(N:F%R"F%X:7,R+7-O87!
M;VYI

```

D.1.4 - Post-Conditions

The following condition MUST be met upon successful completion of a function using Data Compression:

- The result returned to the requester shall be a compressed HTTP message or an empty HTTP fault message.

D.1.5 - HTTP Fault Conditions

An implementation of a function using Data Compression MUST return the appropriate HTTP Status Code (based on values from the HTTP Status Code Registry maintained by IANA) according to the fault conditions defined in the CDR-SF. [\[4\]](#)

Table 10 maps the CDR-SF ^[4] fault conditions to the HTTP status that MUST be returned for each fault.

Table 10 - Output HTTP Fault Conditions

CDR-SF Fault Condition	HTTP Status	HTTP Description
Unsupported Compression Format	415	Unsupported Media Type.

D.1.5.1 - HTTP Fault Message Example

Example D.4. Data Compression HTTP Fault Message

HTTP Error 415: Unsupported Media Type

D.2 - Data Prioritization

Data Prioritization is used to represent the priority of a request message sent by a client to a server and the corresponding response message from the server. A client MUST send the HTTP header priority to specify the priority of the request. The response message from the server MUST include the HTTP header priority with the same priority as the request message. The HTTP header priority is an optional header. If a server does not support the HTTP header priority, the server MUST ignore the header and return a response without including the header.

The priority field in Data Prioritization is patterned after the priority field defined in OASIS Advanced Message Queuing Protocol.^[1] The priority value MUST be an integer in the range [0...9] and the value indicates the relative message priority. Higher priority numbers indicate higher priority messages. Messages with higher priorities MAY be delivered before those with lower priorities.

Although 10 priority values are defined, a server MAY implement less than 10 distinct priority levels. If n (less than 10) distinctive priority levels are implemented by a server, priorities 0 to $(5 - \text{ceiling}(n/2))$ MUST be treated equivalently and MUST be the lowest effective priority. The priorities $(4 + \text{floor}(n/2))$ and above MUST be treated equivalently and MUST be the highest effective priority. The priorities $(5 - \text{ceiling}(n/2))$ to $(4 + \text{floor}(n/2))$ inclusive MUST be treated as distinct priorities.

For example, if two distinct priority levels are implemented, then levels 0 to 4 are equivalent, levels 5 to 9 are equivalent and levels 4 and 5 are distinct. If three distinct priority levels are implemented, then levels 0 to 3 are equivalent, levels 5 to 9 are equivalent and levels 3, 4 and 5 are distinct. This scheme ensures that if two priority levels are distinct for a server which implements m separate priority levels, they are also distinct for a server which implements n different priority levels where $n > m$.

Other specific rules that further define the effects of assigning a particular value or the details of behavior for specific priority values are outside the scope of this specification.

The Data Prioritization is most useful for situations with resource shortage. For example, when there is limited bandwidth between the client and the server, a service MAY transport high priority messages before low priority messages.

If the priority value set by a client is not within the valid range, the server **MUST** return an HTTP error message (see Table 12 Data Prioritization HTTP Fault Conditions) which correspond with CDR-SF fault conditions to indicate an unrecognized priority level.

D.2.1 - Preconditions

No additional preconditions.

D.2.2 - Input

D.2.2.1 - HTTP Method

The Data Prioritization feature **MAY** be applied to the HTTP GET and POST methods.

D.2.2.2 - URL Template

No Change.

D.2.2.3 - HTTP Header

Table 11 - HTTP Header Definition in HTTP Request for Data Prioritization

Element Name and Description	Support
Priority Priority of the request message specified as Integer type. The priority MUST be from 0 to 9 with 9 the highest priority and 0 the lowest priority.	MUST be supported by service.

D.2.2.4 - Body

No change.

D.2.2.5 - Input Message Example

Example D.5. Data Prioritization Input Message

```
GET /Search?q=WMD+Iraq&StartIndex=0&count=20 HTTP/1.1
Host:www.example.com
Priority: 5
```

D.2.3 - Output

The output of a function using Data Prioritization is a REST response message with the corresponding priority in the HTTP header.

D.2.3.1 - HTTP Status Code

No change.

D.2.3.2 - HTTP Header

Table 12 - HTTP Header Definition in HTTP Response for Data Prioritization

Element Name and Description	Support
Priority Priority of the response message specified as Integer type. The priority MUST be from 0 to 9 with 9 the highest priority and 0 the lowest priority. The priority in the response message MUST match that in the request message.	MUST be supported by service.

D.2.3.3 - Body

No change.

D.2.3.4 - Output Example

Example D.6. Data Prioritization Output Message

```

HTTP/1.1 200 OK
Content-Type: application/atom+xml
Content-Length: 4595
Priority: 5

<atom:feed>
  <atom:id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</atom:id>
  <atom:title>Query Results for WMD+Iraq</atom:title>
  <atom:updated>2003-12-13T18:30:02Z</atom:updated>
  <atom:author><atom:name>Example Catalog</atom:name></atom:author>
  <opensearch:totalResults>492420</opensearch:totalResults>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>20</opensearch:itemsPerPage>
  <atom:link rel="self"
    href="http://www.example.com/Search?q=IMD
+Iraq&StartIndex=0&count=20
    type="application/atom+xml"/>
  <atom:entry>
    <atom:id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af7</atom:id>
    <atom:title>Iraq IMD</atom:title>
    <atom:updated>2011-02-21T00:00:00Z</atom:updated>
    <atom:link rel="alternate"
      href="http://www.example.com/report/iraq-2003/index.html"/>
    <relevance:score>0.67</relevance:score>
  </atom:entry>
  ...
</atom:feed>

```

D.2.4 - Post-Conditions

The following condition **MUST** be met upon completion of a function using Data Prioritization:

- The result returned to the requester shall be an HTTP response message with a Priority header or an empty HTTP fault message.

D.2.5 - HTTP Fault Conditions

An implementation of a function using Data Prioritization **MUST** return the appropriate HTTP Status Code (based on values from the HTTP Status Code Registry maintained by IANA) according to the fault conditions defined in the CDR-SF. ^[4]

[Table 13](#) maps the CDR-SF ^[4] fault conditions to the HTTP status that **SHOULD** be returned for each fault.

Table 13 - Data Prioritization HTTP Fault Conditions

CDR-SF Fault Condition	HTTP Status	HTTP Description
Unrecognized Priority Level	400	Bad Request

D.2.5.1 - HTTP Fault Message Example

Example D.7. HTTP Response with Data Prioritization

```
HTTP Error 400: Bad Request
Unrecognized Priority Level
```

D.3 - Adaptive Bandwidth

Adaptive Bandwidth is used to exchange the available bandwidth information between a client and a server in a request and response transaction. A client **MUST** send the HTTP header Bandwidth to specify the available bandwidth from the client for receiving response from a server. To satisfy the client's need in a timely manner, when forming the response message, the server **SHOULD** construct a response with content feasible to be delivered to the client based on the client's available bandwidth.

For example, a client on a handheld device may request an image. The handheld device can only support limited displaying capability due to its size, processing power and limited bandwidth to the wireless network. Therefore, although the original image has a large size and high resolution, the server could resize the image to a smaller footprint and compress it to a lower resolution based on the bandwidth constraint and relevant content adaption rules. In another example, in response to a service request, the server may decide to return a minimal response with only the required fields but not any optional fields based on the bandwidth constraint and relevant content adaption rules.

Specific content adaption rules making a response feasible for delivery using the available bandwidth are outside the scope of this specification.

The server **MAY** include the HTTP header Bandwidth in the response message. If the server is able to support the request bandwidth, it will return the message with the same bandwidth information. Otherwise, it may lower the bandwidth and include the reduced bandwidth in the response message. The HTTP header Bandwidth is an optional header. If a server does not support the HTTP header Bandwidth, the server **MUST** ignore the header and return a response without including the header.

D.3.1 - Preconditions

No additional preconditions.

D.3.2 - Input

D.3.2.1 - HTTP Method

The Adaptive Bandwidth feature MAY be applied to the HTTP GET method.

D.3.2.2 - URL Template

No Change.

D.3.2.3 - HTTP Header

Table 14 - HTTP Header Definition in HTTP Request for Adaptive Bandwidth

Element Name and Description	Support
Bandwidth The network bandwidth in bits per second that is available at the client. The bandwidth is specified as Integer type.	MUST be supported by service.

D.3.2.4 - Body

No change.

D.3.2.5 - Input Message Example

Example D.8. Adaptive Bandwidth Input Message

```
GET /1225c695-cfb8-4ebb-aaaa-6fda344efa6a HTTP/1.1
Host: www.example.com
Bandwidth:9600
```

D.3.3 - Output

The output of a function using Adaptive Bandwidth is an HTTP response message formed by considering the available bandwidth information from the client.

D.3.3.1 - HTTP Status Code

No change.

D.3.3.2 - HTTP Header

Table 15 - HTTP Header Definition in HTTP Response for Adaptive Bandwidth

Element Name and Description	Support
Bandwidth The network bandwidth in bits per second that is used by the server to construct the response. The bandwidth MUST be either similar to or smaller than the bandwidth available at the client. The bandwidth is specified as Integer type.	MAY be supported by service.

D.3.3.3 - Body

No change.

D.3.3.4 - Output Example

Example D.9. Adaptive Bandwidth Output Message

```
HTTP/1.1 200 OK
Content-Type: image/jpegContent-Length: 16392
Bandwidth: 9600

IkdpmUgbWUgYSBsZXZlciBsb25nIGVub3VnaCBhbmQgYSBmdWxjcnVtI
G9uIHdoaWN0IHRvIHBSYWNlIGl0LCBhbmQgSSBzaGFsbCBtb3ZlIHRoZS
B3b3JsZC4iIC0gIEFyY2hpbWVkZXM=
```

D.3.4 - Post-Conditions

The following condition **MUST** be met upon completion of a function using Adaptive Bandwidth:

- The result returned to the requester shall be an HTTP response message.

D.3.5 - HTTP Fault Conditions

There is no fault condition for this feature.

D.4 - Dynamic Session Management

Dynamic Session Management is used to maintain the session information in a long lasting transaction in support of the stateful self-healing, i.e., after a failure due to the loss of network connection or other reasons, the session between a client and a server is resumed and the server starts transmitting the data starting after last piece of data received by the client. As such, there is no need for the server to resend the data from the very beginning.

A client **MUST** first send the HTTP header **RANGE** with empty range parameter to detect if a server supports the **RANGE** header and to query the size of the requested resource in bytes

from the server. If the server does not support the RANGE header, the server SHOULD return the ACCEPT-RANGES header with the value none . If the server supports the RANGE header, the server SHOULD return the ACCEPT-RANGES header in range unit (i.e., bytes) and SHOULD return the CONTENT-LENGTH header with the size of the resource.

After confirming that the server supports the RANGE header, the client MAY send multiple HTTP requests to get the entire resource in multiple pieces. Each HTTP request SHOULD have the HTTP header RANGE to specify the size of the partial resource to be retrieved. After receiving an HTTP request with the HTTP header RANGE, the server MUST return the status code 216 with the CONTENT-RANGE Header specifying the range of the partial resource returned and the DATE header specifying when the response is returned. The client MUST reassemble a resource after receiving all pieces of the resource. How to break a resource into multiple pieces of contents is implementation specific and out of scope of this specification. For example, the criteria for breaking a resource into multiple pieces could be determined based on run-time parameters, such as the available bandwidth information.

If a session is interrupted due to a failure, such as the loss of network connection, the client could resume the retrieval of the resource by requesting the next range of the partial resource right after the last piece of resource received. Therefore, there is no need for the client to request the entire resource from the server.

If none of the specified range in a RANGE header overlap the current extent of the selected resource (for byte-ranges, this means that the first-byte-position were greater than the current length of the selected resource), a server SHOULD return an error message with the status code 416 to indicate unsatisfied requested range. When this 416 error message is returned for a byte-range request, the response SHOULD include a CONTENT-RANGE header specifying the current length of the selected resource.

D.4.1 - Preconditions

No additional preconditions.

D.4.2 - Input

D.4.2.1 - HTTP Method

The Dynamic Session Management feature MAY be applied to the HTTP GET method.

D.4.2.2 - URL Template

No Change.

D.4.2.3 - HTTP Header

The initial request MUST include the empty RANGE HTTP header. A subsequent request MUST include the RANGE HTTP header with a specified range.

Table 16 - HTTP Header Definition in HTTP Request for Dynamic Session Management

Element Name and Description	Support
Range The range of the content in the requested resource specified as Starting Bytes – Ending Bytes. The Ending Bytes MAY be unspecified. The Range MAY also be empty.	MUST be supported by service.

D.4.2.4 - Body

No change.

D.4.2.5 - Input Message Example

Example D.10. Dynamic Session Management Input Message - Initial Range Request

```
GET /1225c695-cfb8-4ebb-aaaa-6fda344efa6a HTTP/1.1
Host: www.example.com
Range:
```

Example D.11. Dynamic Session Management Input Message - Subsequent Range Request

```
GET /1225c695-cfb8-4ebb-aaaa-6fda344efa6a HTTP/1.1
Host: www.example.com
Range: bytes=300-599
```

D.4.3 - Output

D.4.3.1 - HTTP Status Code

If the initial GET request with empty Range header is successful, the function MUST respond with a '200 OK' Status Code. If the subsequent GET request for specific Range is successful, the function MUST respond with a '206 Partial Content' Status Code.

D.4.3.2 - HTTP Header

Table 17 - HTTP Header in Response Message for Initial Range Request

Element Name and Description	Support
Accept-Ranges A String defines the range unit, i.e., bytes.	MUST be supported by service.

Table 18 - HTTP Header in Response Message for Subsequent Range Request

Element Name and Description	Support
Accept-Ranges A String defines the range unit, i.e., bytes.	MUST be supported by service.
Content-Range The range of the content included in the response message. The Range is specified as String type.	MUST be supported by service.
Date Timestamp specified as xs:dateTime type that identifies when the response is returned.	MUST be supported by service.

D.4.3.3 - Body

The body of the response message for initial Range request MUST be empty. The body of the response message for subsequent Range request MUST include the partial resource specified in the Range request.

D.4.3.4 - Output Example

Example D.12. Dynamic Session Management Output Message – Initial Response

```
HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Length: 3000
Accept-Ranges: bytes
```

Example D.13. Dynamic Session Management Output Message – Subsequent Response

```
HTTP/1.1 206 Partial Content
Content-Length: 300
Content-Type: application/pdf
Accept-Ranges: bytes
Content-Range: bytes 300-599/3000
Date: Wed,14 Dec 2011 16:11:26 GMT

IkdpmUgbWUgYSBsZXZlciBsb25nIGVub3VnaCBhbmQgYSBmdWxjcnVtI
G9uIHdoaWN0IHRvIHBSYWNlIGl0LCBhbmQgSSBzaGFsbCBtb3ZlIHRob3R
B3b3JsZC4iIC0gIEFyY2hpbWVhZC4iXQ==
```

D.4.4 - Post-Conditions

The following condition **MUST** be met upon completion of a function using Dynamic Session Management:

- The result returned to the requester shall be an HTTP response message or an HTTP fault message.

D.4.5 - HTTP Fault Conditions

An implementation of a function using Dynamic Session Management **MUST** return the appropriate HTTP Status Code (based on values from the HTTP Status Code Registry maintained by IANA) according to the fault conditions defined in the CDR-SF [\[4\]](#)

[Table 19](#) maps the CDR-SF [\[4\]](#) fault conditions to the HTTP status that **SHOULD** be returned for each fault.

Table 19 - Dynamic Session Management HTTP Fault Conditions

CDR-SF Fault Condition	HTTP Status	HTTP Description
Unsatisfied Requested Range	416	Requested Range Not Satisfiable

D.4.5.1 - HTTP Fault Message Example

Example D.14. Dynamic Session Management HTTP Fault Message

```
HTTP Error 416: Requested Range Not Satisfiable
Content-Range: bytes 0-2999/3000
```

Appendix E Glossary

This appendix lists all the acronyms and abbreviations referenced in this encoding specification.

CDR	Content Discovery and Retrieval
CDR-RA	Content Discovery & Retrieval - Reference Architecture
CDR-SF	Content Discovery & Retrieval - Specification Framework
CIO	Chief Information Officer
CTM	Conformance Test Matrix
CVE	Controlled Vocabulary Enumeration
DIL	Disconnected, Intermittent, and Low-bandwidth
DISR	DoD Information Technology Standards Registry
DNI	Director of National Intelligence
DOD	Department of Defense
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IC	Intelligence Community
IC CIO	Intelligence Community Chief Information Officer
IC ESB	Intelligence Community Enterprise Standards Baseline
IC ITE	IC Information Technology Enterprise
ICD	Intelligence Community Directive
ICS	Intelligence Community Standard
IETF	Internet Engineering Task Force
IPT	Integrated Project Team
ISOO	Information Security Oversight Office
OCIO	Office of the Intelligence Community Chief Information Officer
REST	Representational State Transfer
RFC	Request for Comments
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Appendix F Bibliography

Bibliography

[1] AMQP

Organization for the Advancement of Structured Information Standards. *OASIS Advanced Message Queuing Protocol (AMQP)*.

Available online at: <http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf>

[2] CDR-RA

Intelligence Community/Department of Defense Content Discovery & Retrieval Integrated Project Team. *Content Discovery and Retrieval Integrated Project Team Reference Architecture (CDR-RA)*.

Available online Intelink-U at: <http://purl.org/IC/Standards/CDR-RA>

Available online at: <http://purl.org/IC/Standards/public>

[3] CDR-SD

Intelligence Community/Department of Defense Content Discovery & Retrieval Integrated Project Team. *SOAP Interface Specification for Content Discovery and Retrieval: Describe (CDR-SD)*.

Available online Intelink-U at: <http://purl.org/IC/Standards/CDR-SD>

Available online at: <http://purl.org/IC/Standards/public>

[4] CDR-SF

Intelligence Community/Department of Defense Content Discovery & Retrieval Integrated Project Team. *Data Encoding Specification for Content Discovery and Retrieval: Specification Framework (CDR-SF)*.

Available online Intelink-U at: <http://purl.org/IC/Standards/CDR-SF>

Available online at: <http://purl.org/IC/Standards/public>

[5] IC ITE INC1 IMPL

Office of the Director of National Intelligence. *Intelligence Community Information Technology Enterprise (IC ITE) Increment 1 Implementation Plan*. July 2012.

Available online Intelink-TS at: <http://go.ic.gov/HvBHBmY>

[6] ICD 500

Office of the Director of National Intelligence. *Director of National Intelligence Chief Information Officer*. Intelligence Community Directive 500. 7 August 2008.

Available online Intelink-TS at: <http://go.ic.gov/enm8L9x>

Available online at: http://www.dni.gov/files/documents/ICD/ICD_500.pdf

[7] ICPG 710.1

Assistant Director of National Intelligence for . *Application of Dissemination Controls: Originator Control*. Intelligence Community Policy Guidance 710.1. 25 July 2012.

Available online Intelink-TS at: <http://go.ic.gov/yAqVQ0H>

[8] ICS 500-20

Director of National Intelligence Chief Information Officer. *Intelligence Community Enterprise Standards Compliance*. Intelligence Community Standard 500-20. 16 December 2010.

Available online Intelink-TS at: <http://go.ic.gov/QUdIJkZ>

Available online Intelink-U at: https://intelshare.intelink.gov/sites/odni/cio/ea/library/Data%20Specifications/500-21/500_20_signed_16DEC2010.pdf

[9] IETF-RFC 2119

Internet Engineering Task Force. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997.

Available online at: <http://tools.ietf.org/html/rfc2119>

[10] IETF-RFC 4287

M. Nottingham, R. Sayre. *The Atom Syndication Format*. December 2005.

Available online at: <http://www.ietf.org/rfc/rfc4287.txt>

[11] Joint IC/DoD Memorandum

Intelligence Community Chief Information Officer, and Department of Defence Chief Information Officer. *IC and DoD Commitment to an Interoperable Service-Based Environment*. 13 July 2007.

Available online at: <http://www.docstoc.com/docs/797594/Department-of-Defense-DoD-and-Intelligence-Community-IC-Commitment-to-an-Interoperable-Services-Based-Environment---Enterprise-Services>

[12] REST

R Fielding, R Taylor. *Principled Design of the Modern Web Architecture*. ACM Transactions on Internet Technology. Vol 2, No. 2, May 2002, pages 115-150.

Available online at: <http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf>
[<http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf>]

Appendix G Points of Contact

The Intelligence Community Chief Information Officer (IC CIO) facilitates one or more collaboration and coordination forums charged with the adoption, modification, development, and governance of IC technical specifications of common concern. This technical specification was produced by the IC CIO and coordinated with these forums, approved by the IC CIO or a designated representative, and made available at DNI -sponsored web sites. Direct all inquiries about this IC technical specification to the IC CIO, an IC technical specification collaboration and coordination forum, or IC element representatives involved in those forums.

Public Website: <http://purl.org/ic/standards/public>

E-mail: ic-standards-support@intelink.gov [mailto:ic-standards-support@intelink.gov].

Appendix H IC CIO Approval Memo

An Office of the Intelligence Community Chief Information Officer (OCIO) Approval Memo should accompany this enterprise technical data specification bearing the signature of the Intelligence Community Chief Information Officer (IC CIO) or an IC CIO -designated official(s). If an OCIO Approval Memo is not accompanying this specification's version release package, then refer back to the authoritative web location(s) for this specification to see if a more complete package or a specification update is available.

Specification artifacts display a date representing the last time a version's artifacts as a whole were modified. This date most often represents the conclusion of the IC Element collaboration and coordination process. Once the IC Element coordination process is complete, the specification goes through an internal OCIO staffing and coordination process leading to signature of the OCIO Approval Memo. The signature date of the OCIO Approval Memo will be later than the last modified date shown on the specification artifacts by an indeterminable time period.

Upon signature of the OCIO Approval Memo, IC Elements may begin to use this specification version in order to address mission and business objectives. However, it is critical for IC Elements, prior to disseminating information encoded with this new specification version, to ensure that key enterprise services and consumers are prepared to accept this information. IC Elements should work with enterprise service providers and consumers to orchestrate an orderly implementation transition to this specification version in concert with mandatory and retirement usage decisions captured in the IC Enterprise Standards Baseline as defined in Intelligence Community Standard (ICS) 500-20.^[8]