



S1000D

THE
BRIDGE PROJECT

SCORM

S1000D – SCORM Bridge Application Programming Interface (API)

Bridging the gap between S1000D and SCORM

Ensuring learning data and technical publication data are developed and maintained based on consistent Integrated Logistic Support data

9/30/2011

Version 1.0

Table of Contents

1	Introduction	3
1.1	Overview	3
1.2	Purpose	3
1.3	Scope	3
1.4	Conceptual Environment and Assumptions.....	5
1.5	Intended Audience.....	6
1.6	Organization of this Document.....	6
2	S1000D – SCORM Bridge SOAP API Overview.....	7
2.1	Services Architecture	9
3	Web Service Operation Definitions	9
3.1	Connect	9
3.2	Disconnect.....	11
3.3	Search	11
3.4	GetCSDBObject	14
3.5	AddCSDBObject	14
3.6	ApproveCSDBObject	15
3.7	CheckOut.....	17
3.8	UndoCheckOut	18
3.9	CheckIn.....	19
3.10	GetListOfCheckedOutCSDBObjects	20
4	Data Types	21
4.1	S1StructuredIdentifier_Type	21
4.2	SearchResult_Type.....	22
4.3	CheckedOutData_Type.....	24
4.4	Faults	24

1 Introduction

1.1 Overview

This document defines the S1000D – SCORM Bridge Web Service Application Programming Interface (API). The Bridge API uses a set of non-proprietary Web Service specifications to define and transport data through the web service operations. This document also provides additional clarifications and recommendations to enhance interoperability through the use of the Bridge API. The specifications and resources used in the creation of the Bridge API include, but are not limited to:

- Simple Object Access Protocol (SOAP) Version 1.1
- Web Service Definition Language (WSDL) Version 1.1
- Extensible Markup Language (XML) Version 1.0 (Second Edition)
- Web Services Interoperability (WS-I) Basic Profile Version 1.0

1.2 Purpose

The purpose of the S1000D-SCORM Bridge Web Service API is to define and describe an interoperable web service and set of web service operations through the use of WSDL. This specification defines a set of minimum requirements using SOAP to transport data and requests between systems (web service and a client). The main driver of this specification is to enable client applications and services to be developed and to communicate with each other using the SOAP transactions described by this specification. The specification is written in a manner to support multiple types of server-side services and client applications. However, one of the main business drivers and use cases is between a Common Source Database (CSDB) and a Learning Content Authoring Tool (LCAT). Section 1.3 provides more background and scoping of the business drivers and needs being addressed by this specification.

1.3 Scope

This document provides a technical specification for a web service to electronically exchange data between two entities. The document focuses on defining an S1000D-SCORM Bridge API WSDL and a transport mechanism (i.e., SOAP). The scope of use for the web service is focused on transferring structured S1000D data and SCORM content between two entities.

The scope of this specification was designed and developed to support S1000D version 4.0. The specification may be usable as-is with past versions of the S1000D specification; however, there is no guarantee. Some operations defined by this specification may behave differently for past versions of the S1000D specification (e.g., based on attribute/element name that have changed over these versions or introduction of new attributes/elements).

The new S1000D 4.0 specification introduces functionalities that tighten the relationship between S1000D and SCORM. They provide opportunities to improve how technical data and associated learning content are life cycle-managed and produced in a CSDB environment. S1000D can help control information re-use between the technical documentation and training development disciplines as well as improve how training material is directly configured to the systems it supports.

Because of the new S1000D 4.0 functionality enabling learning content support, traditional SCORM-based learning content development tools may take advantage of having access to S1000D CSDB Management Systems that embed the ability to create SCORM content packages. These opportunities may be achieved using a common communication protocol between learning content development tools and S1000D databases, and between S1000D databases and applications that create SCORM conformant content packages. Improved harmonization between S1000D and SCORM is the basis for the S1000D-SCORM Bridge Application Programming Interface (API), known as “the Bridge”.

Product Life Cycle Support (PLCS) is an ISO standard that enables the creation and management through time of a controlled and authoritative set of product and support data. Test cases have demonstrated interoperability opportunities between S1000D and PLCS. With the emergence of the Bridge, the opportunity arises to improve integration of learning and human skills in a fast changing product support environment. Interoperability and cost of ownership reduction may now be enhanced by a controlled and automated provision of validated data for product support content development, usage and feedback.

As groups producing technical publications and training products operate within an integrated logistics environment, a CSDB must receive input from disparate production systems through a common data exchange. To achieve this vision, exchange packages must be defined, and the work must be founded on internationally agreed upon Integrated Logistic Support (ILS) standards, such as those provided by the PLCS standard (ISO 10303-239).

The Bridge specification focuses on data exchange between learning content authoring environments and CSDB Management Systems during the production of learning information to be used in SCORM-conformant training products. Figure 1 illustrates the domain and the areas of exchange that are being specified.

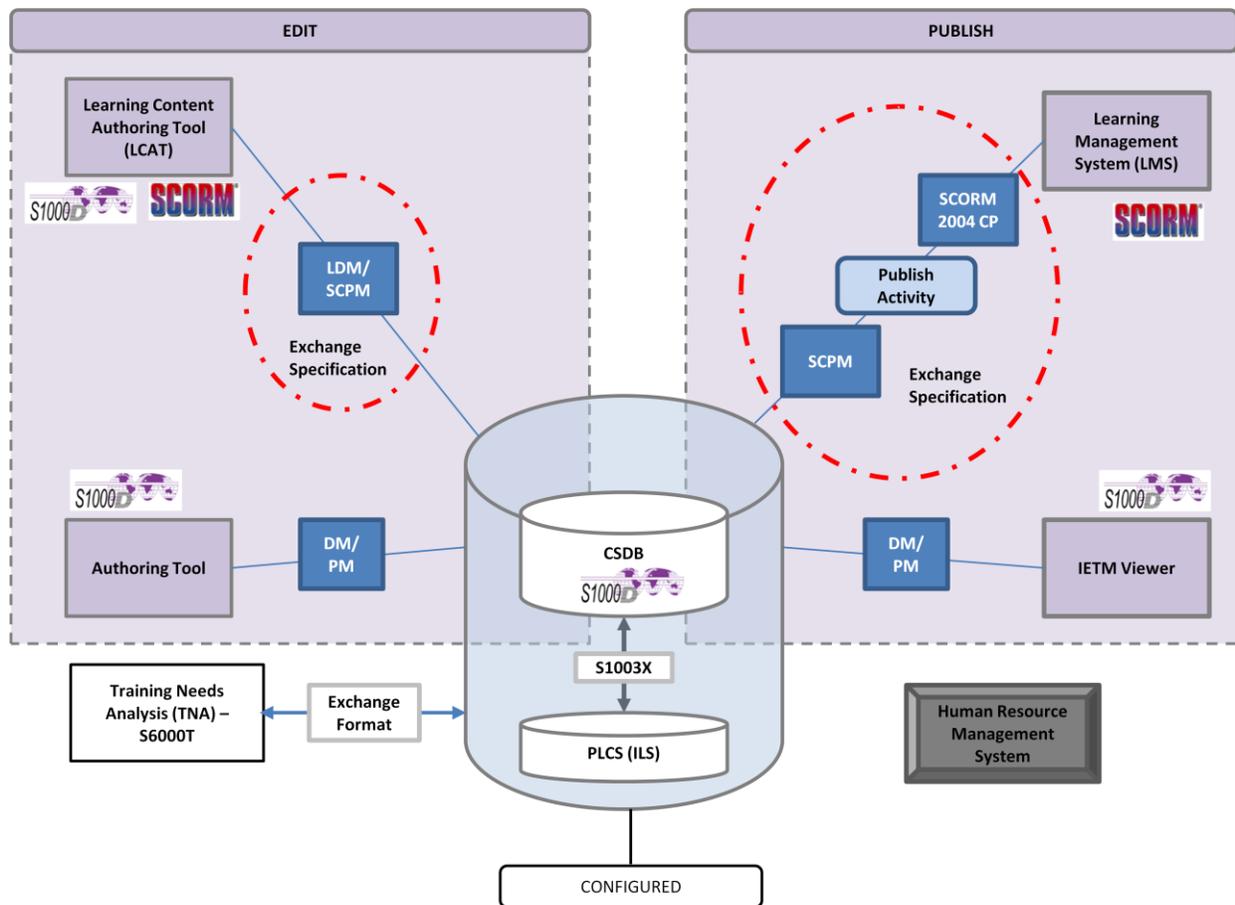


Figure 1: API Specification Focus Areas

During the editing phase, Learning Content Authoring Environments have a need to interface and exchange data with a CSDB. A common interoperable exchange mechanism (i.e., communication protocol and standard data) currently does not exist. The Bridge defines the communication protocol and the standardized information data sets that are exchanged across the communication protocol.

During the publish phase, content that resides within a CSDB is not interoperable with common on-line training environment in use today. Learning Management Systems (LMS) continue to evolve and support SCORM conforming content packages as the preferred data format for training delivery. The Bridge specifies functions that could be used in the transformational processes that must occur in order to meet the requirements of deploying and publishing CSDB data into a SCORM LMS environment.

1.4 Conceptual Environment and Assumptions

The Bridge API was defined to support a conceptual environment where authoring tools were separate from the environments managing the S1000D data. However, nothing in particular about the specification prohibits the Bridge API from being used in cases where the authoring tool is tied to the CSDB Management System (or provided as part of the CSDB Management System solution). With the Bridge API there are some general assumptions that have been defined:

1. The implementation and deployment of the Bridge API provides a web service endpoint that is uniquely identifiable (e.g., URL).
2. This web service endpoint is provided or is known to the LCAT prior to any use of the web services defined. This endpoint is needed by the Bridge API in order to provide the connection from the LCAT to the CSDB Management System. How this endpoint is provided to the LCAT is outside the scope of this specification.
3. Appropriate accounts are configured with the CSDB Management System prior to using the web service. These accounts provide the distinguishing characteristics of users, roles and rights to perform appropriate actions. In some cases these user accounts, roles and rights may have to be coordinated with the LCAT. How these user accounts, roles and rights are determined are outside the scope of this specification.
4. Any project creation and/or setup procedures needed with CSDB Management Systems or LCATs have been configured ahead of time. These types of procedures are outside the scope of this specification.

1.5 Intended Audience

This document is intended for web service developers with a good working knowledge of web services architectures, SOAP and WSDL. The intent is to lay the foundation for these developers to be able to build services that match the requirements outlined and client applications to leverage those services.

1.6 Organization of this Document

This S1000D – SCORM Bridge API document consists of the following sections:

- **Section 1: Introduction:** provides a high-level overview, scope and purpose for the document
- **Section 2: S1000D – SCORM Bridge SOAP API Overview:** provides an introduction to the S1000D – SCORM Bridge API.
- **Section 3: S1000D – SCORM Bridge API WSDL:** provides an overview of the purpose of the WSDL and outlines the requirements for constructing valid SOAP request and response messages. This section also defines the various methods and method syntax defined by the S1000D-SCORM Bridge API.
- **Section 4: S1000D – SCORM Bridge API SOAP Data Types:** describes all of the different data types used in conjunction with the WSDL.

2 S1000D – SCORM Bridge SOAP API Overview

The goal of the Bridge is to develop a common, interoperable communication protocol and data exchange mechanism that could be implemented by a variety of applications. The API is web service-based and defines a set of operations, data requirements, message format constraints and behaviors associated with each operation. The API has a defined WSDL that enables multiple CSDB Vendors to build a set of common, standardized web service operations that can be utilized by a variety of different Learning Content Authoring Environments. Figure 2 illustrates a conceptual view of the Bridge.

The WSDL enables multiple implementations (CSDB Management System 1, CSDB Management System 2 and CSDB Management System 3) to define a set of standard operations that can be invoked using commonly accepted SOAP protocols by any application (LCAT 1 – 4) that adheres to the message descriptions defined in the WSDL. This type of implementation enables a single learning content authoring tool (e.g., LCAT 1 in Figure 2) to interface with multiple CSDB implementations without the need to define or utilize a different set of communication protocols and data exchange requirements.

The Bridge implementation enables CSDB Management Systems to abstract the interface of the operations from the actual implementation details. In this case the interface becomes well known and established and the implementation details can be proprietarily developed.

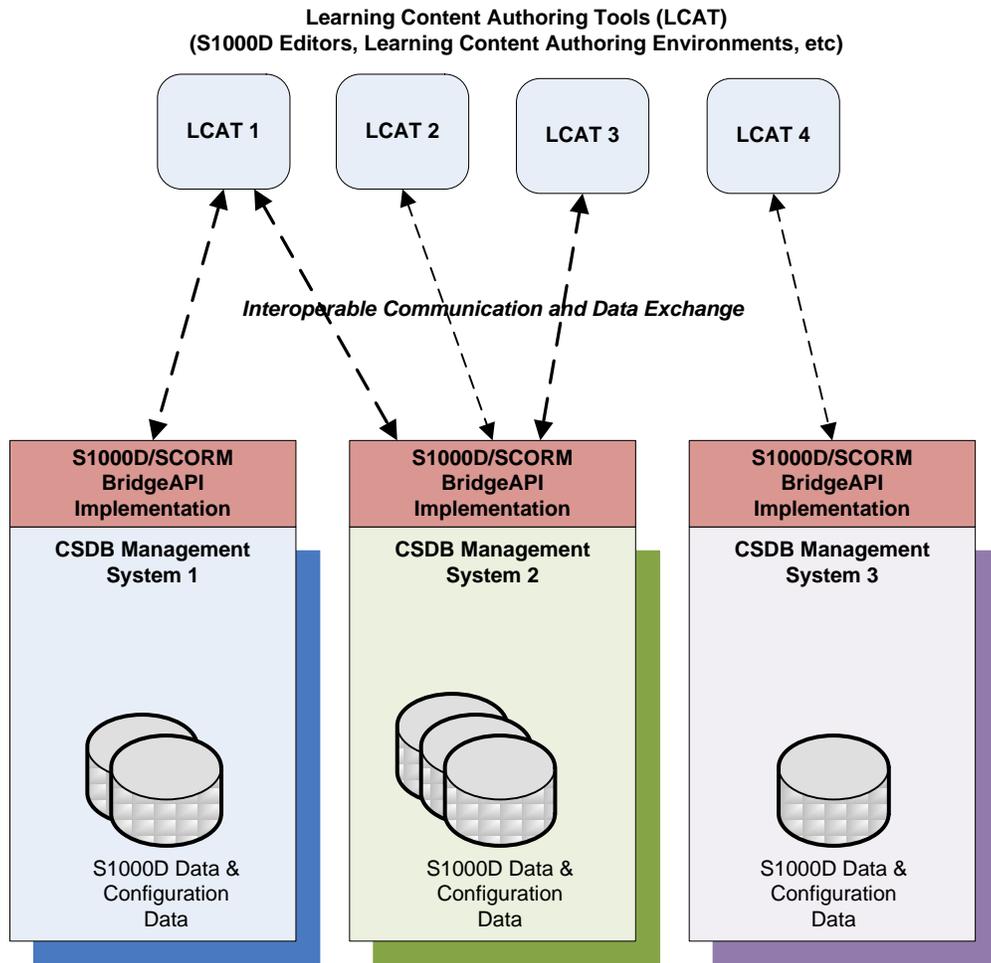


Figure 2: The Bridge Conceptual Model

2.1 Services Architecture

Like traditional web services, the S1000D-SCORM Bridge SOAP services architecture is a combination of client-side and service-side software, hardware, schemas and other implementation specific services. This service architecture can be depicted as in the following figure.

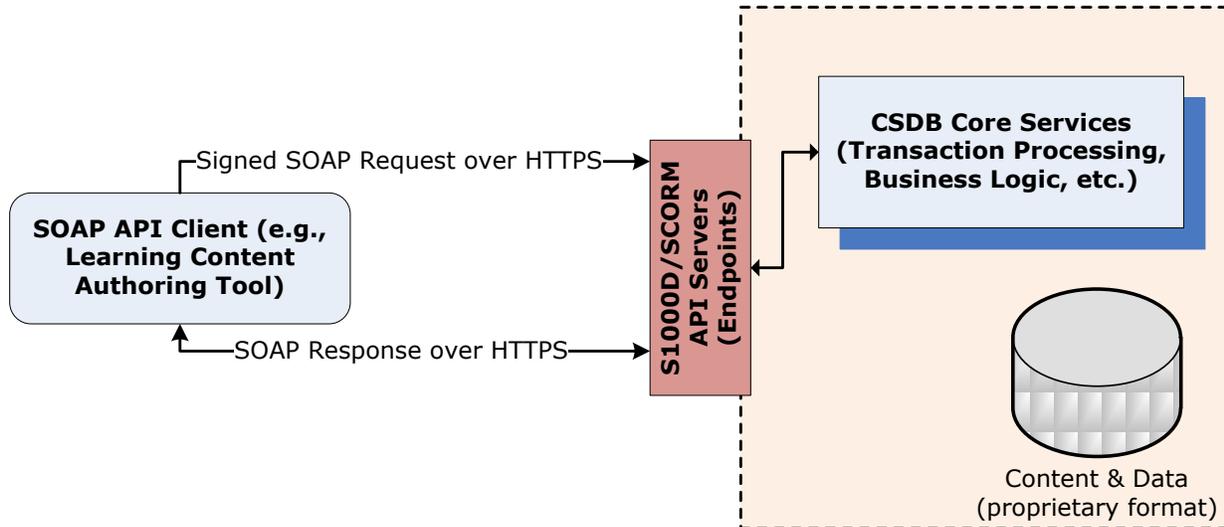


Figure 2.1.a: S1000D-SCORM Bridge Conceptual Services Architecture

3 Web Service Operation Definitions

3.1 Connect

The `Connect` operation associates a learning content authoring tool or application with a CSDB Management System. The `Connect` operation enables the CSDB Management System to establish any pertinent initiation procedures or setup in order to accept other web service calls from a learning content authoring tool or application.

The `Connect` operation must create a session identifier that is returned to the application that invoked the operation. The session identifier will be needed for other operations in order for the CSDB Management System to verify and authenticate the operation. The syntax and format of the session identifier should be defined by the CSDB Management System and must be unique enough for that CSDB Management System to recognize multiple web-service requests from multiple authoring environments or applications.

The ability to use a CSDB Management System provided session identifier to reconnect to a CSDB Management System is not supported by this specification. If a LCAT would like to establish a new connection with the CSDB Management System or an older session has completed, it should do so by issuing a new `Connect` request. This request will return a new session identifier.

From time to time, CSDB Management Systems may wish to clean up and maintain session information potentially being stored to help manage the different sessions. This specification does not provide any requirements for management, maintenance or clean up of sessions. How these types of activities are

handled by the CSDB Management System are outside the scope of this specification. If supported, it is recommended that information should be provided in some form to users of the CSDB Management Systems in order to make them aware of this maintenance policy.

Input Parameters:

- `UserName (xs:string)`: The user name associated with the end user invoking the operation through a learning content authoring tool or application. It is assumed that the end user has already established a user name with the CSDB Management System.
- `Password (xs:string)`: The password associated with the end user invoking the operation through a learning content authoring tool or application. It is assumed that the end user has already established a password with the CSDB Management System.
- `Identifier (xs:string)`: The identifier of the CSDB Management System which the end user wishes to connect to. How the `Identifier` has been acquired for this operation is outside the scope of this specification. It is assumed that the `Identifier` is provided to the LCAT at some point.

Output Parameters:

- `SessionID (xs:string)`: The session identifier for the connecting end user. The session identifier is important because it enables the CSDB Management System to recognize information about the incoming web-service call (e.g., source of the call and rights that can be performed). The session identifier is needed for other web-service operations and it is the responsibility of the end user to protect and manage the session identifier. The session identifier must be unique to the end user of the CSDB Management System. The CSDB Management System may be maintaining multiple session identifiers per given learning content authoring tool or application. The session identifier is a means for the CSDB Management System to track the different users of the system and what rights/roles they have access to.
 - NOTE: Since some of the operations defined in the specification can be invoked across sessions, the CSDB Management System may need to track session information, user identification/credentials and user rights/roles across sessions. For example, a user may check a file out (`CheckOut`) in one session, but may check it in (`CheckIn`) in another. In these cases the CSDB Management System will have to track the user across these sessions in order to verify that the user can check in the file in a new session. If the CSDB Management System is only tracking this information during a single session, then an erroneous fault may be detected. By tracking operations across sessions, the CSDB Management System will be able to support these scenarios.

Error Conditions:

- `INVALID_USER_ID`
- `INVALID_PASSWORD`
- `CSDB_MGMT_SYSTEM_NOT_RECOGNIZED`

3.2 Disconnect

The `Disconnect` operation terminates an association of a learning content authoring tool or application with a CSDB Management System. The `Disconnect` operation enables the CSDB Management Systems to service any remaining actions associated with closing out a connection with a given application (e.g., perform any environment clean up responsibilities).

During the disconnecting process, the session identifier for the end user is disabled. The CSDB Management System will not process any more requests with that session identifier.

Input Parameters:

- `SessionID (xs:string)`: The session identifier for the connected end user.

Output Parameters:

- None

Error Conditions:

- `INVALID_SESSION_IDENTIFIER`

3.3 Search

The `Search` operation allows an authoring tool or application to search for a variety of S1000D data within a CSDB Management System. The `Criteria` input parameter is used by CSDB Management Systems to determine the criteria in which to perform the search. The `Criteria` input parameter takes the form of a simplified XPath expression. This expression can reference any valid S1000D XML element or attribute. This XPath expression will then be evaluated by the CSDB Management System and a set of search results matching the request will be returned. If the XPath expression is not valid, the CSDB Management System will indicate an error through a SOAP Fault (see Error Conditions).

Due to the nature of a CSDB Management Systems ability to manage an exorbitant number of CSDB Objects, there will be cases where `Search()` operations will cause numerous hits and/or cause processing delays or errors within the CSDB Management System. In order to assist with different processing challenges, the `Search()` operation supports the ability to request a certain number of results to be returned. The client applications can use this to narrow down the number of results returned (refer to the `RequestedNumberOfResults` attribute for more requirements and behavior details).

Using this XPath syntax, there are several types of searches that could be performed. These types of searches often combine Boolean operators.

Table: Examples of XPath Searches

Type of Search	Description	Example
General	This is a search that uses a general string	<code>//*[/@*='<general_search_string>']</code>

Text Search	and is used to search the entire CSDB Object field set for an attribute that contains the <general_search_string> value.	<pre>//*/@*='007'</pre> <pre>//*[contains(@*,'007')] – uses the contains() function</pre> <p>Show me anything that contains an attribute value of 007.</p>
General Text Search	This is a search that uses a general string and is used to search the entire CSDB Object field set for an element that contains the <general_search_string> value.	<pre>//*[.='<general_search_string>']</pre> <pre>//*[.='wheel']</pre> <pre>//*[contains(.,'wheel')] – uses the contains() function</pre> <p>Show me anything that contains an element with a value of wheel.</p>
Single Term	This is a search using a single term. This type of search is used to return anything that has the term provided in the specified field.	<pre>//dmAddress/dmIdent/dmCode[@infoCode='720']</pre> <p>Show me anything that has an information code of 720.</p>
General Listing	This type of search can be used to list out a set of specific types of CSDB Objects. The search can also be supplemented with use of Boolean operators to narrow search results down to a manageable list.	<pre>//dmIcode</pre> <p>Show me a listing of Data Module Requirement List objects stored in the CSDB Management System.</p> <pre>//scormContentPackageCode</pre> <p>Show me a listing of SCORM Content Package Module objects stored in the CSDB Management System.</p>
Boolean AND	This is a search using the Boolean operator and with two single terms. This type of search is used to return anything that has both single terms in any field (XML element or attribute).	<pre>//dmAddress/dmIdent/dmCode[@infoCode='720'] and</pre> <pre>//dmAddress/dmIdent/dmCode[@learnCode='T12']</pre> <p>Show me anything that has the infoCode of 720 and a learnCode of T12.</p> <pre>//dmAddress/dmIdent/dmCode[@modelIdentCode='S1000DBIKE'] and</pre> <pre>//dmAddress/dmIdent/dmCode[@systemCode='DA1'] and</pre> <pre>//*[not (//dmCode[@learnCode])]</pre> <p>Show me all Technical Data Modules (no learning Data Modules) that have a Model Ident Code of S1000DBIKE and a System Code of DA1.</p>
Boolean OR	This is a search using the Boolean	<pre>//dmAddress/dmIdent/dmCode[@infoCode='720'] or</pre> <pre>//dmAddress/dmIdent/dmCode[@learnCode='T12']</pre>

	operator <code>or</code> with two single terms. This type of search is used to return anything that has either one of the single terms in any fields specified (XML element or attribute).	Show me anything that has the infoCode of 720 or a learnCode of T12.
Not Equivalent	This is a general text search looking for CSDB Objects that do not contain a certain value.	<code>//dmAddress/dmIdent/dmCode[@learnCode!='T12']</code> Show me anything that does not have a learnCode of T12.
Grouping	Grouping allows for complex criteria to be built using supported Boolean operators (or, and).	<code>(//dmAddress/dmIdent/dmCode[@infoCode='720'] or //dmAddress/dmIdent/dmCode[@learnCode='T12']) and //dmAddress/dmIdent/language[@countryIsoCode='US']</code> Show me anything thing that has an infoCode of 720 or a learnCode of T12 and a country code of US.

Input Parameters:

- `Criteria (xs:string)`: The specific search criteria.
- `RequestedNumberOfResults (xs:integer)`: This is an optional parameter that provides a requested number of results that the client application would like to see. This integer number should be used by the CSDB Management System as an indicator for the number of search results returned.
- `SessionID (xs:string)`: The session identifier for the connected end user.

Output Parameters:

- `SearchResults (SearchResult_Type)`: The Search Results matching the input search criteria (Criteria).

Error Conditions:

- `INVALID_SESSION_IDENTIFIER`
- `SESSION_NOT_ACTIVE`
- `INVALID_SEARCH_CRITERIA`
- `OPERATION_NOT_PERMITTED`
- `PROCESSING_ERROR_DURING_SEARCH_REQUEST`

3.4 GetCSDBObject

The `GetCSDBObject` operation returns a read only version of the CSDB Object identified. If the identified object (`S1StructuredIdentifier`) is attempted to be checked in (by the `CheckIn` method), then the CSDB Management System should not permit the CSDB Object to be checked in and the system should indicate that an error condition has been encountered (See `CheckIn` Error Conditions). The CSDB Object identified is returned as an attachment through the SOAP Response. The user of the object should not permit an author to make any changes to the CSDB Object. By retrieving the CSDB Object, a private viewing copy of the object is created for the client application that made the service call.

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object wishing to be retrieved (a read-only copy).
- `SessionID` (`xs:string`): The session identifier for the connected end user.

Output Parameters:

- `CSDBObject` (`Attachment_Type`): This parameter represents the read-only version of the CSDB Object that is being retrieved. The object being retrieved can be an object of different formats (e.g., XML, PDF, PNG). The `Attachment_Type` contains an `ObjectMimeType` indicator to assist in the understanding of the object being returned.

Error Conditions:

- `INVALID_STRUCTURED_IDENTIFIER`
- `UNRECOGNIZED_S1_STRUCTURED_IDENTIFIER`
- `INVALID_SESSION_ID`
- `SESSION_NOT_ACTIVE`
- `OPERATION_NOT_PERMITTED`

3.5 AddCSDBObject

The `AddCSDBObject` operation indicates to the CSDB Management System that the given CSDB Object should be persisted within the CSDB. Once the object is persisted within the CSDB Management System, the CSDB Object will become available for other operations (e.g., check out, edit and check in). This method is defined to support adding one CSDB Object into the CSDB Management System at a time.

Although not required by this specification, the CSDB Management System may provide support for validating the CSDB Object being added in accordance with requirements of the S1000D Specification (e.g., schema validation). The CSDB Management System can use the `schemaLocation` attribute to determine the schemas to be used for validation purposes. The `schemaLocation` attribute is required to be present in the S1000D CSDB Objects.

This operation can also be used to add media files to the CSDB Management System; however, in these cases there are no S1000D XML constructs to validate. In these cases, the CSDB Management System could validate the Information Control Number (ICN) according to the S1000D Specification (refer to

Section 4.4). If a CSDB Object has references to ICNs, the CSDB Management System should not attempt to validate the reference objects, it should validate only the current CSDB Object XML Instance it is processing. Some CSDB Management Systems may not accept a new CSDB Object instance unless the media files referenced (e.g., figures) are already existing in the CSDB Management System. This specification recommends that all reference media (i.e., through an ICN reference) exist within a CSDB Management System prior to adding a new CSDB Object.

Prior to issuing the `AddCSDBObject` operation, client applications could provide validation services to ensure that the data being persisted in the CSDB Management System follows the rules and requirements of S1000D. In these cases the client applications could use the S1000D Schemas and any identified Business Rules Exchange (BREX) data modules associated with the object.

The CSDB Management System is required to set the `issueNumber` to “000” and the `inWork` number to “01” for CSDB Objects that have S1000D XML with these values (e.g., Data Modules). In the case where the object being added is an ICN, the CSDB Management System has no data to update.

The `AddCSDBObject` operation passes the CSDB Object to the Web-service by using SOAP Attachments.

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object to add to the CSDB Management System.
- `CSDBObject` (`Attachment_Type`): The CSDB Object to be added to the CSDB Management System.
- `SessionID` (`xs:string`): The Session Identifier used for validating that authentication has been performed. The session id may also be used for verifying that the authenticated user has permission to perform the method.

Output Parameters:

- None (if there are errors during the processing of the operation, error conditions will arise)

Error Conditions:

- `INVALID_STRUCTURED_IDENTIFIER`
- `INVALID_SESSION_IDENTIFIER`
- `CSDB_OBJECT_STRUCTURED_ID_MISMATCH`
- `CSDB_OBJECT_INVALID_ACCORDING_TO_SCHEMA`
- `CSDB_OBJECT_INVALID_ACCORDING_TO_DEFAULT_BREX`
- `CSDB_OBJECT_INVALID_ACCORDING_TO_PROJECT_BREX`
- `SESSION_NOT_ACTIVE`
- `OPERATION_NOT_PERMITTED`
- `CSDB_OBJECT_ALREADY_EXISTS`

3.6 ApproveCSDBObject

The `ApproveCSDBObject` operation indicates to the CSDB Management System that the current CSDB Object referenced is approved for release. At this point all project/organization QA conditions and

processes have been met (e.g., project defined use of quality assurance element). During this process, the `issueNumber` needs to be incremented by 1.

The S1000D specification defines the behavior for information management in the form of version control of data modules (Refer to Chapter 4.7 Information Management – Version control of data modules).

An issue number is set to “000” when a data module is initially created. The issue number will remain at “000” until the data module is approved for release. Once approved for release, the issue number is incremented by 1 and is set to “001”. For every subsequent approval for release, the issue number is incremented by 1. The issue number works in conjunction with the `inWork` number in the sense that the `inWork` number is incremented every time a data module is changed within an issue. Once a new issue of the data module is made the `inWork` number is set back to “00”. The CSDB Management System is responsible for understanding these rules and is required to change the `issueNumber` and `inWork` number when appropriate.

Step	Attribute Value	Comment
New data module, first in work version	<code>inWork = "01"</code> <code>issueNumber="000"</code>	First instance of a CSDB Object within a CSDB Management System
New data module, second in work version	<code>inWork="02"</code> <code>issueNumber="000"</code>	Change has been made to the CSDB Object
New data module, in work version "NN"	<code>inWork="NN"</code> <code>issueNumber="000"</code>	"NN"th change to the CSDB Object
First issue of data module	<code>inWork="00"</code> <code>issueNumber="001"</code>	Once a CSDB Object is approved for release, the <code>inWork</code> number is set to “00” and the <code>issueNumber</code> is set to “001” indicating the first issue or release of the CSDB Object
First issue of data module, first in work	<code>inWork="01"</code> <code>issueNumber="001"</code>	When the newly issued object is checked out and changed, the CSDB Management System is required to increment the <code>inWork</code> number by 1 upon check in

When an `ApproveCSDBObject` method is invoked, the CSDB Management System shall adhere to certain requirements. There may be rules, policies or workflow where CSDB Objects need to be approved through some sort of project/organization quality assurance (QA) processes. These may include certain QA conditions being met, verification process (first verified, second verified elements being set, etc.) The requirements are as follows:

1. If the CSDB Object has not been approved (i.e., project/organization defined QA procedures), then the method should make no change and the error (`CSDB_OBJECT_NOT_APPROVED`) shall be reported.
2. If the CSDB Object has been approved, then the CSDB Management System shall
 - a. increment the `issueNumber` by 1
 - b. set the `inWork` number to 00

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object to approve. The `S1StructuredIdentifier` shall be represented as valid S1000D Uniform Resource Name (URN) syntax (e.g., URN:S1000D:DMC-S1000D-A-07-05-0000-00A-000A-A_I-001_W-00_L-SX_C-US). For more information, refer to S1000D Specification: Chapter 7.2.1.4 IETP – Resource Resolution.
- `SessionID` (`xs:string`): The session identifier for the connected end user.

Output Parameters:

- `IssueNumber` (`xs:string`): This parameter represents the newly incremented issue number. This value is returned to the application invoking this method. What the invoking application does with this number is outside the scope of this specification.

Error Conditions:

- `INVALID_STRUCTURED_IDENTIFIER`
- `INVALID_SESSION_IDENTIFIER`
- `SESSION_NOT_ACTIVE`
- `OPERATION_NOT_PERMITTED`

3.7 CheckOut

The `CheckOut` operation signifies to a CSDB Management System that a learning content authoring tool or application would like to reserve a CSDB object. The `CheckOut` operation requires the CSDB Management System to perform appropriate CSDB Management System specific checkout procedures. Minimally, the CSDB Object is required to be locked (or reserved) and not permitted to be edited by any other user or entity until the CSDB Object’s lock is released. By checking out the CSDB Object, a private working copy of the object is created for the client application that made the service call.

The `S1StructuredIdentifier` can be used in several ways during a `CheckOut` operation:

1. If the client would like the latest version of the CSDB Object, the client application should not include any of the following issue information: `inWork`, `issueNumber`. When these values are left out of the `S1StructureIdentifier`, the CSDB Management System is required to return the latest version of that `CSDBObject`.
2. If the client provides the `inWork` and `issueNumber`, then the CSDB Management System has two options:
 - a. Check out and return the specific identified CSDB Object. This implies that the CSDB Management System supports some form of branching/merging feature to align the changed file (e.g., specifically if the identified CSDB Object is not the current version). NOTE: A branching/merging feature is not required to be supported by this specification.
 - b. Determine if the `inWork` and `issueNumber` provided represents the latest version. If so, then the CSDB Object can be returned. If the `inWork` and `issueNumber` does not identify the latest version, then the CSDB Management System can issue an appropriate fault (`INVALID_STRUCTURED_IDENTIFIER`) to indicate the error along with a description of the error.

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object to check out. The `S1StructuredIdentifier` shall be represented as valid S1000D URN syntax (e.g., `URN:S1000D:DMC-S1000D-A-07-05-0000-00A-000A-A_I-001_W-00_L-SX_C-US`).
- `SessionID` (`xs:string`): The session identifier for the connected end user.

Output Parameters:

- `CSDBObject` (`Attachment_Type`): This parameter represents the CSDB Object that is being checked out.

Error Conditions:

- `INVALID_STRUCTURED_IDENTIFIER`
- `INVALID_SESSION_IDENTIFIER`
- `SESSION_NOT_ACTIVE`
- `OPERATION_NOT_PERMITTED`
- `CSDB_OBJECT_ALREADY_CHECKED_OUT`
- `CHECKED_OUT_OBJECT_LIMIT_REACHED`

3.8 UndoCheckOut

The `UndoCheckOut` operation enables an authoring tool or application to reverse the effect of a `CheckOut` operation call and remove the private working copy of the checked-out CSDB Object. During the processing of an `UndoCheckOut`, the CSDB Management System should ignore any changes that may have been made to the CSDB Object. This also requires the CSDB Management System to not adjust the `inWork` number. Since the `UndoCheckOut` operation was called, the authoring tool or application has stated its intent to ignore all work that was done; therefore, there is no need to create a new in work version.

This operation can only be performed by a user who has the rights to perform the operation:

- The original user who checked out the file, or
- A systems administrator

NOTE: Because an `UndoCheckOut` can be performed across sessions, the CSDB Management System must design a way to track who has checked out what CSDB Objects. How this is implemented is outside the scope of this specification.

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object to perform the `UndoCheckOut` operation on. The `S1StructuredIdentifier` shall be represented as valid S1000D URN syntax (e.g., `URN:S1000D:DMC-S1000D-A-07-05-0000-00A-000A-A_I-001_W-00_L-SX_C-US`).
- `SessionID` (`xs:string`): The session identifier for the connected end user.

Output Parameters:

- None

Error Conditions:

- INVALID_STRUCTURED_IDENTIFIER
- INVALID_SESSION_IDENTIFIER
- SESSION_NOT_ACTIVE
- OPERATION_NOT_PERMITTED
- CSDB_OBJECT_NOT_CHECKED_OUT

3.9 CheckIn

The `CheckIn` operation allows an authoring tool or application to make the private working copy of the CSDB Object the current version of the CSDB Object in the CSDB Management System.

The `CheckIn` operation signifies to a CSDB Management System that a learning content authoring tool or application would like to persist changes made to the CSDB Object identified and unlock the object for future operations. The `CheckIn` operation requires the CSDB Management System to perform appropriate CSDB Management System defined check in procedures. Minimally, it is required that the CSDB Management System performs the following:

- increases the `inWork` number by 1. The `inWork` number is used to track the different drafts of the CSDB Object. The `inWork` number is part of the data module identification information found within the Identification and status section of a data module. If the `inWork` number was changed during the authoring process, the value will be ignored and set according to this rule by the CSDB Management System. This will avoid any discrepancies with understanding the requirements of the `inWork` number by authors.
- persists the CSDB Object identified through the input parameter
- unlocks the CSDB Object identified through the input parameter

Although not required by this specification, the CSDB Management System could validate the CSDB Object in accordance with requirements of the S1000D Specification (e.g., schema validation). The CSDB Management System can use the `schemaLocation` attribute to determine the schemas to be used for validation purposes. The `schemaLocation` attribute is required to be present in the S1000D CSDB Objects. Prior to issuing the `CheckIn` operation, applications could possibly reduce validation errors by using the S1000D Schemas and any identified Business Rules Exchange (BREX) data modules associated with the object.

This operation can only be performed by a user who has the rights to perform the operation:

- The original user who checked out the file, or
- A systems administrator

NOTE: Because a `CheckIn` can be performed across sessions, the CSDB Management System must design a way to track who has checked out what CDSBObjects. How this is implemented is outside the scope of this specification.

Input Parameters:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object to check in. The `S1StructuredIdentifier` shall be

represented as valid S1000D URN syntax (e.g., URN:S1000D:DMC-S1000D-A-07-05-0000-00A-000A-A_I-001_W-00_L-SX_C-US).

- `CSDBObject (Attachment_Type)`: This parameter represents the CSDB Object being checked in.
- `SessionID (xs:string)`: The session identifier for the connected end user.

Output Parameters:

- `NewInWorkNumber (xsd:string)`: This output parameter represents the final number used by the CSDB Management System for the `InWork` number. If an `InWork` number is returned, the operation was successful.

Error Conditions:

- `INVALID_STRUCTURED_IDENTIFER`
- `INVALID_SESSION_IDENTIFIER`
- `SESSION_NOT_ACTIVE`
- `OPERATION_NOT_PERMITTED`
- `CSDB_OBJECT_NOT_CHECKED_OUT`
- `CSDB_OBJECT_NOT_VALID_TO_S1000D`

3.10 GetListOfCheckedOutCSDBObjects

The `GetListOfCheckedOutCSDBObjects` operation enables a user or an authoring environment to understand which CSDB Objects the user has checked out. The CSDB Management System can determine the user by the Session Identifier provided. CSDB Management Systems should keep in mind that users may have checked out CSDB Objects in previous sessions. This implies that the CSDB Management Systems will need to maintain some sort of tracking of users across sessions.

Input Parameters:

- `SessionID (xs:string)`: The session identifier for the connected end user. The session identifier can be used to determine the user identifier for the request.

Output Parameters:

- `CheckedOutData (CheckedOutData_Type)`: A list of data about the CSDB Object(s) that are checked out by a given user identifier (associated to the Session ID).

Error Conditions:

- `INVALID_SESSION_IDENTIFIER`
- `SESSION_NOT_ACTIVE`

4 Data Types

This section defines a set of data types that are used by various API operations. These data types are common across several of the operations defined by the specification.

4.1 S1StructuredIdentifier_Type

The `S1StructuredIdentifier_Type` represents the unique name and identifier for the CSDB Objects. The `S1StructuredIdentifier_Type` is specified as a URN. Within S1000D, the use of URNs for uniquely naming and identifying CSDB Objects enables a means to provide a location independent identification structure (refer to Chapter 7.4.1.2 IETP – Resource resolution in the S1000D specification for more details).

The format of an `S1StructuredIdentifier_Type` follows the rules and requirements for URNs:

URN:NID:NSS

URN: “URN” is a required reserved prefix

NID: the namespace identifier. The reserved namespace identifier is “S1000D”.

NSS: The namespace specific string. S1000D also has a set of registered namespace identifiers that are reserved as sub namespaces for use in identifying the different information objects within S1000D. These sub namespaces should be used in the `S1StructuredIdentifier` as a prefix for the NSS followed by its corresponding S1000D syntax:

- DMC – Data Module Code
- DME – Data Module Code Extended
- PMC – Publication Module Code
- PME - Publication Module Code Extended
- SMC – SCORM Content Package Code
- SME - SCORM Content Package Code Extended
- CSN – Catalog Sequence Number
- ICN – Illustration Control Number
- COM – Comment Code
- DDN – Data Dispatch Note Code
- DML – Data Module List Code

This leads to the following required syntax for the `S1StructuredIdentifier`, based on information object being identified:

- URN:S1000D:DMC-{Code in DMC syntax}
- URN:S1000D:DME-{Code in DME syntax}
- URN:S1000D:PMC-{Code in PMC syntax}
- URN:S1000D:PME-{Code in PME syntax}
- URN:S1000D:SMC-{Code in SMC syntax}
- URN:S1000D:SME-{Code in SME syntax}
- URN:S1000D:CSN-{Code in CSN syntax}
- URN:S1000D:ICN-{Code in ICN syntax}
- URN:S1000D:COM-{Code in COM syntax}

- URN:S1000D:DDN-{Code in DDN syntax}
- URN:S1000D:DML-{Code in DML syntax}

4.2 SearchResult_Type

The `SearchResult_Type` contains information returned from a search operation in the specification. There is a different set of search results information that is returned based on the type of S1000D CSDB Object. The following list represents what types of data could be returned:

- `S1StructuredIdentifier` (`S1StructuredIdentifier_Type`): The S1000D structured identifier that represents the CSDB Object. The `S1StructuredIdentifier` is a required return value.
- `Language` (`xsd:string`): The S1000D structured identifier that represents the CSDB Object.
- `IssueNumber` (`xsd:string`): The current issue number of the requested object
- `InWork` (`xsd:string`): The current in work number of the requested object. The `InWork` is an optional return value.
- `TechName` (`xsd:string`): The tech name of the requested object. The `TechName` is an optional return value.
- `InfoName` (`xsd:string`): The info name of the requested object. The `InfoName` is an optional return value
- `SCORMContentPackageTitle` (`xsd:string`): The SCORM Content Package title.
- `PMTitle` (`xsd:string`): The Publication Module title.
- `CheckOutBy` (`xsd:string`): The current user identifier that has the requested object checked out.
- `IsReadable` (`xsd:boolean`): A Boolean indicator of whether or not this CSDB Object is read only to the user.
- `IsWritable` (`xsd:boolean`): The Boolean indicator of whether or not this CSDB Object can be updated by the user (writable).

The following table describes the requirements and more information related to the `SearchResult_Type` based on the CSDB Object identified by the search criteria.

S1000D CSDB Object	
Data Module	<ul style="list-style-type: none"> • S1StructuredIdentifier – The data module code (in URN format) • Language – made up of the ISO Country Code (<code>countryIsoCode</code> attribute of the <code><language></code> element) and ISO Language Code (<code>languageIsoCode</code> attribute of the <code><language></code> element). The syntax shall be "<code>countryIsoCode</code>" - "<code>languageIsoCode</code>". Example: US-sx • Issue Number – the current approved issue of the data module (<code>issueNumber</code> attribute of the <code><issueInfo></code> element) • In Work Number – the current in work number of the data module (<code>inWork</code> attribute of the <code><issueInfo></code> element) • Tech Name – the name of the hardware of function of the S1000D CSDB Object (<code><techName></code> element value)

Publication Module

- **Info Name** – short description of the information code for the S1000D CSDB Object (<infoName> element value). The <infoName> element is optional. If it is not provided the search results will not contain the information name
- **Check Out By** – the user identification for the user who has the CSDB Object currently checked out
- **Is Readable** – a true or false value indicating whether or not the user can read/view the data module identified
- **Is Writeable** – a true or false value indicating whether or not the user can write/modify the data module identified

SCORM Content Package Module

- **S1StructuredIdentifier** – The publication module code (in URN format)
- **Language** – made up of the ISO Country Code (countryIsoCode attribute of the <language> element) and ISO Language Code (languageIsoCode attribute of the <language> element). The syntax shall be "countryIsoCode" - "languageIsoCode". Example: US-sx
- **IssueNumber** – the current approved issue of the data module (issueNumber attribute of the <issueInfo> element)
- **InWork** – the current in work number of the data module (inWork attribute of the <issueInfo> element)
- **PMTitle** – the title for the Publication Module (<pmTitle>)
- **CheckOutBy** – the user identification for the user who has the CSDB Object currently checked out
- **IsReadable** – a true or false value indicating whether or not the user can read/view the data module identified
- **IsWriteable** – a true or false value indicating whether or not the user can write/modify the data module identified

- **S1StructuredIdentifier** – The SCORM Content Package module code (in URN format)
- **Language** – made up of the ISO Country Code (countryIsoCode attribute of the <language> element) and ISO Language Code (languageIsoCode attribute of the <language> element). The syntax shall be "countryIsoCode" - "languageIsoCode". Example: US-sx
- **IssueNumber** – the current approved issue of the data module (issueNumber attribute of the <issueInfo> element)
- **InWork** – the current in work number of the data module (inWork attribute of the <issueInfo> element)
- **SCORMContentPackageTitle** – the title for the SCORM Content Package Module (<scormContentPackageTitle>)
- **CheckOutBy** – the user identification for the user who has the

Information Control Number (ICN)

CSDB Object currently checked out

- **IsReadable** – a true or false value indicating whether or not the user can read/view the data module identified
- **IsWritable** – a true or false value indicating whether or not the user can write/modify the data module identified
- **S1StructuredIdentifier** – The Information Control Number (in URN format)
- **CheckOutBy** – the user identification for the user who has the CSDB Object currently checked out
- **IsReadable** – a true or false value indicating whether or not the user can read/view the data module identified
- **IsWritable** – a true or false value indicating whether or not the user can write/modify the data module identified

4.3 CheckedOutData_Type

A `CheckedOutData_Type` contains information that would be returned when a listing of CSDB Objects that are currently checked out is requested by a client application. The data returned contains the following information:

- `S1StructuredIdentifier (S1StructuredIdentifier_Type)`: The S1000D structured identifier that represents the CSDB Object. The `S1StructuredIdentifier` is a required return value.
- `IssueNumber (xsd:string)`: The current issue number of the requested object. The `IssueNumber` is an optional return value.
- `InWork (xsd:string)`: The current in work number of the requested object. The `InWork` is an optional return value.
- `TechName (xsd:string)`: The tech name of the requested object. The `TechName` is an optional return value.
- `InfoName (xsd:string)`: The info name of the requested object. The `InfoName` is an optional return value.
- `CheckedOutBy (xsd:string)`: The current user identifier that has the requested object checked out. The `CheckOutBy` is a required return value.
- `ObjectMIMETYPE (xsd:string)`: The Multipurpose Internet Mail Extensions (MIME) Type for the CSDB Object being added.

4.4 Faults

The S1000D-SCORM Bridge API defines a set of faults that could be returned by implementations of the API. A fault contains information that would be returned when an error condition is encountered during the processing of operations defined in this specification. A fault contains the following information:

- `ReturnCode`: An error condition token, representing the type error condition encountered. There is a Fault Type declared for each operation. Each Fault Type contains the possible set of error conditions that could be encountered.

- **ReturnCodeDescription:** A short description of the error condition. The CSDB Management System can provide as much detail as needed to describe the error condition and possibly any remedies.

The follow table defines the set of error conditions that may be encountered using the set of web-service operations defined in this specification.

Table 3.6a: Error Conditions and Descriptions

Error Condition	Error Description
INVALID_USER_ID	The user identifier is not recognized by the CSDB Management System.
INVALID_PASSWORD	The password is not valid for the given user identifier being managed by the CSDB Management System.
CSDB_MGMT_SYSTEM_NOT_RECOGNIZED	The identifier for the CSDB Management System is invalid or not recognized by the Service Provider (endpoint).
INVALID_STRUCTURED_IDENTIFIER	The S1000D structured identifier provided is invalid or not recognized by the CSDB Management System.
INVALID_SESSION_IDENTIFIER	The Session Identifier is not recognized or valid within that CSDB Management System.
NO_DATA_PROVIDERS_FOUND	The CSDB Management System could not identify any CSDBs (data providers) to return to the client application.
SESSION_NOT_ACTIVE	The Session is no longer active. The session enters the inactive state when a <i>Disconnect</i> operation has been called. No other operations can be invoked unless the session is in an active state (<i>Connect</i> operation is invoked).
INVALID_SEARCH_CRITERIA	The criterion provided is invalid and not supported by the CSDB Management System.
OPERATION_NOT_PERMITTED	The operation being performed is not permitted by the user.
INVALID_CONTENT_OBJECT_TYPE	The content object type is not recognized by the CSDB Management System.
INVALID_INFO_CONTROL_NUM	The S1000D Information Control Number (ICN) provided is invalid or not recognized by the CSDB Management System.
CSDB_OBJECT_ALREADY_CHECKED_OUT	The CSDB Object that is being attempted to be checked out is already locked and checked out by another user or entity.
CHECKED_OUT_OBJECT_LIMIT_REACHED	The number of objects checked out by the

	user has met its CSDB Management System defined limit. No more checkouts are permitted.
CSDB_OBJECT_NOT_CHECKED_OUT	The CSDB Object being attempted to be checked in is not checked out.
CSDB_OBJECT_IS_NOT_VALID_TO_S1000D	The CSDB Object is not valid according to S1000D or its schemas.
CSDB_OBJECT_ALREADY_EXISTS	The CSDB Object being added to the CSDB Management Systems already exists. The S1StructuredIdentifier used in the operation is already in the CSDB Management System.