s1kd-tools

Documentation

S1KDTOOLS-KHZAE-00000-00

Issue No. 024, 2019-12-13

List of effective data modules

The listed documents are included in issue 024, dated 2019-12-13, of this publication.

C = Changed data module

N = New data module

Document title	Data module code		Issue date	No. of pages	Applicable to
Title page	S1KDTOOLS-A-00-00-00-00A-001A-D	C	2019-12-13	1	All
List of effective data modules	S1KDTOOLS-A-00-00-00-00A-00SA-D	C	2019-12-13	3	All
Highlights	S1KDTOOLS-A-00-00-00-00A-00UA-D	C	2019-12-13	1	All
Table of contents	S1KDTOOLS-A-00-00-00-00A-009A-D	C	2019-12-13	3	All
List of abbreviations	S1KDTOOLS-A-00-00-00-00A-005A-D	C	2019-09-06	1	All
s1kd-tools - Description	S1KDTOOLS-A-00-00-00-00A-040B-D	C	2019-09-06	1	All
s1kd-tools - Introduction	S1KDTOOLS-A-00-00-00-00A-018A-D	C	2019-12-13	3	All
s1kd-tools - Building, installing and uninstalling	S1KDTOOLS-A-00-00-00-00A-920A-D	C	2019-09-06	3	All
s1kd-tools - Usage examples	S1KDTOOLS-A-00-00-00-00A-130A-D	C	2019-11-15	12	All
s1kd-tools - .defaults file identifiers	S1KDTOOLS-A-00-00-00-00A-014A-D	C	2019-12-13	6	All
s1kd-tools - Issue compatibility	S1KDTOOLS-A-00-00-00-00A-C30A-D	C	2019-12-13	3	All
s1kd-defaults - Description	S1KDTOOLS-A-30-00-00-00A-040A-D	C	2019-09-06	3	All
s1kd-dmrl - Description	S1KDTOOLS-A-22-00-00-00A-040A-D	C	2019-11-15	2	All
s1kd-newcom - Description	S1KDTOOLS-A-16-00-00-00A-040A-D	C	2019-10-04	3	All

Document title	Data module code		Issue date	No. of pages	Applicable to
s1kd-newddn - Description	S1KDTOOLS-A-17-00-00-00A-040A-D	C	2019-10-04	3	All
s1kd-newdm - Description	S1KDTOOLS-A-07-00-00-00A-040A-D	C	2019-12-13	7	All
s1kd-newdml - Description	S1KDTOOLS-A-21-00-00-00A-040A-D	C	2019-10-04	3	All
s1kd-newimf - Description	S1KDTOOLS-A-13-00-00-00A-040A-D	C	2019-10-04	3	All
s1kd-newpm - Description	S1KDTOOLS-A-12-00-00-00A-040A-D	C	2019-10-04	3	All
s1kd-newsmc - Description	S1KDTOOLS-A-35-00-00-00A-040A-D	C	2019-10-04	3	All
s1kd-newupf - Description	S1KDTOOLS-A-31-00-00-00A-040A-D	C	2019-10-04	2	All
s1kd-addicn - Description	S1KDTOOLS-A-27-00-00-00A-040A-D	C	2019-09-06	2	All
s1kd-ls - Description	S1KDTOOLS-A-06-00-00-00A-040A-D	C	2019-10-25	3	All
s1kd-ref - Description	S1KDTOOLS-A-08-00-00-00A-040A-D	C	2019-12-13	6	All
s1kd-metadata - Description	S1KDTOOLS-A-09-00-00-00A-040A-D	C	2019-12-13	4	All
s1kd-mvref - Description	S1KDTOOLS-A-19-00-00-00A-040A-D	C	2019-09-06	2	All
s1kd-sns - Description	S1KDTOOLS-A-34-00-00-00A-040A-D	C	2019-09-06	2	All
s1kd-transform - Description	S1KDTOOLS-A-15-00-00-00A-040A-D	C	2019-09-06	2	All
s1kd-upissue - Description	S1KDTOOLS-A-05-00-00-00A-040A-D	C	2019-12-13	4	All
s1kd-appcheck - Description	S1KDTOOLS-A-11-00-00-00A-040A-D	C	2019-10-25	8	All
s1kd-brexcheck - Description	S1KDTOOLS-A-04-00-00-00A-040A-D	C	2019-10-04	6	All

Document title	Data module code		Issue date	No. of pages	Applicable to
s1kd-refs - Description	S1KDTOOLS-A-25-00-00-00A-040A-D	C	2019-12-13	5	All
s1kd-repcheck - Description	S1KDTOOLS-A-18-00-00-00A-040A-D	C	2019-12-13	3	All
s1kd-validate - Description	S1KDTOOLS-A-02-00-00-00A-040A-D	C	2019-10-25	3	All
s1kd-acronyms - Description	S1KDTOOLS-A-20-00-00-00A-040A-D	C	2019-09-06	3	All
s1kd-aspp - Description	S1KDTOOLS-A-26-00-00-00A-040A-D	C	2019-10-04	7	All
s1kd-flatten - Description	S1KDTOOLS-A-23-00-00-00A-040A-D	C	2019-11-15	3	All
s1kd-fmgen - Description	S1KDTOOLS-A-33-00-00-00A-040A-D	C	2019-11-15	5	All
s1kd-icncatalog - Description	S1KDTOOLS-A-32-00-00-00A-040A-D	C	2019-10-04	7	All
s1kd-index - Description	S1KDTOOLS-A-28-00-00-00A-040A-D	C	2019-09-06	3	All
s1kd-instance - Description	S1KDTOOLS-A-03-00-00-00A-040A-D	C	2019-12-13	19	All
s1kd-neutralize - Description	S1KDTOOLS-A-14-00-00-00A-040A-D	C	2019-10-25	2	All
s1kd-syncrefs - Description	S1KDTOOLS-A-01-00-00-00A-040A-D	C	2019-09-06	2	All
s1kd-uom - Description	S1KDTOOLS-A-10-00-00-00A-040A-D	C	2019-12-13	9	All

Highlights

The listed changes are introduced in issue 024, dated 2019-12-13, of this publication.

Data module code	Reason for update
S1KDTOOLS-A-08-00-00-00A-040A-D	Add -c (--content), -g (--guess-prefix), -L (--list), -T (--transform) and -x (--xpath) options.
S1KDTOOLS-A-25-00-00-00A-040A-D	Add -w (--where-used), -L (--dml), -S (--smc) and -t (--format) options.
S1KDTOOLS-A-03-00-00-00A-040A-D	Add -7 (--dry-run) option.
S1KDTOOLS-A-10-00-00-00A-040A-D	Change behaviour of the -F (--format) option.
	Add -D (--duplicate-format) option.

Table of contents

The listed documents are included in issue 024, dated 2019-12-13, of this publication.

Document title	Document identifier	Issue date	No. of pages	Applicable to
s1kd-newcom - Description	S1KDTOOLS-A-16-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newddn - Description	S1KDTOOLS-A-17-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newdm - Description	S1KDTOOLS-A-07-00-00-00A-040A-D	2019-12-13	7	All
s1kd-newdml - Description	S1KDTOOLS-A-21-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newimf - Description	S1KDTOOLS-A-13-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newpm - Description	S1KDTOOLS-A-12-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newsmc - Description	S1KDTOOLS-A-35-00-00-00A-040A-D	2019-10-04	3	All
s1kd-newupf - Description	S1KDTOOLS-A-31-00-00-00A-040A-D	2019-10-04	2	All
Tools for authoring and managing data				
s1kd-addicn - Description	S1KDTOOLS-A-27-00-00-00A-040A-D	2019-09-06	2	All
s1kd-ls - Description	S1KDTOOLS-A-06-00-00-00A-040A-D	2019-10-25	3	All
s1kd-ref - Description	S1KDTOOLS-A-08-00-00-00A-040A-D	2019-12-13	6	All
s1kd-metadata - Description	S1KDTOOLS-A-09-00-00-00A-040A-D	2019-12-13	4	All
s1kd-mvref - Description	S1KDTOOLS-A-19-00-00-00A-040A-D	2019-09-06	2	All
s1kd-sns - Description	S1KDTOOLS-A-34-00-00-00A-040A-D	2019-09-06	2	All
s1kd-transform - Description	S1KDTOOLS-A-15-00-00-00A-040A-D	2019-09-06	2	All
s1kd-upissue - Description	S1KDTOOLS-A-05-00-00-00A-040A-D	2019-12-13	4	All
Tools for validating data				

Document title	Document identifier	Issue date	No. of pages	Applicable to
s1kd-appcheck - Description	S1KDTOOLS-A-11-00-00-00A-040A-D	2019-10-25	8	All
s1kd-brexcheck - Description	S1KDTOOLS-A-04-00-00-00A-040A-D	2019-10-04	6	All
s1kd-refs - Description	S1KDTOOLS-A-25-00-00-00A-040A-D	2019-12-13	5	All
s1kd-repcheck - Description	S1KDTOOLS-A-18-00-00-00A-040A-D	2019-12-13	3	All
s1kd-validate - Description	S1KDTOOLS-A-02-00-00-00A-040A-D	2019-10-25	3	All
Tools for delivering data				
s1kd-acronyms - Description	S1KDTOOLS-A-20-00-00-00A-040A-D	2019-09-06	3	All
s1kd-aspp - Description	S1KDTOOLS-A-26-00-00-00A-040A-D	2019-10-04	7	All
s1kd-flatten - Description	S1KDTOOLS-A-23-00-00-00A-040A-D	2019-11-15	3	All
s1kd-fmgen - Description	S1KDTOOLS-A-33-00-00-00A-040A-D	2019-11-15	5	All
s1kd-icncatalog - Description	S1KDTOOLS-A-32-00-00-00A-040A-D	2019-10-04	7	All
s1kd-index - Description	S1KDTOOLS-A-28-00-00-00A-040A-D	2019-09-06	3	All
s1kd-instance - Description	S1KDTOOLS-A-03-00-00-00A-040A-D	2019-12-13	19	All
s1kd-neutralize - Description	S1KDTOOLS-A-14-00-00-00A-040A-D	2019-10-25	2	All
s1kd-syncrefs - Description	S1KDTOOLS-A-01-00-00-00A-040A-D	2019-09-06	2	All
s1kd-uom - Description	S1KDTOOLS-A-10-00-00-00A-040A-D	2019-12-13	9	All

List of abbreviations

BREX	Business Rules EXchange
CIR	Common Information Repository
CSDB	Common Source Database
PCT	Product Cross-reference Table
SNS	Standard Numbering System

s1kd-tools

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
https://github.com/kibook/S1000D-XSL-Stylesheets	S1000D XSL stylesheets
https://github.com/kibook/s1kd-tools	s1kd-tools

Description

1 General

s1kd-tools are a set of small tools for manipulating S1000D data. They are maintained at https://github.com/kibook/s1kd-tools.

This publication is meant to serve as an example of an S1000D data set produced using these tools. The stylesheets used to produce this PDF can be found at https://github.com/kibook/S1000D-XSL-Stylesheets

s1kd-tools

Introduction

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

This document gives a basic overview of the relationship of the s1kd-tools to S1000D, and defines some common terms used throughout the s1kd-tools documentation.

2 S1000D

S1000D is "an international specification for the procurement and production of technical publications", part of the S-Series of ILS specifications. The main focus of S1000D is the breakdown and classification of documents in to individual components, called "data modules", which can be re-used in multiple publications. These data modules are typically authored using a set of provided XML schemas, allowing them to be automatically managed in a CSDB and validated against a defined set of project "business rules".

3 s1kd-tools

The **s1kd-tools** are a set of small tools for creating and manipulating S1000D data. They are designed to be used as a standalone method of maintaining a simple S1000D CSDB, in

conjunction with a more typical version control system such as Git or SVN, as a backend to implement a more complex S1000D CSDB, or to support an existing S1000D CSDB already in use by a project.

4 CSDB

Common Source Databases can be implemented in any number of ways. For the purposes of the s1kd-tools, the CSDB is simply a directory within a filesystem. Use of the "File-based transfer" file naming conventions in Chap 7 of the S1000D specification are recommended, and most of the tools will use these conventions when creating or listing CSDB objects represented by files. In order to use these tools in conjuction with other implementations of CSDBs, a project can make use of "transfer packages" also described in Chap 7 to facilitate interchange between the two kinds of CSDB.

5 Relationship to the S1000D process

The s1kd-tools can support multiple parts of the basic S1000D process:

1 **Generation:** The generation of new CSDB objects is supported by the **s1kd-dmrl** tool and the **s1kd-new*** set of tools. These provide two methods of creating objects, either using a data management requirements list (DMRL) or a more on-the-fly approach using the s1kd-new* tools directly.

   The **s1kd-defaults** tool is used to manage the files which contain default metadata for new CSDB objects.

2 **Authoring:** These tools support the authoring process.

   The **s1kd-addicn** tool creates the notation and entity elements to reference an ICN in a data module.

   The **s1kd-ls** tool lists data modules within a directory.

   The **s1kd-metadata** tool lists and edits S1000D metadata on CSDB objects.

   The **s1kd-mvref** tool changes references to one object into references to another.

   The **s1kd-ref** tool can be used to quickly insert references to other CSDB objects.

   The **s1kd-sns** tool can be used to organize the CSDB using a given SNS structure.

   The **s1kd-transform** tool applies XSLT transformations to CSDB objects.

   The **s1kd-upissue** tool moves CSDB objects through the standard S1000D workflow, between "inwork" (draft) and "official" states.

3 **Validation:** These tools all validate different aspects of CSDB objects.

   The **s1kd-appcheck** tool validates the applicability of CSDB objects.

   The **s1kd-brexcheck** tool validates CSDB objects against a business rules exchange (BREX) data module, which contains the project-defined computable business rules.

The **s1kd-refs** tool lists references in a CSDB object to generate a list of dependencies on other CSDB objects.

The **s1kd-repcheck** tool validates CIR references in CSDB objects.

The **s1kd-validate** tool validates CSDB objects according to their S1000D schema and general correctness as XML documents.

4 **Publication:** These tools support the production of publications from a CSDB.

The **s1kd-acronyms** tool can automatically mark up acronyms within data modules, and can also generate lists of acronyms marked up within data modules.

The **s1kd-aspp** tool preprocesses applicability statements in a data module, generating display text and "presentation" applicability statements.

The **s1kd-flatten** tool flattens a publication module and referenced data modules in to a single "deliverable" file for a publishing system.

The **s1kd-fmgen** tool generates front matter data module content from a publication module.

The **s1kd-icncatalog** tool resolves ICN references in objects.

The **s1kd-index** tool flags index keywords in a data module based on a user-defined list.

The **s1kd-instance** tool produces "instances" of CSDB objects using applicability filtering and/or common information repositories (CIRs).

The **s1kd-neutralize** tool generates IETP neutral metadata for CSDB objects.

The **s1kd-syncrefs** tool generates the References table within data modules.

The **s1kd-uom** tool converts units of measure used in data modules.

s1kd-tools

Building, installing and uninstalling

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
http://khzae.net/1/s1000d/s1kd-tools/releases/latest	
	libxml2, libxslt, libexslt
	pandoc
	s1kd2db

Description

1 General

There are multiple ways to install the s1kd-tools:

– using a package manager and the pre-compiled Debian (.deb) or Red Hat (.rpm) packages

– building from source

2 Using a package manager

Debian (.deb) and Red Hat (.rpm) packages are provided to easily install, upgrade or uninstall the s1kd-tools on Linux systems using a package manager. The examples below focus on the standard `dpkg` (for Debian-based distributions) and `rpm` (for Red Hat-based distributions).

2.1 Installing

You can download the latest release of the s1kd-tools from http://khzae.net/1/s1000d/s1kd-tools/releases/latest. Then use one of the following commands to install it:

Debian:

```
# dpkg -i s1kd-tools_[version]_[arch].deb
```

Red Hat:

```
# rpm -i s1kd-tools.[version].[arch].rpm
```

2.2 Uninstalling

To uninstall using the package manager, use one of the following commands:

Debian:

```
# dpkg -r s1kd-tools
```

Red Hat:

```
# rpm -e s1kd-tools
```

3 Building from source

3.1 Requirements

To build the executables:

- coreutils and binutils
- xxd
- libxml2, libxslt, libexslt

To build the documentation from source:

- s1kd2db
- pandoc

3.2 Building and installing

Run the following commands to build the executables, and install both the executables and documentation:

```
$ make
# make install
```

To change where these are installed, specify the PREFIX make variable. The default value of this variable is /usr/local.

For example:

```
# make PREFIX=/usr install
```

3.3 Uninstalling

To uninstall the executables and documentation:

```
# make uninstall
```

Remember to specify the PREFIX make variable if a different prefix was used during installation.

s1kd-tools

Usage examples

Table of contents

List of tables

List of figures

References

Table 1 References

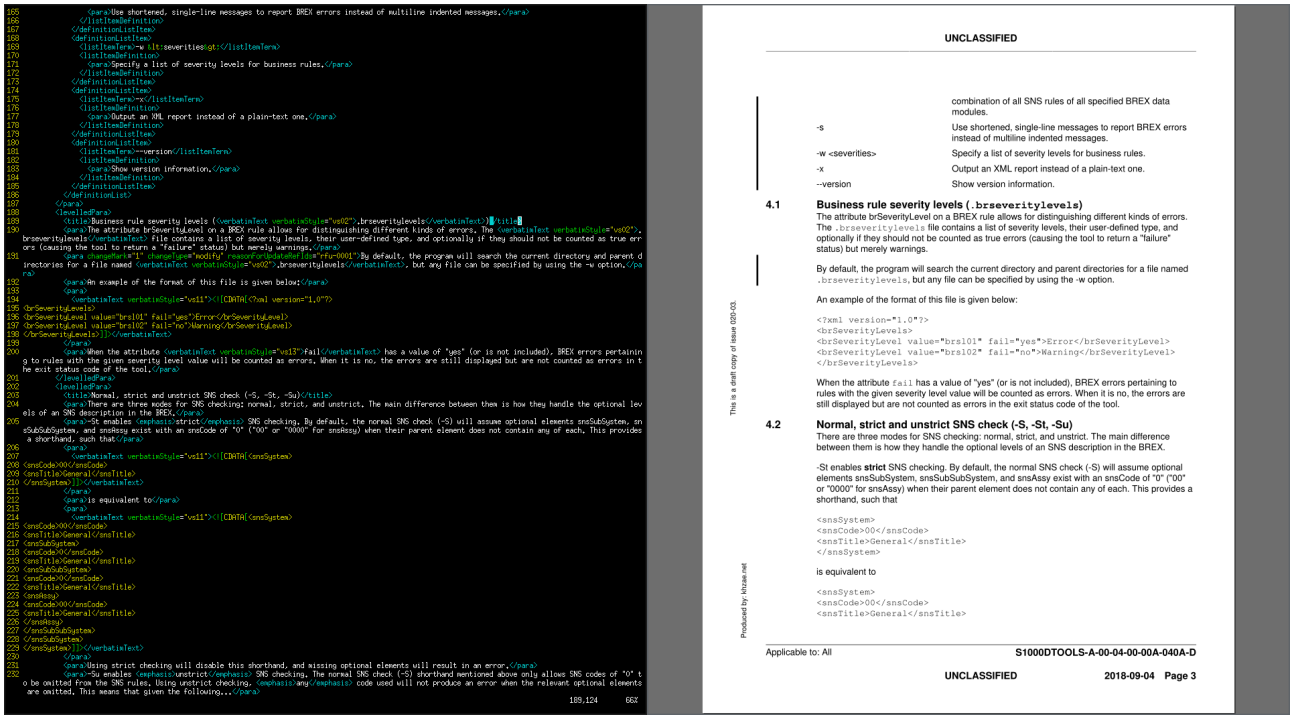
Data module/Technical publication	Title
None	

Description

1 General

This document provides examples of the usage of the **s1kd-tools**.

The sample commands have been written as they would be used on a Linux or other Unix-like system, but should work more-or-less the same on most operating systems. OS-specific commands used in examples (e.g., `mkdir`) may need to be adapted.



ICN-S1KDTOOLS-A-000000-A-KHZAE-00002-A-001-01

Fig 1 Example - Authoring with Vim + MuPDF

2 Initial setup

This first step is to create a folder for the new S1000D project. Example:

```
$ mkdir myproject
```

```
$ cd myproject
```

After that, you should create two files: `.defaults` and `.dmtypes`. These files can be created automatically using the **s1kd-defaults** tool to initialize the new CSDB:

```
$ s1kd-defaults -i
```

Afterwards, these files can be edited to customize them for your project. More information on the contents of these files is provided below.

Note

If the tools are run in a directory that does not have these configuration files, they will search for them in the parent directories to find the top of the CSDB directory tree.

2.1 `.defaults` file

The `.defaults` file is used by all of the s1kd-new* tools. It provides default values for various S1000D metadata. The `.defaults` file can be written in either a simple text format or an XML format.

Example of simple text format:

```
languageIsoCode            en
countryIsoCode             CA
responsiblePartnerCompany  khzae.net
originator                 khzae.net
brex                       MYPRJ-A-00-00-00-00A-022A-D
techName                   My project
```

Example of XML format:

```
<?xml version="1.0"?>
<defaults>
<default ident="languageIsoCode" value="en"/>
<default ident="countryIsoCode" value="CA"/>
<default ident="responsiblePartnerCompany" value="khzae.net"/>
<default ident="originator" value="khzae.net"/>
<default ident="brex" value="MYPRJ-A-00-00-00-00A-022A-D"/>
<default ident="techName" value="My project"/>
</defaults>
```

2.2 `.dmtypes` file

The `.dmtypes` file is used by the **s1kd-newdm** tool. It contains a list of information codes and associated info names and schemas to be used when creating new data modules. Like the `.defaults` file, it can be written using either the simple text format or XML format.

Example of simple text format:

```
009  frontmatter  Table of contents
022  brex         Business rules exchange
040  descript     Description
```

```
130  proced       Normal operation
```

Example of XML format:

```
<?xml version="1.0"?>
<dmtypes>
<type infoCode="009" infoName="Table of contents"
schema="frontmatter"/>
<type infoCode="022" infoName="Business rules exchange"
schema="brex"/>
<type infoCode="040" infoName="Description"
schema="descript"/>
<type infoCode="130" infoName="Normal operation"
schema="proced"/>
</dmtypes>
```

The s1kd-newdm tool contains a default set of information code definitions. This can be used to create a default `.dmtypes` file by use of the `-.` (simple text format) or `-,` (XML) options:

```
$ s1kd-newdm -, > .dmtypes
```

The generated `.dmtypes` file can then be customized to fit your project.

3 Creating the DMRL and populating the CSDB

The next step is to prepare the Data Management Requirements List (DMRL) for the project. The DMRL will contain a list of all the CSDB objects initially required by your project, and can be used to automatically populate your CSDB.

If you do not already have a DMRL, the **s1kd-newdml** tool can be used to create a new one:

```
$ s1kd-newdml -# MYPRJ-NCAGE-C-2017-00001
```

This would create the file `DML-MYPRJ-NCAGE-C-2017-00001_000-01.XML` in your CSDB folder.

3.1 Adding DMRL entries

Each entry in the DMRL describes a data module that is planned to be created:

```
<dmlContent>
<dmlEntry>
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="MYPRJ" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00"
disassyCode="00" disassyCodeVariant="A" infoCode="040"
infoCodeVariant="A" itemLocationCode="D"/>
</dmRefIdent>
<dmRefAddressItems>
<dmTitle>
<techName>My project</techName>
<infoName>Description</infoName>
```

```
</dmTitle>
</dmRefAddressItems>
</dmRef>
<security securityClassification="01"/>
<responsiblePartnerCompany>
<enterpriseName>khzae.net</enterpriseName>
</responsiblePartnerCompany>
</dmlEntry>
...
</dmlContent>
```

The XML for the `dmRef` of each entry can be quickly generated using the **s1kd-ref** tool:

```
$ s1kd-ref DMC-MYPRJ-A-00-00-00-00A-040A-D
```

3.2 Populating the CSDB from the DMRL

Once the DMRL is prepared, the **s1kd-dmrl** tool can be used to automatically populate the CSDB based on the CSDB objects listed in the DMRL:

```
$ s1kd-dmrl DML-MYPRJ-NCAGE-C-2017-00001_000-01.XML
```

Information not included in the DMRL entry for a CSDB object is pulled from the `.defaults` file (and the `.dmtypes` file for data modules).

The DMRL should be updated throughout the lifecycle of a project. When new entries are added, simply use the **s1kd-dmrl** tool again to create the newly added data modules. Already existing data modules will not be overwritten, unless the -f option is specified. The -q option will suppress those messages indicating that a data module that already exists will not be overwritten:

```
$ s1kd-dmrl -q DML-MYPRJ-NCAGE-C-2017-00001_000-02.XML
```

3.3 Creating CSDB objects on-the-fly

Data modules and other CSDB objects can also be created in an "on-the-fly" manner, without the use of a DMRL, by invoking the s1kd-new* set of tools directly, as with s1kd-newdml above. For example, to create a new data module:

```
$ s1kd-newdm -# MYPRJ-A-00-00-00-00A-040A-D
```

This would create the file `DMC-MYPRJ-A-00-00-00-00A-040A-D_000-01_EN-CA.XML` in your CSDB folder.

Each of the s1kd-new* tools has various options for setting specific metadata, and information not included as arguments to these commands is pulled from the `.defaults` and `.dmtypes` files.

4 Data module workflow

Data modules are put through the general S1000D workflow with the **s1kd-upissue** tool. Whenever a data module will be changed, the s1kd-upissue tool should first be used to indicate the forthcoming change, creating the next inwork issue of the data module.

4.1 Inwork data modules

To increment the inwork issue of a data module, the s1kd-upissue tool is called without any additional options:

```
$ s1kd-upissue DMC-MYPRJ-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
```

Assuming this data module was just created, it would be incremented from initial inwork issue 000-01 to initial inwork issue 000-02. After upissuing, make the changes. For example:

DMC-MYPRJ-A-00-00-00-00A-040A-D_000-01_EN-CA.XML:

```
<content>
<description>
<levelledPara>
<title>General</title>
<para>This is my project.</para>
</levelledPara>
</description>
</content>
```

DMC-MYPRJ-A-00-00-00-00A-040A-D_000-02_EN-CA.XML:

```
<content>
<description>
<levelledPara>
<title>General</title>
<para>This is my project.</para>
<para>My project is maintained using S1000D.</para>
</levelledPara>
</description>
</content>
```

4.2 Making data modules official

Before a data module can be made official, it must be validated. This means:

– It is a valid XML file
– It is valid according to the relevant S1000D schema
– It is valid according to the relevant business rules
– Any applicability filtering applied will not affect the above
– The actual narrative (content) is correct

4.2.1 Validating against the schema

The first two points can be verified with the **s1kd-validate** tool. This tool will indicate any problems with the data module in terms of XML syntax and its correctness regarding its S1000D schema:

```
$ s1kd-validate DMC-MYPRJ-A-00-00-00-00A-040A-D_000-03_EN-CA.XML
```

4.2.2 Validating against a BREX data module

The third point can be verified using the **s1kd-brexcheck** tool. This tool will indicate any places where a data module violates computable business rules as specified in a Business Rules Exchange (BREX) data module.

```
$ s1kd-brexcheck DMC-MYPRJ-A-00-00-00-00A-040A-D_000-03_EN-CA.XML
```

The BREX allows a project to customize S1000D, for example, by disallowing certain elements or attributes:

```
<structureObjectRule>
<objectPath allowedObjectFlag="0">//emphasis</objectPath>
<objectUse>The emphasis element is not allowed.</objectUse>
</structureObjectRule>
```

Or by tailoring the allowed values of certain elements or attributes:

```
<structureObjectRule>
<objectPath allowedObjectFlag="2">
//@securityClassification
</objectPath>
<objectUse>
The security classification must be 01 (Unclassified)
or 02 (Classified).
</objectUse>
<objectValue valueAllowed="01">Unclassified</objectValue>
<objectValue valueAllowed="02">Classified</objectValue>
</structureObjectRule>
```

Each data module references the BREX it should be checked against, and BREX data modules can reference other BREX data modules to create a layered set of business rules, for example, Project-related rules and Organization-related rules.

Unless otherwise specified, data modules will reference the S1000D default BREX, which contains a base set of business rules.

To get started with your project's own business rules, you can create a simple BREX data module based on the current defaults of your CSDB using the -B option of the s1kd-newdm tool:

```
$ s1kd-newdm -B# MYPRJ-A-00-00-00-00A-022A-D
```

This will use the customized `.defaults` and `.dmtypes` files to generate a basic set of business rules.

4.2.3 Checking applicability

The fourth point can be tested using the **s1kd-appcheck** tool:

```
$ s1kd-appcheck DMC-MYPRJ-A-00-00-00-00A-040A-D_000-03_EN-CA.XML
```

The S1000D applicability model allows for conditional processing to be applied both to whole data modules as well as parts of a data module. However, this latter functionality means that, if

elements are removed as part of applicability filtering, the validity of the data module in regards to the S1000D schema and business rules can change.

The s1kd-appcheck tool can report product attribute or condition assignments which would cause the data module to become invalid after filtering.

4.2.4 Quality assurance verification

In contrast to the first four points, which can be verified automatically, the last point is generally not an automatic process, and involves quality assurance testing by a human. That a data module has been first or second QA tested can be indicated with the s1kd-upissue tool:

```
$ s1kd-upissue -1 tabtop -2 ttandoo ...
```

Once the data module is validated, the s1kd-upissue tool is used to make it official with the `-i` option:

```
$ s1kd-upissue -i DMC-MYPRJ-A-00-00-00-00A-040A-D_000-03_EN-CA.XML
```

4.3 Changes to official data modules

When a change must be made to an official data module (for example, as a result of feedback), the s1kd-upissue tool is used again to bring the data module back to the inwork state:

```
$ s1kd-upissue DMC-MYPRJ-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
```

Changes between official issues of a data module are indicated with reasons for update and change marking. For example:

DMC-MYPRJ-A-00-00-00-00A-040A-D_001-00_EN-CA.XML:

```
<content>
<description>
<levelledPara>
<title>General</title>
<para>This is my project.</para>
<para>My project is maintained using S1000D.</para>
</levelledPara>
</description>
</content>
```

DMC-MYPRJ-A-00-00-00-00A-040A-D_001-01_EN-CA.XML:

```
<dmStatus issueType="changed">
<!-- ...... -->
<reasonForUpdate id="rfu-0001">
<simplePara>Added reference to tools used.</simplePara>
</reasonForUpdate>
</dmStatus>
<!-- ...... -->
<content>
<description>
```

```
<levelledPara>
<title>General</title>
<para>This is my project.</para>
<para changeType="modify" changeMark="1"
reasonForUpdateRefIds="rfu-0001">My project is maintained using
S1000D and s1kd-tools.</para>
</levelledPara>
</description>
</content>
```

Reasons for update from the previous official issue are automatically removed when upissuing to the first inwork issue.

4.4 Deleting data modules

The basic cycle continues until a data module is deleted. "Deleting" a data module is a special case of upissuing:

```
$ s1kd-upissue -is deleted ...
```

The data module is upissued to the next official issue, and it's issue type is set to "`deleted`".

Deleted data modules may be reinstated later in a similar way:

```
$ s1kd-upissue -s rinstate-status ...
```

The data module is upissued to the next inwork issue, and the issue type is set to one of the "`rinstate-x`" types.

5 Building publications

S1000D publications are managed by use of publication modules. Like data modules, publication modules may be created as part of the project's DMRL:

```
<dmlEntry>
<pmRef>
<pmRefIdent>
<pmCode modelIdentCode="MYPRJ" pmIssuer="12345" pmNumber="00001"
pmVolume="00"/>
</pmRefIdent>
<pmRefAddressItems>
<pmTitle>My publication</pmTitle>
</pmRefAddressItems>
</pmRef>
<responsiblePartnerCompany>
<enterpriseName>khzae.net</enterpriseName>
</responsiblePartnerCompany>
</dmlEntry>
```

or "on-the-fly" with the **s1kd-newpm** tool:

```
$ s1kd-newpm -# MYPRJ-12345-00001-00
```

5.1 Publication module content

The publication module lays out the hierarchical structure of the data modules in a publication:

```
<content>
<pmEntry>
<pmEntryTitle>Front matter</pmEntryTitle>
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="MYPRJ" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="001" infoCodeVariant="A"
itemLocationCode="D"/>
</dmRefIdent>
<dmRefAddressItems>
<dmTitle>
<techName>My project</techName>
<infoName>Title page</infoName>
</dmTitle>
</dmRefAddressItems>
</dmRef>
</pmEntry>
<pmEntry>
<pmEntryTitle>General info</pmEntryTitle>
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="MYPRJ" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
itemLocationCode="D"/>
</dmRefIdent>
<dmRefAddressItems>
<dmTitle>
<techName>My project</techName>
<infoName>Description</infoName>
</dmTitle>
</dmRefAddressItems>
</dmRef>
</pmEntry>
</content>
```

5.2 Creating a customized publication

The S1000D applicability model and the **s1kd-instance** tool enable the creation of customized publications, which are filtered for a particular customer or product. For example, a data module may contain applicabilty for two versions of a product:

```
<para>
This is some common information about the product.
</para>
<para applicRefId="app-versionA">
```

```
This information only applies to version A.
</para>
<para applicRefId="app-versionB">
This information only applies to version B.
</para>
```

When you deliver this data module to a customer with Version B, you can exclude information which is not applicable to them by filtering it:

```
$ s1kd-instance -s version:prodattr=B <DM>
```

To filter a whole publication, use the -O option of the s1kd-instance tool to output multiple filtered objects into a directory:

```
$ s1kd-instance -s version:prodattr=B -O customerB DMC-*.XML
```

The newly created `customerB` directory will contain the filtered versions of these data modules.

If your CSDB contains multiple, separate publications, the **s1kd-refs** tool can be used to select only those data modules which apply to a particular publication module:

```
$ s1kd-refs -s <PM> |
> xargs s1kd-instance -s version:prodattr=B -O customerB
```

The above command will filter the publication module and all included data modules, and output the resulting objects to the `customerB` directory.

5.3 Creating a script for publishing

The publishing process will often involve many different steps, and many different tools, so it's a good idea to create a script to automate it. Below is an example of a script which publishes a CSDB for a given product serial number:

```
#!/bin/sh

# Usage: sh build.sh <zip> <csdb> <serialno>
zip="$1"
csdb="$2"
serialno="$3"

# Create a temporary directory.
tmp=$(mktemp -d)

# Copy all CSDB objects to the temp directory. The CSDB objects
# are filtered for a given serial number.
s1kd-ls -d "$csdb" |
  xargs s1kd-instance -O "$tmp" -s serialno:prodattr="$serialno"

# Synchronize references in the filtered DMs. This is necessary
# since some references may have been removed during filtering.
s1kd-ls -D "$tmp" |
```

```
  xargs s1kd-syncrefs -f

# Create the ZIP package.
zip -jr "$zip" "$tmp"

# Clean up the temp directory.
rm -r "$tmp"
```

6 Use with other version control systems

The issue/inwork numbers and S1000D file naming conventions as seen above provide a basic form of version control. In this case, each file represents a single issue of a CSDB object, and multiple files together represent the whole logical object. For example, all of the following files represent different versions of the same object:

– `DMC-MYPRJ-A-00-00-00-00A-040A-D_000-01_EN-CA.XML`
– `DMC-MYPRJ-A-00-00-00-00A-040A-D_000-02_EN-CA.XML`
– `DMC-MYPRJ-A-00-00-00-00A-040A-D_001-00_EN-CA.XML`

However, if you prefer to use an existing version control system such as Git or SVN, it is often more useful for each file to represent a whole object.

The s1kd-tools support an alternate naming convention for this case. Specifying the -N option to certain tools will omit the issue and inwork numbers from filenames of CSDB objects. Taking the s1kd-newdm tool example from above, but adding the -N option as follows:

```
$ s1kd-newdm -N -# MYPRJ-A-00-00-00-00A-040A-D
```

would create the file `DMC-MYPRJ-A-00-00-00-00A-040A-D_EN-CA.XML` in your CSDB folder. The s1kd-upissue tool works similarly:

```
$ s1kd-upissue -N -i DMC-MYPRJ-A-00-00-00-00A-040A-D_EN-CA.XML
```

The issue and inwork numbers are updated in the XML metadata, but instead of creating a new file, the original is overwritten. The previous inwork issues are therefore stored as part of the external version control system's history, rather than as individual files.

s1kd-tools

.defaults file identifiers

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

This document contains an alphabetic index of all valid `.defaults` file identifiers, and a description of the values they may be assigned. It also contains examples of a `.defaults` file using all identifiers in both the XML format and simple text format.

2 Alphabetic index

Table 2 `.defaults` file - Identifier value descriptions

Identifier	Value description
`act`	Data module code of ACT data module
`assyCode`	2 to 4 alphanumeric characters
`authorization`	string
`brex`	Data module code of BREX data module

Identifier	Value description
city	string (Sender city)
commentPriorityCode	cp01-cp99
commentType	Q, I, or R
countryIsoCode	ISO 2-character country code
country	string (Sender country)
disassyCodeVariant	1 to 3 alphanumeric characters
disassyCode	2 alphanumeric characters
dmlType	C, P, or S
includePrevSnsTitle	true or false
infoCodeVariant	1 alphanumeric character
infoCode	3 alphanumeric characters
infoName	string
infoNameVariant	string
inWork	2 digits
issueNumber	3 digits
issueType	S1000D issue type (new, changed, ...)
issue	S1000D issue number (5.0, 4.2, 4.1, ...)
itemLocationCode	A, B, C, D, or T
languageIsoCode	2 to 3 character ISO language code
learnCode	3 alphanumeric characters
learnEventCode	A, B, C, D, or E
maintainedSns	string
modelIdentCode	1 to 14 alphanumeric characters
originatorCode	5-character NCAGE code
originator	string
pmIssuer	5-character NCAGE code
pmNumber	5 alphanumeric characters
pmVolume	2 digits

Identifier	Value description
receiver	string
receiverCity	string
receiverCountry	string
receiverIdent	5-character NCAGE code
remarks	string
responsiblePartnerCompanyCode	5-character NCAGE code
responsiblePartnerCompany	string
schema	URI
scormContentPackageIssuer	5-character NCAGE code
scormContentPackageNumber	5 alphanumeric characters
scormContentPackageVolume	2 digits
securityClassification	2 digits
sender	string
senderIdent	5-character NCAGE code
seqNumber	00001-99999
skillLevelCode	sk01-sk99
sns	Data module code of BREX data module
subSubSystemCode	1 alphanumeric character
subSystemCode	1 alphanumeric character
systemCode	2 to 3 alphanumeric characters
systemDiffCode	1 to 4 alphanumeric characters
techName	string
templates	Path to custom XML templates directory
yearOfDataIssue	4 digits

3 Example - XML format

```
<?xml version="1.0"?>
<defaults>
<default ident="act" value="MYPRJ-A-00-00-00-00A-00WA-D"/>
<default ident="assyCode" value="00"/>
```

```
<default ident="authorization" value="khzae.net"/>
<default ident="brex" value="MYPRJ-A-00-00-00-00A-022A-D"/>
<default ident="city" value="Toronto"/>
<default ident="commentPriorityCode" value="cp01"/>
<default ident="commentType" value="Q"/>
<default ident="countryIsoCode" value="CA"/>
<default ident="country" value="Canada"/>
<default ident="disassyCodeVariant" value="A"/>
<default ident="disassyCode" value="00"/>
<default ident="dmlType" value="C"/>
<default ident="includePrevSnsTitle" value="true"/>
<default ident="infoCodeVariant" value="A"/>
<default ident="infoCode" value="258"/>
<default ident="infoName" value="Other procedure to clean"/>
<default ident="infoNameVariant" value="Clean with water"/>
<default ident="inWork" value="01"/>
<default ident="issueNumber" value="000"/>
<default ident="issueType" value="new"/>
<default ident="issue" value="5.0"/>
<default ident="itemLocationCode" value="D"/>
<default ident="languageIsoCode" value="en"/>
<default ident="learnCode" value="H10"/>
<default ident="learnEventCode" value="A"/>
<default ident="maintainedSns" value="General sea vehicles"/>
<default ident="modelIdentCode" value="MYPRJ"/>
<default ident="omitIssueInfo" value="true"/>
<default ident="originatorCode" value="12345"/>
<default ident="originator" value="khzae.net"/>
<default ident="pmIssuer" value="12345"/>
<default ident="pmNumber" value="00000"/>
<default ident="pmVolume" value="00"/>
<default ident="receiver" value="khzae.net"/>
<default ident="receiverCity" value="Toronto"/>
<default ident="receiverCountry" value="Canada"/>
<default ident="receiverIdent" value="12345"/>
<default ident="remarks" value="Comments on a data module"/>
<default ident="responsiblePartnerCompanyCode" value="12345"/>
<default ident="responsiblePartnerCompany" value="khzae.net"/>
<default ident="schema" value="descript.xsd"/>
<default ident="securityClassification" value="01"/>
<default ident="senderIdent" value="12345"/>
<default ident="seqNumber" value="00001"/>
<default ident="skillLevelCode" value="sk01"/>
<default ident="sns" value="MYPRJ-A-00-00-00-00A-022A-D"/>
<default ident="subSubSystem" value="0"/>
<default ident="subSystem" value="0"/>
<default ident="systemCode" value="00"/>
<default ident="techName" value="My project"/>
```

```
<default ident="templates" value="/usr/share/s1kd-tools/templ"/>
<default ident="yearOfDataIssue" value="2017"/>
</defaults>
```

4 Example - Simple text format

act	MYPRJ-A-00-00-00-00A-00WA-D
assyCode	00
authorization	khzae.net
brex	MYPRJ-A-00-00-00-00A-022A-D
city	Toronto
commentPriorityCode	cp01
commentType	Q
countryIsoCode	CA
country	Canada
disassyCodeVariant	A
disassyCode	00
dmlType	C
includePrevSnsTitle	true
infoCodeVariant	A
infoCode	258
infoName	Other procedure to clean
infoNameVariant	Clean with water
inWork	01
issueNumber	000
issueType	new
issue	5.0
itemLocationCode	D
languageIsoCode	en
learnCode	H10
learnEventCode	A
maintainedSns	General sea vehicles
modelIdentCode	MYPRJ
omitIssueInfo	true
originatorCode	12345
originator	khzae.net
pmIssuer	12345
pmNumber	00000
pmVolume	00
receiver	khzae.net
receiverCity	Toronto
receiverCountry	Canada
receiverIdent	12345
remarks	Comments on a data module
responsiblePartnerCompanyCode	12345
responsiblePartnerCompany	khzae.net
schema	descript.xsd
securityClassification	01

sender	khzae.net
senderIdent	12345
seqNumber	00001
skillLevelCode	sk01
sns	MYPRJ-A-00-00-00-00A-022A-D
subSubSystem	0
subSystem	0
systemCode	00
techName	My project
templates	/usr/share/s1kd-tools/templ
yearOfDataIssue	2017

s1kd-tools

Issue compatibility

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

This document summarizes the compatibility between the s1kd-tools and each issue of the S1000D specification for which support is planned.

[empty]	Support is planned but not yet implemented.
X	Support is implemented.
/	Support is partially implemented.
~	Support is not planned. Usually this is because older issues of the specification did not cover the function of the tool.

Note

Although a tool may not directly support an issue of S1000D, it may still be possible to use with that issue.

For example, the s1kd-brexcheck tool is said not to support Issue 2.0 or Issue 2.1, because the BREX data module schema was not introduced until Issue 2.2. However, an Issue

2.2 or greater BREX data module can still be used to check Issue 2.0 or Issue 2.1 CSDB objects.

Table 2 Compatibility with supported issues of S1000D

Tool	5.0	4.2	4.1	4.0	3.0	2.3	2.2	2.1	2.0
s1kd-acronyms	X	X	X	X	X	X	X	X	X
s1kd-addicn	X	X	X	X	X	X	X	X	X
s1kd-appcheck	X	X	X	X	X	~	~	~	~
s1kd-aspp	X	X	X	X	X	~	~	~	~
s1kd-brexcheck	X	X	X	X	X	X	X	~	~
s1kd-defaults	X	X	X	X	X	X	X	X	X
s1kd-dmrl	X	X	X	X	X	X	X	X	X
s1kd-flatten	X	X	X	X	X	X	X	X	X
s1kd-fmgen	X	X	X	~	~	~	~	~	~
s1kd-icncatalog	X	X	X	X	X	X	X	X	X
s1kd-index	X	X	X	X	X	X	X	X	X
s1kd-instance	X	X	X	X	/	~	~	~	~
s1kd-ls	X	X	X	X	X	X	X	X	X
s1kd-metadata	X	X	X	X	X	X	X	X	X
s1kd-mvref	X	X	X	X	X	X	X	X	X
s1kd-neutralize	X	X	X	X	X	X	X	X	X
s1kd-newcom	X	X	X	X	X	X	X	X	X
s1kd-newddn	X	X	X	X	X	X	X	X	X
s1kd-newdm	X	X	X	X	X	X	X	X	X
s1kd-newdml	X	X	X	X	X	X	X	X	X
s1kd-newimf	X	X	~	~	~	~	~	~	~
s1kd-newpm	X	X	X	X	X	X	X	X	X
s1kd-newsmc	X	X	X	~	~	~	~	~	~
s1kd-newupf	X	X	X	~	~	~	~	~	~
s1kd-ref	X	X	X	X	X	X	X	X	X

Tool	5.0	4.2	4.1	4.0	3.0	2.3	2.2	2.1	2.0
s1kd-refs	X	X	X	X	X	X	X	X	X
s1kd-repcheck	X	X	X	X	X	X	~	~	~
s1kd-sns	X	X	X	X	~	~	~	~	~
s1kd-syncrefs	X	X	X	X	X	X	X	X	X
s1kd-transform	X	X	X	X	X	X	X	X	X
s1kd-uom	X	X	X	X	X	X	~	~	~
s1kd-upissue	X	X	X	X	X	X	X	X	X
s1kd-validate	X	X	X	X	X	X	X	X	X

s1kd-defaults

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-defaults** tool generates a basic `.defaults` file for a new CSDB, which is used by several of the other s1kd-tools to determine default values for S1000D metadata. It also provides a way to convert between the simple text and XML formats of the `.defaults`, `.dmtypes` and `.fmtypes` files.

2 Usage

```
s1kd-defaults [-DdFfisth?] [-b <BREX>] [-j <map>] [<file>...]
```

3 Options

-b, --brex <BREX>	Use the specified BREX data module to build the `.defaults` and `.dmtypes` files. This can be used both when initializing a new CSDB (-i) or either file can be generated from a BREX data module separately.
-D, --dmtypes	Convert a `.dmtypes` file.
-d, --defaults	Convert a `.defaults` file.
-F, --fmtypes	Convert a `.fmtypes` file.
-f, --overwrite	Overwrite the existing file after conversion.
-h, -?, --help	Show help/usage message.
-i, --init	Initialize a new CSDB by generating the `.defaults`, `.dmtypes` and `.fmtypes` files in the current directory.
-J, --dump-brexmap	Dump the default `.brexmap` file to stdout.
-j, --brexmap <map>	Use a custom `.brexmap` file to map a BREX DM to a `.defaults` or `.dmtypes` file.
-s, --sort	Sort the entries alphabetically for either file/output format.
-t, --text	Output using the simple text format. Otherwise, the XML format is used by default.
--version	Show version information.
<file>...	Names of files to convert. If none are specified, the default names of `.defaults` (for the -d option), `.dmtypes` (for the -D option) or `.fmtypes` (for the -F option) in the current directory are used.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.brexmap` file

This file specifies a mapping between BREX structure object rules and `.defaults` and `.dmtypes` files. The path to an object can be written in many different ways in a BREX rule, so the `.brexmap` file allows any project's BREX to be used to generate these files without having to modify the BREX data module itself.

By default, the program will search for a file named `.brexmap` in the current directory and parent directories, but any file can be specified using the -j option. If there is no `.brexmap` file and the -j option is not specified, a default mapping will be used.

Example of `.brexmap` file:

```
<brexMap>
<dmtypes path="//@infoCode"/>
<default path="//@languageIsoCode" ident="languageIsoCode"/>
<default path="//@countryIsoCode" ident="countryIsoCode"/>
</brexMap>
```

4 Examples

4.1 Initialize a new CSDB, using the XML format

```
$ mkdir mycsdb
$ cd mycsdb
$ s1kd-defaults -i
```

4.2 Initialize a new CSDB, using the simple text format

```
$ mkdir mycsdb
$ cd mycsdb
$ s1kd-defaults -ti
```

4.3 Generate a custom-named `.defaults` file

```
$ s1kd-defaults > custom-defaults.xml
```

4.4 Convert a simple text formatted file to XML

```
$ s1kd-defaults -df
```

4.5 Sort entries and output in text format

```
$ s1kd-defaults -dts custom-defaults.txt
```

s1kd-dmrl

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-dmrl** tool reads S1000D data management lists and creates CSBD objects for the entries specified using the s1kd-new* tools.

2 Usage

```
s1kd-dmrl [options] <DML>...
```

3 Options

-$, --issue <issue>	Specify which issue of S1000D to use when creating objects.
-@, --out <dir>	Create new objects in <dir>.
-%, --templates <dir>	Use XML templates in the specified directory instead of the built-in templates of each of the s1kd-new* tools.
-D, --dmtypes <path>	Specify the .dmtypes file name.

-d, --defaults <path>	Specify the .defaults file name.
-F, --fail	Fail on the first error generated by any of the s1kd-new* commands. Normally, errors with individual DMRL entries will be reported but the other entries will still be processed.
-f, --overwrite	Overwrite existing CSDB objects.
-h, -?, --help	Show help/usage message.
-m, --use-remarks	Use the remarks for an entry as the remarks for the new CSDB object.
-N, --omit-issue	Omit issue/in-work numbers from the filenames of created CSDB objects.
-q, --quiet	Do not report errors when any of the CSDB objects already exist.
-s, --commands	Do not create CSDB objects, only output the s1kd-new* commands to create them.
-v, --verbose	Print the filenames of newly created CSDB objects.
--version	Show version information.
<DML>...	One or more S1000D data management lists.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ s1kd-dmrl DML-EX-12345-C-2018-00001_001-00.XML
```

s1kd-newcom

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newcom** tool creates a new S1000D comment with the code and metadata specified.

2 Usage

```
s1kd-newcom [options]
```

3 Options

-#, --code <code>	The code of the comment, in the form of MODELIDENTCODE-SENDERIDENT-YEAR-SEQ-TYPE.
-$, --issue <issue>	Specify which issue of S1000D to use. Currently supported issues are: – 5.0 (default)

	– 4.2 – 4.1 – 4.0 – 3.0 – 2.3 – 2.2 – 2.1 – 2.0
-@, --out <path>	Save the new comment to <path>. If <path> is an existing directory, the comment will be created in it instead of the current directory. Otherwise, the comment will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in the specified directory instead of the built-in template. The template must be named `comment.xml` inside <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-b, --brex <BREX>	BREX data module code.
-C, --country <country>	The country ISO code of the new comment.
-c, --security <sec>	The security classification of the new comment.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	The issue date of the new comment in the form of YYYY-MM-DD.
-L, --language <lang>	The language ISO code of the new comment.
-m, --remarks <remarks>	Set the remarks for the new comment.
-o, --origname <orig>	The enterprise name of the originator of the comment.
-P, --priority <code>	The priority code of the new comment.
-p, --prompt	Prompt the user for values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-r, --response <type>	The response type of the new comment.
-t, --title <title>	The title of the new comment.
-v, --verbose	Print the file name of the newly created comment.
-z, --issue-type <type>	The issue type of the new comment.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newcom -# EX-12345-2018-00001-Q
```

s1kd-newddn

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newddn** tool creates a new S1000D data dispatch note with the code, metadata, and list of files specified.

2 Usage

```
s1kd-newddn [options] <files>...
```

3 Options

-#, --code <code>	The code of the new data dispatch note, in the form of MODELIDENTCODE-SENDER-RECEIVER-YEAR-SEQUENCE.
-$, --issue <issue>	Specifiy which issue of S1000D to use. Currently supported issues are:

	– 5.0 (default) – 4.2 – 4.1 – 4.0 – 3.0 – 2.3 – 2.2 – 2.1 – 2.0
-@, --out <path>	Save the new DDN to <path>. If <path> is an existing directory, the DDN will be created in it instead of the current directory. Otherwise, the DDN will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in the specified directory instead of the built-in template. The template must be named `ddn.xml` inside <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-a, --authorization <auth>	Specify the authorization.
-b, --brex <BREX>	BREX data module code.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	The issue date of the new DDN in the form of YYYY-MM-DD.
-m, --remarks <remarks>	Set the remarks for the new data dispatch note.
-N, --receiver-country <country>	The receiver's country.
-n, --sender-country <country>	The sender's country.
-o, --sender <name>	The enterprise name of the sender.
-p, --prompt	Prompt the user for values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-r, --receiver <name>	The enterprise name of the receiver.
-T, --receiver-city <city>	The receiver's city.
-t, --sender-city <city>	The sender's city.
-v, --verbose	Print the file name of the newly created DDN.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newddn -# EX-12345-54321-2018-00001
```

s1kd-newdm

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-30-00-00-00A-040A-D	s1kd-defaults - Description

Description

1 General

The **s1kd-newdm** tool creates a new S1000D data module with the data module code and other metadata specified.

2 Usage

```
s1kd-newdm [options]
```

3 Options

-#, --code <DMC> — The data module code of the new data module. The prefix "DMC-" is optional.

	If - is given for the code, a random data module code will be generated. If only a model identification code is given instead (e.g., `-# TEST`), or the `.defaults` file specifies a default model identification code, this will be used as part of the random code. The information type of the random code will be 000A-D.
-$, --issue <issue>	Specify which issue of S1000D to use. Currently supported issues are: – 5.0 (default) – 4.2 – 4.1 – 4.0 – 3.0 – 2.3 – 2.2 – 2.1 – 2.0
-@, --out <path>	Save the new data module to <path>. If <path> is an existing directory, the data module will be created in it instead of the current directory. Otherwise, the data module will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use XML templates in the specified directory instead of the built-in templates.
-~, --dump-templates <dir>	Dump the built-in XML templates to the specified directory.
-,, --dump-dmtypes-xml	Dumps the built-in default `.dmtypes` XML. This can be used to quickly set up a starting point for a project's custom info codes, from which info names can be modified and unused codes can be removed to fit the project.
-., --dump-dmtypes	Dumps the simple text form of the built-in default `.dmtypes`.
-!, --no-infoname	Do not include an info name for the new data module.
-a, --act <ACT>	ACT data module code.
-B, --generate-brex-rules	When creating a new BREX data module, use the `.defaults` and `.dmtypes` files to add a basic set of context rules.
-b, --brex <BREX>	BREX data module code.
-C, --country <country>	The country ISO code of the new data module.
-c, --security <sec>	The security classification of the new data module.
-D, --dmtypes <dmtypes>	Specify the `.dmtypes` file name.

-d, --defaults <defaults>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	Issue date of the new data module in the form of YYYY-MM-DD.
-i, --infoname <info>	The info name of the new data module.
-j, --brexmap <map>	Use a custom `.brexmap` file when using the -B option.
-k, --skill <skill>	The skill level code of the new data module.
-L, --language <language>	The language ISO code of the new data module.
-M, --maintained-sns <SNS>	Determine the tech name from on one of the built-in S1000D maintained SNS. Supported SNS: – Generic – Support and training equipment – Ordnance – General communications – Air vehicle, engines and equipment – Tactical missiles – General surface vehicles – General sea vehicles When creating a BREX data module, this SNS will be included as the SNS rules of the new data module. The "`maintainedSns`" `.defaults` file key can be used to set one of the above SNS as the default.
-m, --remarks <remarks>	Set remarks for the new data module.
-N, --omit-issue	Omit issue/inwork numbers from filename.
-n, --issno <issue>	The issue number of the new data module.
-O, --origcode <CAGE>	The CAGE code of the originator.
-o, --origname <orig>	The originator enterprise name of the new data module.
-P, --two-sns-levels	When determining tech name from an SNS (-S or -M), include the previous level of SNS in the tech name. This means that: – tech names derived from a subsystem will be formatted as "System - Subsystem" – tech names derived from a subsubsystem will be formatted as "Subsystem - Subsubsystem" – and tech names derived from an assembly will be formatted as "Subsubsystem - Assembly".

	If both levels have the same title, then only one will be used. The "`includePrevSnsTitle`" `.defaults` file key can also be set to control this option.
-p, --prompt	Prompts the user for any values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-R, --rpccode <CAGE>	The CAGE code of the responsible partner company.
-r, --rpcname <RPC>	The responsible partner company enterprise name of the new data module.
-S, --sns <BREX>	Determine the tech name from the SNS rules of a specified BREX data module. This can also be specified in the `.defaults` file with the key "`sns`".
-s, --schema <schema>	The schema URL.
-T, --type <schema>	The type (schema) of the new data module. Supported schemas: – appliccrossreftable - Applicability cross-reference table – brdoc - Business rule document – brex - Business rule exchange – checklist - Maintenance checklist – comrep - Common information repository – condcrossreftable - Conditions cross-reference table – container - Container – crew - Crew/Operator information – descript - Descriptive – fault - Fault information – frontmatter - Front matter – ipd - Illustrated parts data – learning - Technical training information – prdcrossreftable - Product cross-reference table – proced - Procedural – process - Process – sb - Service bulletin – schedul - Maintenance planning information – scocontent - SCO content information – techrep - Technical repository (replaced by comrep in issue 4.1) – wrngdata - Wiring data

	– wrngflds - Wiring fields
-t, --techname <tech>	The tech name of the new data module.
-V, --infoname-variant <variant>	The info name variant of the new data module.
-v, --verbose	Print the file name of the newly created data module.
-w, --inwork <inwork>	The inwork number of the new data module.
-z, --issue-type <type>	The issue type of the new data module.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 Prompt (-p) option

If this option is specified, the program will prompt the user to enter values for metadata which was not specified when calling the program. If a piece of metadata has a default value (from the `.defaults` and `.dmtypes` files), it will be displayed in square brackets [] in the prompt, and pressing Enter without typing any value will select this default value.

3.2 `.defaults` file

This file sets default values for each piece of metadata. By default, the program will search the current directory and parent directories for a file named `.defaults`, but any file can be specified by using the -d option.

All of the s1kd-new* commands use the same `.defaults` file format, so this file can contain default values for multiple types of metadata.

Each line consists of the identifier of a piece of metadata and its default value, separated by whitespace. Lines which do not match a piece of metadata are ignored, and may be used as comments. Example:

```
# General
countryIsoCode              CA
languageIsoCode             en
originator                  khzae.net
responsiblePartnerCompany   khzae.net
securityClassification      01
```

Alternatively, the `.defaults` file can be written using an XML format, containing a root element `defaults` with child elements `default` which each have an attribute `ident` and an attribute `value`.

```
<?xml version="1.0"?>
```

```
<defaults>
<!-- General -->
<default ident="countryIsoCode" value="CA"/>
<default ident="languageIsoCode" value="en"/>
<default ident="originator" value="khzae.net"/>
<default ident="responsiblePartnerCompany" value="khzae.net"/>
<default ident="securityClassification" value="01"/>
</defaults>
```

3.3 `.dmtypes` file

This file sets the default schema and info name for data modules based on their info code. By default, the program will search the current directory and parent directories for a file named `.dmtypes`, but any file can be specified by using the -D option.

Each line consists of an info code, a schema identifier, and optionally a default info name. Example:

```
000    descript
022    brex        Business rules
040    descript    Description
520    proced      Remove procedure
```

Like the `.defaults` file, the `.dmtypes` file may also be written in an XML format, where each child has an attribute `infoCode`, an attribute `schema`, and optionally an attribute `infoName`.

```
<?xml version="1.0">
<dmtypes>
<type infoCode="000" schema="descript"/>
<type infoCode="022" schema="brex" infoName="Business rules"/>
<type infoCode="040" schema="descript" infoName="Description"/>
<type infoCode="520" schema="proced" infoName="Remove procedure"/>
</dmtypes>
```

The info code field can also include an info code variant, item location code, learn code, and learn event code, which allows for more specific default schemas and info names.

Example of info code variants:

```
258A  proced  Other procedure to clean
258B  proced  Other procedure to clean, Clean with air
258C  proced  Other procedure to clean, Clean with water
```

Example of item location codes:

```
200A-A  proced  Servicing, while installed
200A-C  proced  Servicing, on the bench
200A-T  proced  Servicing, training
```

Example of learn codes:

```
100A-A-H10A  learning  Operation: Performance analysis
```

```
100A-A-T5CC  learning  Operation: Simulation
100A-A-T80E  learning  Operation: Assessment
```

The XML format additionally supports the use of the attribute `infoNameVariant`, for use with S1000D Issue 5.0 and up, allowing extensions of the info name to be encoded separately:

```
<dmtypes>
<type
infoCode="258A"
schema="proced"
infoName="Other procedure to clean"/>
<type
infoCode="258B"
schema="proced"
infoName="Other procedure to clean"
infoNameVariant="Clean with air"/>
<type
infoCode="258C"
schema="proced"
infoName="Other procedure to clean"
infoNameVariant="Clean with water"/>
</dmtypes>
```

Defaults are chosen in the order they are listed in the `.dmtypes` file. An info code which does not specify a variant, item location code, learn code or learn event code, or uses asterisks in their place, matches all possible variants, item location codes, learn codes and learn event codes.

3.4 `.brexmap` file

Refer to S1KDTOOLS-A-30-00-00-00A-040A-D for a description of the `.brexmap` file.

3.5 Custom XML templates (-%)

A minimal set of S1000D templates are built-in to this tool, but customized templates may be used with the -% option. This option takes a path to a directory where the custom templates are located. Each template should be named `<schema>.xml`, where `<schema>` is the name of the schema, matching one of the schema names in the `.dmtypes` file or the schema specified with the -T option.

The templates must be written to conform to the default S1000D issue of this tool (currently 5.0). They will be automatically transformed when another issue is specified with the -$ option.

The `templates` default can also be specified in the `.defaults` file to use these custom templates by default.

4 Example

```
$ s1kd-newdm -# S1KDTOOLS-A-00-07-00-00A-040A-D
```

s1kd-newdml

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newdml** tool creates a new S1000D data management list with the code and other metadata specified.

2 Usage

```
s1kd-newdml [options] [<object>...]
```

3 Options

-#, --code <code>
: The data management list code of the new DML.

-$, --issue <issue>
: Specify which issue of S1000D to use. Currently supported issues are:

  – 5.0 (default)

	– 4.2 – 4.1 – 4.0 – 3.0 – 2.3 – 2.2 – 2.1 – 2.0
-@, --out <path>	Save the new DML to <path>. If <path> is an existing directory, the DML will be created in it instead of the current directory. Otherwise, the DML will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in the specified directory instead of the built-in template. The template must be named `dml.xml` inside <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-b, --brex <BREX>	BREX data module code.
-c, --security <sec>	The security classification of the new DML.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show usage message.
-I, --date <date>	The issue date of the new DML in the form of YYYY-MM-DD.
-i, --info-code <info code>	When creating a DMRL from SNS rules (-S), use the specified info code for each entry. Specify this option multiple times to create multiple data modules for each part of the SNS. <info code> can specify: – the base info code (e.g., 520) – the info code variant (e.g., 520B) – the item location code (e.g., 520B-C)
-l, --list	Treat input (stdin or arguments) as lists of CSDB objects to add to the new list.
-m, --remarks <remarks>	Set the remarks for the new data management list.
-N, --omit-issue	Omit the issue/inwork numbers from filename.
-n, --issno <issue>	The issue number of the new DML.
-p, --prompt	Prompts the user for any values left unspecified.
-q, --quiet	Do not report an error when the file already exists.

-R, --rpccode <NCAGE>	Specifies a default responsible partner company enterprise code for entries which do not carry this in their ID STATUS section (ICN, COM, DML).
-r, --rpcname <name>	Specifies a default responsible partner company enterprise name for entries which do not carry this in their IDSTATUS section (ICN, COM, DML).
-S, --sns <SNS>	Create a DMRL using the specified SNS rules.
-v, --verbose	Print the file name of the newly created DML.
-w, --inwork <inwork>	The inwork number of the new DML.
-z, --issue-type <type>	The issue type of the new DML.
--version	Show version information.
<object>...	Any number of CSDB object file names to automatically add to the list.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newdml -# EX-12345-C-2018-00001
```

s1kd-newimf

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newimf** tool creates a new S1000D ICN metadata file for specified ICN files.

2 Usage

```
s1kd-newimf [options] <ICNs>...
```

3 Options

-$, --issue <issue>

Specify which issue of S1000D to use. Currently supported issues are:

– 5.0 (default)

– 4.2

-@, --out <path>	Save the new IMF to <path>. If <path> is an existing directory, the IMF will be created in it instead of the current directory. Otherwise, the IMF will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in <dir> instead of the built-in template. The template must be named `icnmetadata.xml` inside <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-b, --brex <BREX>	BREX data module code.
-c, --security <sec>	The security classification of the new ICN metadata file.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	The issue date of the new ICN metadata file in the form of YYYY-MM-DD.
-m, --remarks <remarks>	Set the remarks for the new ICN metadata file.
-n, --issno <issue>	The issue number of the new ICN metadata file.
-O, --origcode <CAGE>	The CAGE code of the originator.
-o, --orignamе <orig>	The originator enterprise name of the new ICN metadata file.
-p, --prompt	Prompts the user for any values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-R, --rpccode <CAGE>	The CAGE code of the responsible partner company.
-r, --rpcname <RPC>	The responsible partner company enterprise name of the new ICN metadata file.
-t, --title <title>	The ICN title (if creating multiple ICNs, they will all use this title).
-v, --verbose	Print the file name of the newly created IMF.
-w, --inwork <inwork>	The inwork issue of the new ICN metadata file.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newimf ICN-EX-00001-001-01.PNG
```

s1kd-newpm

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newpm** tool creates a new S1000D publication module with the publication module code and other metadata specified.

2 Usage

```
s1kd-newpm [options] [<DM>...]
```

3 Options

-#, --code <PMC>
: The publication module code of the new publication module.

-$, --issue <issue>
: Specify which issue of S1000D to use. Currently supported issues are:

  – 5.0 (default)

	– 4.2 – 4.1 – 4.0 – 3.0 – 2.3 – 2.2 – 2.1 – 2.0
-@, --out <path>	Save the new publication module to <path>. If <path> is an existing directory, the publication module will be created in it instead of the current directory. Otherwise, the publication module will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in <dir> instead of the built-in template. The template must be named `pm.xml` in <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-a, --act <ACT>	ACT data module code.
-b, --brex <BREX>	BREX data module code.
-C, --country <country>	The country ISO code of the new publication module.
-c, --security <sec>	The security classification of the new publication module.
-D, --include-date	Include issue date in referenced data modules.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	The issue date of the new publication module in the form of YYYY-MM-DD.
-i, --include-issue	Include issue information in referenced data modules.
-L, --language <language>	The language ISO code of the new publication module.
-l, --include-lang	Include language information in referenced data modules.
-m, --remarks <remarks>	Set remarks for the new publication module.
-n, --issno <issue>	The issue number of the new publication module.
-p, --prompt	Prompt the user for any values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-R, --rpccode <CAGE>	The CAGE code of the responsible partner company.
-r, --rpcname <RPC>	The responsible partner company enterprise name of the new publication module.

-s, --short-title <title>	The short title of the new publication module.
-T, --include-title	Include titles in referenced data modules.
-t, --title <title>	The title of the new publication module.
-v, --verbose	Print the file name of the newly created publication module.
-w, --inwork <inwork>	The inwork number of the new publication module.
-z, --issue-type <type>	The issue type of the new publication module.
--version	Show version information.
<DM>...	Any number of data modules to automatically reference in the new publication module's content.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newpm -# EX-12345-00001-00
```

s1kd-newsmc

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newsmc** tool creates a new S1000D SCORM content package with the SCORM content package code and other metadata specified.

2 Usage

```
s1kd-newsmc [options] [<DM>...]
```

3 Options

-#, --code <SMC>	The SCORM content package code of the new SCORM content package.
-$, --issue <issue>	Specify which issue of S1000D to use. Currently supported issues are:

	– 5.0 (default) – 4.2 – 4.1
-@, --out <path>	Save the new SCORM content package to <path>. If <path> is an existing directory, the SCORM content package will be created in it instead of the current directory. Otherwise, the SCORM content package will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use the XML template in <dir> instead of the built-in template. The template must be named `scormcontentpackage.xml` in <dir> and must conform to the default S1000D issue (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-a, --act <ACT>	ACT data module code.
-b, --brex <BREX>	BREX data module code.
-C, --country <country>	The country ISO code of the new SCORM content package.
-c, --security <sec>	The security classification of the new SCORM content package.
-D, --include-date	Include issue date in referenced data modules.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-I, --date <date>	The issue date of the new SCORM content package in the form of YYYY-MM-DD.
-i, --include-issue	Include issue information in referenced data modules.
-k, --skill <skill>	The skill level code of the new SCORM content package.
-L, --language <language>	The language ISO code of the new SCORM content package.
-l, --include-lang	Include language information in referenced data modules.
-m, --remarks <remarks>	Set remarks for the new SCORM content package.
-n, --issno <issue>	The issue number of the new SCORM content package.
-p, --prompt	Prompt the user for any values left unspecified.
-q, --quiet	Do not report an error when the file already exists.
-R, --rpccode <CAGE>	The CAGE code of the responsible partner company.
-r, --rpcname <RPC>	The responsible partner company enterprise name of the new SCORM content package.
-T, --include-title	Include titles in referenced data modules.

-t, --title <title>	The title of the new SCORM content package.
-v, --verbose	Print the file name of the newly created SCORM content package.
-w, --inwork <inwork>	The inwork number of the new SCORM content package.
-z, --issue-type <type>	The issue type of the new SCORM content package.
--version	Show version information.
<DM>...	Any number of data modules to automatically reference in the new SCORM content package's content.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newsmc -# EX-12345-00001-00
```

s1kd-newupf

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1KDTOOLS-A-07-00-00-00A-040A-D	s1kd-newdm - Description

Description

1 General

The **s1kd-newupf** tool creates a new S1000D data update file for two specified issues of a CIR data module. Changes to items between the source and target issues of the CIR are recorded in the resulting UPF, along with update instructions.

2 Usage

```
s1kd-newupf [options] <SOURCE> <TARGET>
```

3 Options

-$, --issue <issue>

Specify which issue of S1000D to use. Currently supported issues are:

– 5.0 (default)

– 4.2
– 4.1

-@, --out <path>	Save the new update file to <path>. If <path> is an existing directory, the update file will be created in it instead of the current directory. Otherwise, the update file will be saved as the filename <path> instead of being automatically named.
-%, --templates <dir>	Use XML template in the specified directory instead of the built-in template. The template must be named `update.xml` in the directory <dir>, and must conform to the default S1000D issue of this tool (5.0).
-~, --dump-templates <dir>	Dump the built-in XML template to the specified directory.
-d, --defaults <file>	Specify the `.defaults` file name.
-f, --overwrite	Overwrite existing file.
-h, -?, --help	Show help/usage message.
-q, --quiet	Do not report an error when the file already exists.
-v, --verbose	Print the file name of the newly created data update file.
--version	Show version information.
<SOURCE>	The source (original) issue of the CIR data module.
<TARGET>	The target (updated) issue of the CIR data module.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.defaults` file

Refer to S1KDTOOLS-A-07-00-00-00A-040A-D for information on the `.defaults` file which is used by all the s1kd-new* commands.

4 Example

```
$ s1kd-newupf \
    DMC-EX-A-00-00-00-00A-00GA-D_001-00_EN-CA.XML \
    DMC-EX-A-00-00-00-00A-00GA-D_002-00_EN-CA.XML
```

s1kd-addicn

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-addicn** tool adds the required DTD entity and notation declarations to an S1000D module in order to reference an ICN file.

2 Usage

```
s1kd-addicn [-o <file>] [-s <src>] [-fh?] <ICN>...
```

3 Options

-F, --full-path	Use the whole path given for the ICN file as the SYSTEM ID.
-f, --overwrite	Overwrite source file instead of writing to stdout.
-h, -?, --help	Show help/usage message.
-o, --out <out>	The filename to output to. Default is to write to stdout.

-s, --source <src>	The source module to add the ICN(s) to. Default is to read from stdin.
--version	Show version information.
<ICN>..	Any number of ICN files to add.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ s1kd-addicn -fs <DM> ICN-EX-12345-001-01.JPG
```

s1kd-ls

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-ls** tool searches the current directory or specified directory trees and lists the file names of CSDB objects matching certain criteria.

The files representing the CSDB objects must use either the standard S1000D file naming conventions, or the alternate naming convention supported by these tools using the -N option.

2 Usage

```
s1kd-ls [-0CDGIiLlMNnoPRrSUwX7] [-e <cmd>]
        [<object>|<dir> ...]
```

3 Options

-0, --null — Output a null-delimited list of CSDB object paths.

-C, -D, -G, -L, -M, -P, -S, -U, -X	List comments, data modules, ICNs, data management lists, ICN metadata files, publication modules, SCORM content packages, data update files, and data dispatch notes respectively. If none are specified, -CDGLMPSUX is assumed. The following long options can also be used for each: --com, --dm, --icn, --dml, --imf, --pm, --smc, --upf, --ddn.
-e, --exec <cmd>	Execute a command for each CSDB object instead of listing them. The string "{}" is replaced by the current CSDB object file name everywhere it occurs in the arguments to the command.
-h, -?, --help	Show the usage message.
-I, --inwork	Show only inwork issues of objects (inwork != 00).
-i, --official	Show only official issues of objects (inwork = 00).
-l, --latest	Show only the latest official/inwork issue of objects.
-N, --omit-issue	Assume that the files being listed do not include the issue info in their filenames, i.e. they were created using the -N option of the s1kd-new* tools.
-n, --other	List non-S1000D files.
-o, --old	Show only old official/inwork issues of objects.
-R, --read-only	Show only non-writable object files.
-r, --recursive	Recursively descend in to directories.
-w, --writable	Show only writable object files.
-7, --list	Treat input as a list of CSDB objects to process.
--version	Show version information.
<object>|<dir> ...	An optional list of CSDB objects to list or directories to search for CSDB objects in. If none are specified, CSDB objects in the current directory are listed by default.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ s1kd-ls
DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
```

```
DMC-EX-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
DMC-EX-B-00-00-00-00A-040A-D_000-01_EN-CA.XML
ICN-12345-00001-001-01.JPG
ICN-12345-00001-002-01.JPG
PMC-EX-12345-00001-00_000-01_EN-CA.XML

$ s1kd-ls -l
DMC-EX-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
DMC-EX-B-00-00-00-00A-040A-D_000-01_EN-CA.XML
ICN-12345-00001-002-01.JPG
PMC-EX-12345-00001-00_000-01_EN-CA.XML

$ s1kd-ls -o
DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
ICN-12345-00001-001-01.JPG

$ s1kd-ls -D | s1kd-metadata -lt -ntechName -ninfoName -nissueDate
Example A    Description    2018-03-20
Example A    Description    2018-03-29
Example B    Description    2018-03-29

$ s1kd-ls -Dl -e 'stat --printf="%n %Y\n" {}'
DMC-EX-A-00-00-00-00A-040A-D_000-02_EN-CA.XML 1553738720
DMC-EX-B-00-00-00-00A-040A-D_000-01_EN-CA.XML 1553738751
```

s1kd-ref

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-ref** tool generates the XML for S1000D reference elements using the specified code or filename. When using a filename, it can parse the CSDB object to include the issue, language, and/or title information in the reference.

2 Usage

```
s1kd-ref [-cdfgiLlqRrStuvh?] [-$ <issue>] [-s <src>] [-T <opts>]
         [-x <xpath>] [-3 <file>] [-o <dst>] [<code>|<file> ...]
```

3 Options

-$, --issue <issue>	Output XML for the specified issue of S1000D.
-c, --content	When using the -T option, only transform textual references found in the content section of CSDB objects.

-d, --include-date	Include the issue date in the reference (target must be a file)
-f, --overwrite	Overwrite source data module instead of writing to stdout.
-g, --guess-prefix	Accept references which do not include a standard prefix (e.g., "DMC-", "PMC-") and guess what they are based on their format and, when using the -T option, the XML context in which they occur.
-h, -?, --help	Show the usage message.
-i, --include-issue	Include the issue information in the reference (target must be a file)
-L, --list	Treat input as a list of CSDB objects.
-l, --include-lang	Include the language information in the reference (target must be a file)
-o, --out <dst>	Output to <dst> instead of stdout.
-q, --quiet	Quiet mode. Do not print errors.
-R, --repository-id	Generate a `<repositorySourceDmIdent>` for a data module.
-r, --add	Add the generated reference to the source data module's `refs` table and output the modified data module to stdout.
-S, --source-id	Generate a `<sourceDmIdent>` (for data modules) or `<sourcePmIdent>` (for publication modules).
-s, --source <src>	Specify a source data module <src> to add references to when using the -r option.
-T, --transform <opts>	Transform textual references into the appropriate XML within text nodes in the XML document(s) specified. The textual references must include the standard prefixes (e.g., "DMC-", "PMC-'), unless the -p option is specified. <opts> is a sequence of characters from "CDEGLPSY", for comment, data module, external publication, ICN, DML, publication module, SCORM content package and CSN references respectively. If "all" is given, then all types of references will be transformed.
-t, --include-title	Include the title in the reference (target must be a file).
-u, --include-url	Include the full URL/filename of the reference with the `xlink:href` attribute.
-v, --verbose	Verbose output.
-x, --xpath <xpath>	When using the -T option, this specifies which nodes to transform textual references in. By default, only the elements which can contain each type of reference are considered.
-3, --externalpubs <file>	Use a custom `.externalpubs` file.
--version	Show version information.

<code>|<file>	Either a code, including the prefix (DMC, PMC, etc.), or the filename of a CSDB object.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.externalpubs` file

The `.externalpubs` file contains definitions of external publication references. This can be used to generate the XML for an external publication reference by specifying the external publication code.

Example of a `.externalpubs` file:

```
<externalPubs>
<externalPubRef>
<externalPubRefIdent>
<externalPubCode>ABC</externalPubCode>
<externalPubTitle>ABC Manual</externalPubTitle>
</externalPubRefIdent>
</externalPubRef>
</externalPubs>
```

4 Examples

Reference to data module with data module code:

```
$ s1kd-ref DMC-EX-A-00-00-00-00A-040A-D
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
itemLocationCode="D"/>
</dmRefIdent>
</dmRef>
```

Reference to data module with data module code and issue/language:

```
$ s1kd-ref -il DMC-EX-A-00-00-00-00A-040A-D_001-03_EN-CA
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
```

```
itemLocationCode="D"/>
<issueInfo issueNumber="001" inWork="03"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
</dmRefIdent>
</dmRef>
```

Reference to data module with all information, from a file:

```
$ s1kd-ref -dilt DMC-EX-A-00-00-00-00A-040A-D_001-03_EN-CA.XML
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
itemLocationCode="D"/>
<issueInfo issueNumber="001" inWork="03"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
</dmRefIdent>
<dmRefAddressItems>
<dmTitle>
<techName>Example</techName>
<infoName>Description</infoName>
</dmTitle>
<issueDate year="2018" month="06" day="25"/>
</dmRefAddressItems>
</dmRef>
```

Reference to a catalog sequence number:

```
$ s1kd-ref CSN-EX-A-00-00-00-01A-004A-D
<catalogSeqNumberRef modelIdentCode="EX" systemDiffCode="A"
systemCode="00" subSystemCode="0" subSubSystemCode="0" assyCode="00"
figureNumber="01" figureNumberVariant="A" item="004" itemVariant="A"
itemLocationCode="D"/>
```

Reference to a comment:

```
$ s1kd-ref COM-EX-12345-2018-00001-Q
<commentRef>
<commentRefIdent>
<commentCode modelIdentCode="EX" senderIdent="12345"
yearOfDataIssue="2018" seqNumber="00001" commentType="q"/>
</commentRefIdent>
</commentRef>
```

Reference to a data management list:

```
$ s1kd-ref DML-EX-12345-C-2018-00001
<dmlRef>
<dmlRefIdent>
```

```
<dmlCode modelIdentCode="EX" senderIdent="12345" dmlType="c"
yearOfDataIssue="2018" seqNumber="00001"/>
</dmlRefIdent>
</dmlRef>
```

Reference to an information control number:

```
$ s1kd-ref ICN-EX-A-000000-A-00001-A-001-01
<infoEntityRef infoEntityRefIdent="ICN-EX-A-000000-A-00001-A-001-01"/>
```

Reference to a publication module:

```
$ s1kd-ref PMC-EX-12345-00001-00
<pmRef>
<pmRefIdent>
<pmCode modelIdentCode="EX" pmIssuer="12345" pmNumber="00001"
pmVolume="00"/>
</pmRefIdent>
</pmRef>
```

Reference to a SCORM content package:

```
$ s1kd-ref SMC-EX-12345-00001-00
<scormContentPackageRef>
<scormContentPackageRefIdent>
<scormContentPackageCode
modelIdentCode="EX"
scormContentPackageIssuer="12345"
scormContentPackageNumber="00001"
scormContentPackageVolume="00"/>
</scormContentPackageRefIdent>
</scormContentPackageRef>
```

Source identification for a data module:

```
$ s1kd-ref -S DMC-EX-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
<sourceDmIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
itemLocationCode="D"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
<issueInfo issueNumber="001" inWork="00"/>
</sourceDmIdent>
```

Source identification for a publication module:

```
$ s1kd-ref -S PMC-EX-12345-00001-00_001-00_EN-CA.XML
<sourcePmIdent>
<pmCode modelIdentCode="EX" pmIssuer="12345" pmNumber="00001"
```

```
pmVolume="00"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
<issueInfo issueNumber="001" inWork="00"/>
</sourcePmIdent>
```

Source identification for a SCORM content package:

```
$ s1kd-ref -S SMC-EX-12345-00001-00_001-00_EN-CA.XML
<sourceScormContentPackageIdent>
<scormContentPackageCode
modelIdentCode="EX"
scormContentPackageIssuer="12345"
scormContentPackageNumber="00001"
scormContentPackageVolume="00"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
<issueInfo issueNumber="000" inWork="01"/>
</sourceScormContentPackageIdent>
```

Repository source identification for a CIR data module:

```
$ s1kd-ref -R DMC-EX-A-00-00-00-00A-00GA-D_001-00_EN-CA.XML
<repositorySourceDmIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="00G" infoCodeVariant="A"
itemLocationCode="D"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
<issueInfo issueNumber="001" inWork="00"/>
</repositorySourceDmIdent>
```

Reference to an external publication:

```
$ s1kd-ref ABC
<externalPubRef>
<externalPubRefIdent>
<externalPubCode>ABC</externalPubCode>
</externalPubRefIdent>
</externalPubRef>
```

Reference to an external publication (from the `.externalpubs` file):

```
$ s1kd-ref ABC
<externalPubRef>
<externalPubRefIdent>
<externalPubCode>ABC</externalPubCode>
<externalPubTitle>ABC Manual</externalPubTitle>
</externalPubRefIdent>
</externalPubRef>
```

s1kd-metadata

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-metadata** tool provides a simple way to fetch and change metadata on S1000D CSDB objects.

2 Usage

```
s1kd-metadata [options] [<object>...]
```

3 Options

-0, --null	Print a null-delimited list of values of the pieces of metadata specified with -n, or all available metadata if -n is not specified.
-c, --set <file>	Use <file> to edit metadata files. <file> consists of lines starting with a metadata name, followed by whitespace, followed by the new value for the metadata (the program

	uses this same format when outputting all metadata if no <name> is specified).
-d, --date-format <fmt>	The format to use when printing dates, such as the "issueDate" or "modified" metadata. <fmt> should conform to the format used by strftime. The default is "%Y-%m-%d".
-E, --editable	When showing all metadata, only list editable items. This is useful when creating a file for use with the -c option.
-e, --exec <cmd>	Execute a command for each CSDB object. The string "{}" is replaced by the current CSDB object file name everywhere it occurs in the arguments to the command.
-F, --format <fmt>	Print a formatted line for each CSDB object. Metadata names surrounded with % (e.g. %issueDate%) will be substituted by the value read from the object.
-f, --overwrite	When editing metadata, overwrite the object. The default is to output the modified object to stdout.
-H, --info	Lists all available metadata with a short description of each. Specify specific metadata to describe with the -n option.
-h, -?, --help	Show help/usage message.
-l, --list	Treat input as a list of object filenames to read or edit metadata on, rather than an object itself.
-m, --matches <regex>	Used after a -w or -W option, this specifies a regular expression to match the value of the given metadata against, instead of a literal value (-v).
-n, --name <name>	The name of the piece of metadata to fetch. This option can be specified multiple times to fetch multiple pieces of metadata. If -n is not specified, all available metadata names are printed with their values. This output can be sent to a text file, edited, and then specified with the -c option as a means of editing metadata in any text editor.
-q, --quiet	Quiet mode. Non-fatal errors such as a missing piece of optional metadata in an object will not be printed to stderr.
-T, --raw	Do not format columns in output.
-t, --tab	Print a tab-delimited list of values of the pieces of metadata specified with -n, or all available metadata if -n is not specified.
-v, --value <value>	When following a -n option, this specifies the new value for that piece of metadata. When following a -w or -W option, this specifies the value to compare that piece of metadata to. Each -n, -w, or -W can be followed by -v to edit or define conditions on multiple pieces of metadata.

-W, --not-when <name>	Show or edit metadata only on objects where the value of <name> is not equal to the value specified in the following -v option. If no -v option follows, this will show objects which do not have metadata <name> of any value.
-w, --when <name>	Show or edit metadata only on objects where the value of <name> is equal to the value specified in the following -v option. If no -v option follows, this will show objects which have metadata <name> with any value.
--version	Show version information.
<object>...	The object(s) to show/edit metadata on. The default is to read from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ ls
DMC-S1KDTOOLS-A-09-00-00-00A-040A-D_EN-CA.XML
DMC-S1KDTOOLS-A-0Q-00-00-00A-040A-D_EN-CA.XML

$ DMOD=DMC-S1KDTOOLS-A-09-00-00-00A-040A-D_EN-CA.XML
$ s1kd-metadata $DMOD
issueDate                       2017-08-14
techName                        s1kd-metadata(1) | s1kd-tools
responsiblePartnerCompany       khzae.net
originator                      khzae.net
securityClassification          01
schema                          descript
schemaUrl                       http://www.s1000d.org/S1000D_5-0/xml_
schema_flat/descript.xsd
type                            dmodule
applic                          All
brex                            S1000D-F-04-10-0301-00A-022A-D
issueType                       new
languageIsoCode                 en
countryIsoCode                  CA
issueNumber                     001
inWork                          00
dmCode                          S1KDTOOLS-A-09-00-00-00A-040A-D

$ s1kd-metadata -n techName -v "New title" $DMOD
```

```
$ s1kd-metadata -n techName $DMOD
New title

$ s1kd-metadata -n techName DMC-*.XML
New title
s1kd-aspp(1) | s1kd-tools

$ s1kd-metadata -F "%techName% (%issueDate%) %issueType%" DMC-*.XML
New title (2017-08-14) new
s1kd-aspp(1) | s1kd-tools (2018-03-28) changed

$ s1kd-metadata -F "%techName%" -w subSubSystemCode -v Q DMC-*.XML
s1kd-aspp(1) | s1kd-tools

$ s1kd-metadata -n path -w subSystemCode -v Q
DMC-S1KDTOOLS-A-0Q-00-00-00A-040A-D_EN-CA.XML

$ s1kd-metadata -n path -W subSystemCode -v Q
DMC-S1KDTOOLS-A-09-00-00-00A-040A-D_EN-CA.XML

$ s1kd-metadata -n path -w subSystemCode -m [0-9]
DMC-S1KDTOOLS-A-09-00-00-00A-040A-D_EN-CA.XML
```

s1kd-mvref

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-mvref** tool changes all references to one object (the source object) into references to another object (the target object) in a specified set of objects.

2 Usage

```
s1kd-mvref [-d <dir>] [-s <source>] [-t <target>] [-cflvh?]
           [<object>...]
```

3 Options

-c, --content	Only move references within the content section of objects.
-d, --dir <dir>	Move references in all objects in the specified directory.
-f, --overwrite	Overwrite updated input objects.
-h, -?, --help	Show help/usage message

-l, --list	Treat input as a list of data module filenames, rather than a data module itself.
-s, --source <source>	The source object.
-t, --target <target>	Change all references to the source object specified with -s into references that point to <target>.
-v, --verbose	Verbose output.
--version	Show version information.
<object>...	Objects to move references in.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ s1kd-mvref -f -s <old DM> -t <new DM> DMC-*.XML
```

s1kd-sns

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-sns** tool can be used to automatically organize data modules in a CSDB in to a directory hierarchy based on a specified SNS structure. It may also be used to simply print an indented text version of an SNS structure.

2 Usage

```
s1kd-sns [-D <dir>] [-d <dir>] [-cmnpsh?] [<BREX> ...]
```

3 Options

-c, --copy	Copy files in to the SNS subfolders instead of linking them.
-D, --srcdir <dir>	The flat directory containing the data modules to organize. By default, the current directory is used.
-d, --outdir <dir>	The root directory of the new SNS structure. By default, the tool will use the name "SNS" in the current directory.

-h, -?, --help	Show usage message.
-m, --move	Move files in to the SNS subfolders instead of linking them.
-n, --only-code	Use only the SNS codes when naming directories. By default, each directory will be named in the form of "snsCode - snsTitle".
-p, --print	Print the SNS structure only.
-s, --symlink	Use symbolic links to organize the SNS instead of the default hard links.
--version	Show version information.
<BREX>	Read the SNS structure from the specified BREX data module. If none is specified, the tool will read from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ s1kd-sns DMC-S1000D-A-08-02-0100-00A-022A-D_EN-US.XML
$ tree SNS
SNS
|_ 00 - Product, General
   |_ 0 - Product, General
   |_ 1 - Product, General maintenance
   |_ 2 - Product, Safety
   |
...
|_ 04 - Worthiness (fit for purpose) limitations
   |_ 0 - General
   |_ 1 - Fatigue index calculations
   |_ 2 - Operating spectrums
|_ 05 - Scheduled/unscheduled maintenance
   |_ 0 - General
   |_ 1 - Time limits
   |_ 2 - Scheduled maintenance check lists
...
|_ 18 - Vibration and noise analysis and attenuation
```

s1kd-transform

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

Applies an XSLT stylesheet to S1000D CSDB objects. The DTD of any specified objects is preserved in the resulting output, which leaves external entities such as ICN references intact.

2 Usage

```
s1kd-transform [-s <stylesheet> [-p <name>=<value> ...] ...]
               [-o <file>] [-filvh?] [<object> ...]
```

3 Options

-f, --overwrite	Overwrite the specified CSDB object(s) instead of writing to stdout.
-h, -?, --help	Show usage message.

-i, --identity	Includes an "identity" template in to each specified stylesheet.
-l, --list	Treat input (stdin or arguments) as lists of CSDB objects to transform, rather than CSDB objects themselves.
-o, --out <file>	Output to <file> instead of stdout. This option only makes sense when the input is a single CSDB object.
-p, --param <name>=<value>	Pass a parameter to the last specified stylesheet.
-s, --stylesheet <stylesheet>	An XSLT stylesheet file to apply to each CSDB object. Multiple stylesheets can be specified by supplying this argument multiple times. The stylesheets will be applied in the order they are listed.
-v, --verbose	Verbose output.
--version	Show version information.
<object> ...	Any number of CSDB objects to apply all specified stylesheets to.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 Identity template

The -i option includes an "identity" template in to each stylesheet specified with the -s option. The template is equivalent to this XSL:

```
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()"/>
</xsl:copy>
</xsl:template>
```

This means that any attributes or nodes which are not matched by a more specific template in the user-specified stylesheet are copied.

4 Example

```
$ s1kd-transform -s <XSL> <DM1> <DM2> ...
```

s1kd-upissue

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-upissue** tool increases the in-work or issue number of an S1000D CSDB object.

Any files using an S1000D-esque naming convention, placing the issue and in-work numbers after the first underscore (_) character, can also be "upissued". Files which do not contain the appropriate S1000D metadata are simply copied.

2 Usage

```
s1kd-upissue [-4DdefHIilmNqRruvw] [-1 <type>] [-2 <type>]
             [-c <reason>] [-s <status>] [-t <urt>] [-z <date>]
             [<file>...]
```

3 Options

-1, --first-ver <type>	Set first verification type (tabtop, onobject, ttandoo).
-2, --second-ver <type>	Set second verification type (tabtop, onobject, ttandoo).
-4, --remove-marks	Remove change markup on elements, but not RFUs, in the upissued object. This is automatically applied if the issue type (-s) is not `"changed"` or `"rinstate-changed"`.
-c, --reason <reason>	Add a reason for update to the upissued objects. Multiple RFUs can be added by specifying this option multiple times.
-D, --remove-deleted	Remove elements with change type of "delete". These elements are automatically removed along with all change marks and RFUs when an object is upissued from official to the next inwork issue. This option will remove them when upissuing between inwork issues, or when making the object official.
-d, --dry-run	Do not actually create or modify any files, only print the name of the file that would be created or modified.
-e, --erase	Remove old issue file after upissuing.
-f, --overwrite	Overwrite existing upissued CSDB objects.
-H, --highlight	Mark the last specified reason for update (-c) as a highlight.
-h, -?, --help	Show help/usage message.
-I, --(keep|change)-date	Do not change issue date. Normally, when upissuing to the next inwork or official issue, the issue date is changed to the current date, or the date specified with the -z option. This option will keep the date of the previous inwork or official issue. In -m mode, this option has the opposite effect, causing the date to be changed. The two alternative long option names, --keep-date and --change-date, allow for the intended meaning of this option to be expressed clearly in each mode.
-i, --official	Increase the issue number of the CSDB object. By default, the in-work issue is increased.
-l, --list	Treat input (stdin or arguments) as lists of CSDB objects to upissue, rather than CSDB objects themselves.
-m, --modify	Modify issue-related metadata on objects without incrementing the issue or inwork numbers. The -I, -q, and -r options have the opposite effect in this mode. The modified objects are written to stdout by default, and the -f option can be used to change them in-place.
-N, --omit-issue	Omit issue/inwork numbers from filename.
-q, --(keep|reset)-qa	Keep quality assurance information from old issue. Normally, when upissuing an official CSDB object to the first in-work

	issue, the quality assurance is set back to "unverified". Specify this option to indicate the upissue will not affect the contents of the CSDB object, and so does not require it to be re-verified. In -m mode, this option has the opposite effect, causing the QA status to be reset. The two alternative long option names, --keep-qa and --reset-qa, allow for the intended meaning of this option to be expressed clearly in each mode.
-R, --keep-unassoc-marks	Delete only change markup on elements associated with an RFU (by use of the attribute `reasonForUpdateRefIds`). Change markup on other elements is ignored.
-r, --(keep|remove)-changes	Keep old RFUs and change marks. Normally, when upissuing an offical CSDB object to the first in-work issue, any reasons for update are deleted automatically, along with any change markup attributes on elements (when change type is "add" or "modify") or the elements themselves (when change type is "delete"). This option prevents their deletion. In -m mode, this option has the opposite effect, causing the current RFUs and change marks to be removed. The two alternative long option names, --keep-changes and --remove-changes, allow for the intended meaning of this option to be expressed clearly in each mode.
-s, --status <status>	Set the status of the new issue. Default is 'changed'.
-t, --type <urt>	Set the updateReasonType of the last specified reason for update (-c).
-u, --clean-rfus	Remove RFUs which are not associated with any change markup (by use of the attribute `reasonForUpdateRefIds`).
-v, --verbose	Print the file name of the upissued CSDB object.
-w, --lock	Make the old issue file read-only after upissuing. Official issues (-i) will also be made read-only when they are created.
-z, --date <date>	Specify the issue to use for the upissued object(s). Otherwise, the current date will be used.
--version	Show version information.
<file>...	Any number of CSDB objects or other files to upissue. If none are specified, the object will be read from stdin and the upissued object will be written to stdout.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.

--xinclude	Do XInclude processing.

4 Examples

4.1 Data module with issue/inwork in filename

```
$ ls
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML

$ s1kd-upissue DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
$ ls
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML

$ s1kd-upissue \
  -i DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
$ ls
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
```

4.2 Data module without issue/inwork in filename

```
$ ls
DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_EN-US.XML

$ s1kd-metadata DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_EN-CA.XML \
  -n issueInfo
000-01
$ s1kd-upissue -N DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_EN-CA.XML
$ s1kd-metadata DMC-S1KDTOOLS-A-00-00-00-00A-040A-D_EN-CA.XML \
  -n issueInfo
000-02
```

4.3 Non-XML file with issue/inwork in filename

```
$ ls
TXT-S1KDTOOLS-KHZAE-FOOBAR_000-01_EN-CA.TXT

$ s1kd-upissue TXT-S1KDTOOLS-KHZAE-00001_000-01_EN-CA.TXT
$ ls
TXT-S1KDTOOLS-KHZAE-FOOBAR_000-01_EN-CA.TXT
TXT-S1KDTOOLS-KHZAE-FOOBAR_000-02_EN-CA.TXT
```

s1kd-appcheck

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-appcheck** tool validates the applicability of S1000D CSDB objects, detecting potential errors that could occur when the object is filtered.

By default, the tool validates an object against only the product attribute and condition values which are explicitly used within the object. The products check (-t) and full check (-a) modes allow objects to be checked for issues with implicit applicability, that is, product attribute or condition values which are not explicitly used within an object, but may still affect it.

The s1kd-instance and s1kd-validate tools are used by default to perform the actual validation.

2 Usage

```
s1kd-appcheck [options] [<object>...]
```

3 Options

-A, --act <file>	Specify the ACT to read product attributes from, and to use to find the CCT or PCT. This will override the ACT reference within the individual objects being validated.
-a, --all	Validate objects against all possible combinations of relevant product attribute and condition values as defined in the ACT and CCT. Relevant product attributes and conditions are those that are used by an object with any value.
-b, --brexcheck	Validate objects with a BREX check (using the s1kd-brexcheck tool) in addition to the schema check.
-C, --cct <file>	Specify the CCT to read conditions from. This will override the CCT reference within the ACT.
-c, --custom	Perform a customized check. The default standalone applicability check is disabled. This can then be combined with the -s option, to only check that all product attributes and conditions are defined in the ACT and CCT respectively, and/or the -n option, to only check nested applicability annotations. If neither of these options are specified, no checks will be performed.
-d, --dir <dir>	The directory to start searching for ACT/CCT/PCT data modules in. By default, the current directory is used.
-e, --exec <cmd>	The commands used to validate objects. Multiple commands can be used by specifying this option multiple times. The objects will be passed to each command on stdin, and the exit status of the command will be used to determine if the object is valid (with a non-zero exit status indicating it is invalid). This overrides the default commands (s1kd-validate, and s1kd-brexcheck if -b is specified).
-f, --filenames	Print the filenames of invalid objects.
-h, -?, --help	Show help/usage message.
-K, --filter <cmd>	The command used to filter objects prior to validation. The objects will be passed to the command on stdin, and the filters will be supplied as arguments in the form of "`-s <ident>:<type>=<value>`". This overrides the default command (s1kd-instance).
-k, --args <args>	The arguments to the filter command when filtering objects prior to validation.
-l, --list	Treat input as a list of CSDB objects to validate.
-N, --omit-issue	Assume that the issue/inwork numbers are omitted from object filenames (they were created with the -N option).
-n, --nested	Check that all product attribute and condition values used in nested applicability annotations are subsets of the values used in their parents.

-o, --output-valid	Output valid CSDB objects to stdout.
-P, --pct <file>	Specify the PCT to read product instances from. This will override the PCT reference in the ACT.
-p, --progress	Display a progress bar.
-q, --quiet	Quiet mode. Error messages will not be printed.
-r, --recursive	Search for the ACT/CCT/PCT recursively.
-s, --strict	Check whether product attributes and conditions used by an object are declared in the ACT and CCT respectively.
-T, --summary	Print a summary of the check after it completes, including statistics on the number of objects that passed/failed the check.
-t, --products	Validate objects against the defined product instances within the PCT.
-v, --verbose	Verbose output. Specify multiple times to increase the verbosity.
-x, --xml	Print an XML report of the check.
-~, --dependencies	Check with CCT dependency tests added to assertions which use the dependant values.
--version	Show version information.
<object>...	Object(s) to validate.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Exit status

0	The check completed successfully, and all CSDB objects were valid.
1	The check completed successfully, but some CSDB objects were invalid.
2	One or more CSDB objects could not be read.
3	The number of CSDB objects specified exceeded the available memory.

5 Examples

5.1 Standalone validation

Consider the following data module snippet:

```
<dmodule>
...
<applic>
<displayText>
<simplePara>Version: A or Version: B</simplePara>
</displayText>
<evaluate andOr="or">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
</evaluate>
</applic>
...
<referencedApplicGroup>
<applic id="app-VersionB">
<assert applicPropertyIdent="version" applicPropertyType="prodattr"
applicPropertyValues="B"/>
</applic>
</referencedApplicGroup>
...
<levelledPara id="par-0001" applicRefId="app-VersionB">
<title>Features of version B</title>
<para>...</para>
</levelledPara>
...
<levelledPara>
<title>More information</title>
<para>...</para>
<para>Refer to <internalRef internalRefId="par-0001"/>.</para>
</levelledPara>
...
</dmodule>
```

There are two versions of the product, A and B, and the data module is meant to apply to both.

By itself, the data module is valid:

```
$ s1kd-validate -v <DM>
s1kd-validate: SUCCESS: <DM> validates against schema <url>
```

Checking it with this tool, however, reveals an issue:

```
$ s1kd-appcheck <DM>
s1kd-appcheck: ERROR: <DM> is invalid when:
```

```
s1kd-appcheck: ERROR:   prodattr version = A
```

When the data module is filtered for version A, the first levelled paragraph will be removed, which causes the reference to it in the second levelled paragraph to become broken.

5.2 Full validation

Consider the following data module snippet:

```
<dmodule>
...
<applic>
<displayText>
<simplePara>All</simplePara>
</displayText>
</applic>
...
<referencedApplicGroup>
<applic id="app-IcyOrHot">
<evaluate andOr="or">
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="Icy"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="Hot"/>
</applic>
</referencedApplicGroup>
...
<proceduralStep>
<para>Locate the handle.</para>
</proceduralStep>
<proceduralStep id="stp-0001" applicRefId="app-IcyOrHot">
<para>Put on gloves prior to touching the handle.</para>
</proceduralStep>
<proceduralStep>
<para>Grab the handle and turn it clockwise.</para>
</proceduralStep>
...
<proceduralStep>
<para>Remove the gloves you put on in <internalRef internalRefId="stp-0001"/>.
</proceduralStep>
...
</dmodule>
```

Once again, this data module is valid by itself:

```
$ s1kd-validate -v <DM>
```

```
s1kd-validate: SUCCESS: <DM> validates against schema <url>
```

This time, however, it also initially appears valid when this tool is used:

```
$ s1kd-appcheck -v <DM>
s1kd-appcheck: SUCCESS: <DM> passed the applicability check.
```

However, now consider this snippet from the CCT:

```
<condCrossRefTable>
...
<condType id="weatherType">
<name>Weather type</name>
<descr>Possible types of weather conditions.</descr>
<enumeration applicPropertyValues="Normal"/>
<enumeration applicPropertyValues="Icy"/>
<enumeration applicPropertyValues="Hot"/>
</condType>
...
<cond id="weather" condTypeRefId="weatherType">
<name>Weather</name>
<descr>The current weather conditions.</descr>
</cond>
...
</condCrossRefTable>
```

There is a third value for the `weather` condition which is not explicitly used within the data module, and therefore will not be validated against in the default standalone check. When `weather` has a value of `Normal`, the cross-reference in the last step in the example above becomes broken.

To catch errors with implicit applicability, the full check (-a) can be used instead, which reads the values to check not from the data module itself, but from the ACT and CCT referenced by the data module:

```
$ s1kd-appcheck -a <DM>
s1kd-appcheck: ERROR: <DM> is invalid when:
s1kd-appcheck: ERROR:   condition weather = Normal
```

This can also be fixed by making the applicability of the data module explicit:

```
<applic>
<displayText>
<simplePara>Weather: Normal or Weather: Icy or
Weather: Hot</simplePara>
</displayText>
<evaluate andOr="or">
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
```

```
applicPropertyValues="Normal"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="Icy"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="Hot"/>
</evaluate>
</applic>
```

In which case, the standalone check will now also detect the error:

```
$ s1kd-appcheck <DM>
s1kd-appcheck: ERROR: <DM> is invalid when:
s1kd-appcheck: ERROR:   condition weather = Normal
```

5.3 Nested applicability annotations

Consider the following data module snippet:

```
<dmodule>
...
<applic>
<displayText>
<simplePara>Version: A, B</simplePara>
</displayText>
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
</applic>
...
<referencedApplicGroup>
<applic id="app-C">
<displayText>
<simplePara>Version: C</simplePara>
</displayText>
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="C"/>
</applic>
</referencedApplicGroup>
...
```

```
<proceduralStep>
<para>Step A</para>
</proceduralStep>
<proceduralStep applicRefId="app-C">
<para>Step B</para>
</proceduralStep>
<proceduralStep>
<para>Step C</para>
</proceduralStep>
...
</dmodule>
```

Here, the whole data module is applicable to versions A and B, but an individual step has been made applicable to version C. Normally, this is not reported as an error, since the removal of this step would not cause the data module to become invalid:

```
$ s1kd-appcheck -v <DM>
s1kd-appcheck: SUCCESS: <DM> passed the applicability check
```

However, the content is essentially useless, since it will never appear. The -n option will report when the applicability of an element is incompatible with the applicability of any parent elements or the whole object:

```
$ s1kd-appcheck -n <DM>
s1kd-appcheck: ERROR: <DM>: proceduralStep on line 62 is applicable
when prodattr version = C, which is not a subset of the applicability
of the whole object.
```

s1kd-brexcheck

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-brexcheck** tool validates S1000D CSDB objects using the context, SNS, and/or notation rules of one or multiple BREX data modules. All errors are displayed with the <objectUse> message, the line number, and a representation of the invalid XML tree.

2 Usage

```
s1kd-brexcheck [-b <brex>] [-d <dir>] [-I <path>] [-w <severities>]
               [-BcefLlnopqrS[tu]sTvxh?] [<object>...]
```

3 Options

-B, --default-brex	Check each input object against the appropriate built-in S1000D default BREX only. The actual BREX reference of each object is ignored.
-b, --brex <brex>	Check the CSDB objects against this BREX. Multiple BREX data modules can be specified by adding this option multiple times. When no BREX data modules are specified, the BREX data module referenced in <brexDmRef> in the CSDB object is attempted to be used instead.
-c, --values	When a context rule defines values for an object (objectValue), check if the value of each object is within the allowed set of values.
-d, --dir <dir>	Directory to start searching for BREX data modules in. By default, the current directory is used.
-e, --ignore-empty	Ignore check for empty or non-XML documents.
-f, --overwrite	Print the filenames of CSDB objects with BREX/SNS errors.
-h, -?, --help	Show the help/usage message.
-I, --include <path>	Add a search path for BREX data modules. By default, only the current directory is searched.
-L, --list	Treat input as a list of object filenames to check, rather than an object itself.
-l, --layered	Use the layered BREX concept. BREX data modules referenced by other BREX data modules (either specified with -b or referenced by the specified CSDB objects) will also be checked against.
-n, --notations	Check notation rules. Any notation names listed in any of the BREX data modules with attribute `allowedNotationFlag` set to "1" or omitted are considered valid notations. If a notation in a CSDB object is not present or has `allowedNotationFlag` set to "0", an error will be returned. For notations not included but not explicitly excluded, the `objectUse` of the first inclusion rule will be returned with the error. For explicitly excluded notations, the `objectUse` of the explicit exclusion rule is returned.
-o, --output-valid	Output valid CSDB objects to stdout.
-p, --progress	Display a progress bar.
-q, --quiet	Quiet mode. No errors are printed, they are only indicated via the exit status.
-r, --recursive	Search for BREX data modules recursively.

-S[tu], --sns [--strict|--unstrict]	Check SNS rules. The SNS of each specified data module is checked against the combination of all SNS rules of all specified BREX data modules.
-s, --short	Use shortened, single-line messages to report BREX errors instead of multiline indented messages.
-T, --summary	Print a summary of the check after it completes, including statistics on the number of documents that passed/failed the check.
-v, --verbose	Verbose mode. The success or failure of each test is printed explicitly.
-w, --severity-levels <file>	Specify a list of severity levels for business rules.
-x, --xml	Output an XML report.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 Business rule severity levels (`.brseveritylevels`)

The attribute `brSeverityLevel` on a BREX rule allows for distinguishing different kinds of errors. The `.brseveritylevels` file contains a list of severity levels, their user-defined type, and optionally if they should not be counted as true errors (causing the tool to return a "failure" status) but merely warnings.

By default, the program will search the current directory and parent directories for a file named `.brseveritylevels`, but any file can be specified by using the -w option.

An example of the format of this file is given below:

```
<?xml version="1.0"?>
<brSeverityLevels>
<brSeverityLevel value="brsl01" fail="yes">Error</brSeverityLevel>
<brSeverityLevel value="brsl02" fail="no">Warning</brSeverityLevel>
</brSeverityLevels>
```

When the attribute `fail` has a value of `"yes"` (or is not included), BREX errors pertaining to rules with the given severity level value will be counted as errors. When it is `"no"`, the errors are still displayed but are not counted as errors in the exit status code of the tool.

3.2 Normal, strict and unstrict SNS check (-S, -St, -Su)

There are three modes for SNS checking: normal, strict, and unstrict. The main difference between them is how they handle the optional levels of an SNS description in the BREX.

-St enables **strict** SNS checking. By default, the normal SNS check (-S) will assume optional elements snsSubSystem, snsSubSubSystem, and snsAssy exist with an snsCode of "0" ("00" or "0000" for snsAssy) when their parent element does not contain any of each. This provides a shorthand, such that

```
<snsSystem>
<snsCode>00</snsCode>
<snsTitle>General</snsTitle>
</snsSystem>
```

is equivalent to

```
<snsSystem>
<snsCode>00</snsCode>
<snsTitle>General</snsTitle>
<snsSubSystem>
<snsCode>0</snsCode>
<snsTitle>General</snsTitle>
<snsSubSubSystem>
<snsCode>0</snsCode>
<snsTitle>General</snsTitle>
<snsAssy>
<snsCode>00</snsCode>
<snsTitle>General</snsTitle>
</snsAssy>
</snsSubSubSystem>
</snsSubSystem>
</snsSystem>
```

Using strict checking will disable this shorthand, and missing optional elements will result in an error.

-Su enables **unstrict** SNS checking. The normal SNS check (-S) shorthand mentioned above only allows SNS codes of "0" to be omitted from the SNS rules. Using unstrict checking, **any** code used will not produce an error when the relevant optional elements are omitted. This means that given the following...

```
<snsSystem>
<snsCode>00</snsCode>
<snsTitle>General</snsTitle>
</snsSystem>
```

...SNS codes of 00-00-0000 through 00-ZZ-ZZZZ are considered valid.

3.3 Object value checking (-c)

There are two ways to restrict the allowable values of an object in a BREX rule. One is to use the XPath expression itself. For example, this expression will match any `securityClassification` attribute whose value is neither `"01"` nor `"02"`, and because the `allowedObjectFlag` is `"0"`, will generate a BREX error if any match is found:

```
<objectPath allowedObjectFlag="0">
```

```
//@securityClassification[
. != '01' and
. != '02'
]
</objectPath>
```

However, this method can lead to fairly complex expressions and requires a reversal of logic. The BREX schema provides an alternative method using the element `objectValue`:

```
<structureObjectRule>
<objectPath allowedObjectFlag="2">
//@securityClassification
</objectPath>
<objectValue valueAllowed="01">Unclassified</objectValue>
<objectValue valueAllowed="02">Classified</objectValue>
</structureObjectRule>
```

Specifying the -c option will enable checking of these types of rules, and if the value is not within the allowed set a BREX error will be reported. The `valueForm` attribute can be used to specify what kind of notation the `valueAllowed` attribute will contain:

– `"single"` - A single, exact value.
– `"range"` - Values given in the S1000D range/set notation, e.g. `"a~c"` or `"a|b|c"`.
– `"pattern"` - A regular expression.

The s1kd-brexcheck tool supports all three types. If the `valueForm` attribute is omitted, it will assume the value is in the `"single"` notation.

4 Exit status

0	Check completed successfully, and no CSDB objects had BREX errors.
1	Check completed successfully, but some CSDB objects had BREX errors.
2	One or more CSDB objects specified could not be read.
3	A referenced BREX data module could not be found.
4	An XPath expression given for a BREX rule was invalid.
5	The number of paths or CSDB objects specified exceeded the available memory.

5 Example

```
$ DMOD=DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
$ BREX=DMC-S1000D-G-04-10-0301-00A-022A-D_001-00_EN-US.XML
$ cat $DMOD
[...]
<listItem id="stp-0001">
```

```
<para>List items shouldn't be used as steps...</para>
</listItem>
[...]
<para>Refer to <internalRef internalRefId="stp-0001"
internalRefTargetType="irtt08"/>.</para>
[...]

$ s1kd-brexcheck -b $BREX $DMOD
BREX ERROR: DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
  BREX: DMC-S1000D-G-04-10-0301-00A-022A-D_001-00_EN-US.XML
  BREX-S1-00052
  Only when the reference target is a step can the value of attribute
internalRefTargetType be irtt08 (Chap 3.9.5.2.1.2, Para 2.1).
  line 52 (/dmodule[1]/content[1]/description[1]/para[2]/
internalRef[1]):
    ELEMENT internalRef
      ATTRIBUTE internalRefTargetType
        TEXT
          content=irtt08
      ATTRIBUTE internalRefId
        TEXT
          content=stp-0001
```

Example of XML report format for the above:

```
<?xml version="1.0"?>
<brexCheck>
<document path="DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML">
<brex path="DMC-S1000D-G-04-10-0301-00A-022A-D_001-00_EN-US.XML">
<error fail="yes">
<brDecisionRef brDecisionIdentNumber="BREX-S1-00052"/>
<objectPath allowedObjectFlag="0">...</objectPath>
<objectUse>Only when the refernce target is a step can the value of
attribute internalRefTargetType be irtt08
(Chap 3.9.5.2.1.2, Para 2.1).</objectUse>
<object line="52"
xpath="/dmodule[1]/content[1]/description[1]/para[2]/internalRef[1]">
<internalRef internalRefId="stp-0001"
internalRefTargetType="irtt08"/>
</object>
</error>
</brex>
</document>
</brexCheck>
```

s1kd-refs

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-refs** tool lists external references in CSDB objects, optionally matching them to a filename in the CSDB directory hierarchy.

This allows you to:

– obtain a list of dependencies for CSDB objects, such as ICNs, to ensure they are delivered together

– check for references to objects which do not exist in the current CSDB

– update reference metadata, such as titles, from the matched objects

2 Usage

```
s1kd-refs [-aCcDEFfGHIiLlmNnoPqRrSsTUuvwXxh?] [-d <dir>] [-e <cmd>]
```

```
[-J <ns=URL>] [-j <xpath>] [-t <fmt>] [-3 <file>]
[<object>...]
```

3 Options

-a, --all	List all references, both matched and unmatched.
-C, -D, -E, -G, -H, -L, -P, -S, -T	List references to comments, data modules, external publications, ICNs, hotspots, data management lists, publication modules, SCORM content packages and referred fragments respectively. If none are specified, -CDEGHLPST is assumed. The following long options can also be used for each: --com, --dm, --epr, --icn, --hotspot, --dml, --pm, --smc, --fragment.
-c, --content	List references in the `content` section of a CSDB object only.
-d, --dir <dir>	Directory to search for matches to references in. By default, the current directory is used.
-e, --exec <cmd>	Execute a command for each referenced CSDB object matched. The string "{}" is replaced by the current CSDB object file name everywhere it occurs in the arguments to the command.
-F, --overwrite	When using the -U or -X options, overwrite the input objects that have been updated or tagged.
-f, --filename	Include the filename of the source object where each reference was found in the output.
-h, -?, --help	Show help/usage message.
-I, --update-issue	Update the issue number, issue date, language, and title of references to that of the latest matched object. This option implies the -U and -i options.
-i, --ignore-issue	Ignore issue info when matching. This will always match the latest issue of an object found, regardless of the issue specified in the reference.
-J <ns=URL>	Registers an XML namespace prefix, which can then be used in the hotspot XPath expression (-j). Multiple namespaces can be registered by specifying this option multiple times.
-j <xpath>	Specify a custom XPath expression to use when matching hotspots (-H) in XML-based ICN formats.
-l, --list	Treat input (stdin or arguments) as lists of filenames of CSDB objects to list references in, rather than CSDB objects themselves.
-m, --strict-match	Be more strict when matching codes of CSDB objects to filenames. By default, the name of a file (minus the extension)

	only needs to start with the code to be matched. When this option is specified, the name must match the code exactly. For example, the code "ABC" will normally match either of the files "ABC.PDF" or "ABC_1.PDF", but when strict matching is enabled, it will only match the former.
-N, --omit-issue	Assume filenames of referenced CSDB objects omit the issue info, i.e. they were created with the -N option to the s1kd-new* tools.
-n, --lineno	Include the filename of the source object where each reference was found, and display the line number where the reference occurs in the source file after its filename.
-o, --output-valid	Output valid CSDB objects to stdout.
-q, --quiet	Quiet mode. Errors are not printed.
-R, --recursively	List references in matched objects recursively.
-r, --recursive	Search for matches to references in directories recursively.
-s, --include-src	Include the source object as a reference. This is helpful when the output of this tool is used to apply some operation to a source object and all its dependencies together.
-t, --format <fmt>	Specify a custom format for printed references. <fmt> is a format string, where the following variables can be given: – %line% - The line number where the reference occurs in the source. – %ref% - The reference. – %src% - The source of the reference. – %xpath% - The XPath denoting where the reference occurs in the source. For example, `-t '%src% (%line%): %ref%'` is equivalent to the -n option.
-U, --update	Update the title of matched references from the corresponding object.
-u, --unmatched	Show only unmatched reference errors, or unmatched codes if combined with the -a option.
-v, --verbose	Verbose output.
-w, --where-used	Instead of listing references contained within specified objects, list places within other objects where the specified objects are referenced.
-X, --tag-unmatched	Tag unmatched references with the processing instruction <?`unmatched?>`.
-x, --xml	Output a detailed XML report instead of plain text messages.

-3, --externalpubs <file>	Use a custom `.externalpubs` file.
--version	Show version information.
<object>...	CSDB object(s) to list references in. If none are specified, the tool will read from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.externalpubs` file

The `.externalpubs` file contains definitions of external publication references. This can be used to update external publication references in CSDB objects with -U.

By default, the tool will search the current directory and parent directories for a file named .externalpubs, but any file can be specified by using the -e option.

Example of a `.externalpubs` file:

```
<externalPubs>
<externalPubRef>
<externalPubRefIdent>
<externalPubCode>ABC</externalPubCode>
<externalPubTitle>ABC Manual</externalPubTitle>
</externalPubRefIdent>
</externalPubRef>
</externalPubs>
```

External publication references will be updated whether they are matched to a file or not.

3.2 Hotspot matching (-H)

Hotspots can be matched in XML-based ICN formats, such as SVG or X3D. By default, matching is based on the APS ID of the hotspot and the following attributes:

SVG	`@id`
X3D	`@DEF`

If hotspots are identified in a different way in a project's ICNs, a custom XPath expression can be specified with the -j option. In this XPath expression, the variable `$id` represents the hotspot APS ID:

```
$ s1kd-refs -H -j "//*[@attr = $id]" <DM>
```

4 Exit status

0	No errors, all references were matched.

1	Some references were unmatched.
2	The number of objects found in a recursive check (-R) exceeded the available memory.
3	stdin did not contain valid XML and not in list mode (-l).

5 Example

```
$ s1kd-refs DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
DMC-EX-A-00-00-00-00A-022A-D_001-00_EN-CA.XML
DMC-EX-A-01-00-00-00A-040A-D_000-01_EN-CA.XML
ICN-12345-00001-001-01.JPG
```

s1kd-repcheck

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-repcheck** tool validates references to CIR items within S1000D CSDB objects. Any CIR references which cannot be resolved to a specification within a CIR data module will cause the tool to report an error.

2 Usage

```
s1kd-repcheck [options] [<objects>...]
```

3 Options

-a, --all

In addition to CIR data modules specified with -R or explicitly linked in CIR references, allow CIR references to be resolved against any CIR data modules that were specified as objects to check.

-d, --dir <dir>	The directory to start searching for CIR data modules in. By default, the current directory is used.
-f, --filenames	Print the filenames of invalid objects.
-h, -?, --help	Show help/usage message.
-l, --list	Treat input as a list of CSDB objects to check.
-N, --omit-issue	Assume that the issue/inwork numbers are omitted from object filenames (they were created with the -N option).
-o, --output-valid	Output valid CSDB objects to stdout.
-p, --progress	Display a progress bar.
-q, --quiet	Quiet mode. Error messages will not be printed.
-R, --cir <CIR>	A CIR to resolve references in CSDB objects against. Multiple CIRs can be specified by using this option multiple times. If "*" is given for <CIR>, the tool will search for CIR data modules automatically.
-r, --recursive	Search for CIR data modules recursively.
-T, --summary	Print a summary of the check after it completes, including statistics on the number of objects that passed/failed the check.
-v, --verbose	Verbose output. Specify multiple times to increase the verbosity.
-x, --xml	Print an XML report of the check.
--version	Show version information.
<object>...	Object(s) to check CIR references in.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Exit status

0	The check completed successfully, and all CIR references were resolved.
1	The check completed successfully, but some CIR references could not be resolved.
2	The number of CSDB objects specified exceeded the available memory.

5 Example

Part repository:

```
<partRepository>
<partSpec>
<partIdent manufacturerCodeValue="12345" partNumberValue="ABC"/>
<itemIdentData>
<descrForPart>ABC part</descrForPart>
</itemIdentData>
</partSpec>
</partRepository>
```

Part references in a procedure:

```
<spareDescrGroup>
<spareDescr>
<partRef manufacturerCodeValue="12345" partNumberValue="ABC"/>
<reqQuantity>1</reqQuantity>
</spareDescr>
<spareDescr>
<partRef manufacturerCodeValue="12345" partNumberValue="DEF"/>
<reqQuantity>1</reqQuantity>
</spareDescr>
</spareDescrGroup>
```

Command and results:

```
$ s1kd-repcheck -R <CIR> ... <DM>
s1kd-repcheck: ERROR: <DM> (<line>): Part 12345/DEF not found.
```

s1kd-validate

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-validate** tool validates S1000D CSDB objects, checking whether they are valid XML files and if they are valid against their own S1000D schemas.

2 Usage

```
s1kd-validate [-d <dir>] [-s <path>] [-x <URI>] [-efloqvh?]
              [<object>...]
```

3 Options

-d, --schemas <dir> — Search for schemas in <dir>. Normally, the URI of the schema is used to fetch it locally or over a network, but

	this option will force searching to be performed only in the specified directory. This can also be accomplished through the use of XML catalogs.
-e, --ignore-empty	Ignore validation for empty or non-XML documents.
-f, --filenames	List invalid files.
-h, -?, --help	Show help/usage message.
-l, --list	Treat input as a list of object names to validate, rather than an object itself.
-o, --output-valid	Output valid CSDB objects to stdout.
-q, --quiet	Quiet mode. The tool will not output anything to stdout or stderr. Success/failure will only be indicated through the exit status.
-s, --schema <path>	Validate the objects against the specified schema, rather than the one that they reference.
-v, --verbose	Verbose mode. Success/failure will be explicitly reported on top of any errors.
-x, --exclude <URI>	Exclude an XML namespace from the validation. Elements in the namespace specified by <URI> are ignored.
--version	Show version information.
<object>...	Any number of CSDB objects to validate. If none are specified, input is read from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Substitute entities.
--xinclude	Do XInclude processing.

3.1 Multi-spec directory with -d option

The -d option can point either to a directory containing the XSD schema files for a single S1000D spec (i.e. the last part of the schema URI), or to a directory containing schemas for multiple specs. The latter must follow a particular format for the tool to locate the appropriate schemas for a given spec:

```
schemas/    <-- The directory passed to -d
    S1000D_4-1/
        xml_schema_flat/
            [4.1 XSD files...]
    S1000D_4-2/
        xml_schema_flat/
```

```
            [4.2 XSD files...]
    S1000D_5-0/
        xml_schema_flat/
            [5.0 XSD files...]
```

3.2 XML catalogs vs. -d option

XML catalogs provide a more standard method of redirecting public, network-based resources to local copies. As part of using libxml2, there are several locations and environment variables from which this tool will load catalogs.

Below is an example of a catalog file which maps the S1000D schemas to a local directory:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
<rewriteURI uriStartString="http://www.s1000d.org/"
rewritePrefix="/usr/share/s1kd/schemas/"/>
</catalog>
```

This can be placed in a catalog file automatically loaded by libxml2 (e.g., `/etc/xml/catalog`) or saved to a file which is then specified in an environment variable used by libxml2 (e.g., `XML_CATALOG_FILES`) to remove the need to use the -d option.

4 Exit status

0	No errors.
1	Some CSDB objects are not well-formed or valid.
2	The number of schemas cached exceeded the available memory.
3	A specified schema could not be read.

5 Example

```
$ s1kd-validate DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
```

s1kd-acronyms

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-acronyms** tool is used to manage acronyms in S1000D data modules in one of three ways:

– Generate a list of unique acronyms used in all specified data modules.

– Mark up acronyms automatically based on a specified list.

– Remove acronym markup.

2 Usage

```
s1kd-acronyms -h?
s1kd-acronyms [-dlptvx] [-n <#>] [-o <file>] [-T <types>]
              [<dmodule>...]
s1kd-acronyms [-flv] [-i|-I|-!] [-m|-M <acr>] [-o <file>] [-X <xpath>]
```

```
              [<dmodule>...]
s1kd-acronyms -D [-flv] [-o <file>] [<dmodule>...]
```

3 Options

-D, --delete	Remove acronym markup, flattening it to the acronym term.
-d, --deflist	Format XML output as an S1000D `<definitionList>`.
-f, --overwrite	When marking up acronyms with the -m option, overwrite the input data modules instead of writing to stdout.
-h, -?, --help	Show help/usage message.
-I, --always-ask	In interactive mode, show a prompt for all acronyms, not just those with multiple definitions. This can be useful if some occurrences of acronym terms should be ignored.
-i, --interactive	Markup acronyms in interactive mode. If the specified acronyms list contains multiple definitions for a given acronym term, the tool will prompt the user with the context in which the acronym is used and present a list of the definitions for them to choose from. When not in interactive mode, the first definition found will be used.
-l, --list	Treat input (stdin or arguments) as lists of filenames of data modules to find or markup acronyms in, rather than data modules themselves.
-M, --acronym-list <list>	Like the -m option, but use a custom list of acronyms instead of the default `.acronyms` file.
-m, --markup	Instead of listing acronyms in the specified data modules, automatically markup acronyms in the data module using the `.acronyms` file.
-n, --width <#>	Minimum number of spaces after the term in pretty-printed text output.
-o, --out <file>	Output to <file> instead of stdout.
-p, --pretty	Pretty print text/XML acronym list output.
-T, --types <types>	Only search for acronyms with an attribute `acronymType` whose value is contained within the string <types>.
-t, --table	Format XML output as an S1000D `<table>`.
-v, --verbose	Verbose output.
-X, --select <xpath>	When marking up acronyms with -m/-M, use a custom XPath expression to specify which text nodes to search for acronyms in. By default, this is all text nodes in any element where acronyms are allowed. This must be the path to the text() nodes, not the elements, e.g. `//para/text()` and not simply `//para`.

-x, --xml	Use XML output instead of plain text.
-!, --defer-choice	Mark where acronyms are found using a `<chooseAcronym>` element, whose child elements are all possible acronyms matching the term. Another program can then use this as input to actually prompt the user.
--version	Show version information.
<dmodule>...	Data modules to find acronyms in. If none are specified, input is taken from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.acronyms` file

This file specifies a list of acronyms for a project. By default, the program will search for a file named `.acronyms` in the current directory and parent directories, but any file can be specified using the -M option.

Example of .acronyms file format:

```
<acronyms>
<acronym acronymType="at01">
<acronymTerm>BREX</acronymTerm>
<acronymDefinition>Business Rules Exchange</acronymDefinition>
</acronym>
<acronym acronymType="at01">
<acronymTerm>SNS</acronymTerm>
<acronymDefinition>Standard Numbering System</acronymDefinition>
</acronym>
</acronyms>
```

4 Examples

List all acronyms used in all data modules:

```
$ s1kd-acronyms DMC-*.XML
```

Markup predefined acronyms in a data module:

```
$ s1kd-acronyms -mf DMC-EX-A-00-00-00-00A-040A-D_EN-CA.XML
```

Unmarkup acronyms in a data module:

```
$ s1kd-acronyms -Df DMC-EX-A-00-00-00-00A-040A-D_EN-CA.XML
```

s1kd-aspp

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-aspp** tool has two main functions:

– Generates display text for applicability statements. The text is derived from the logic described by the `assert` and `evaluate` elements.

– Preprocesses "semantic" applicability statements in a data module to produce "presentation" applicability statements which are simpler to parse in an XSLT stylesheet.

"Semantic" applicability statements are those entered by the author to encode the applicability of elements within a data module. "Presentation" applicability statements are those that are actually displayed in page-oriented output, also referred to as the "human-readable" statements.

The applicability in the resulting XML is longer semantically correct, but an XSLT stylesheet can simply place a statement on any element with attribute `applicRefId` without needing to consider inherited applicability statements on elements without the attribute.

2 Usage

```
s1kd-aspp -h?
s1kd-aspp -D
s1kd-aspp -g [-A <ACT>] [-C <CCT>] [-d <dir>] [-F <fmt>] [-G <XSL>]
             [-cfklNrv] [<object>...]
s1kd-aspp -p [-a <ID>] [-flv] [<object>...]
```

3 Options

-A, --act <ACT>	Add an ACT to use when generating display text for product attributes. Multiple ACT data modules can be used by specifying this option multiple times.
-a, --id <ID>	The ID to use for the inline applicability annotation representing the whole data module's applicability. Default is "app-0000".
-C, --cct <CCT>	Add a CCT to use when generating display text for conditions. Multiple CCT data modules can be used by specifying this option multiple times.
-c, --search	Search for the ACT and CCT referenced by each data module, and add them to the list of ACTs/CCTs to use when generating display text for that data module.
-D, --dump	Dump the built-in XSLT used to generate display text for applicability statements.
-d, --dir <dir>	Directory to start searching for ACT/CCT data modules in. By default, the current directory is used.
-F, --format <fmt>	Use a custom format string to generate display text.
-f, --overwrite	Overwrite input data module(s) rather than outputting to stdout.
-G, --xsl <XSLT>	Use custom XSLT to generate display text for applicability statements.
-g, --generate	Generate display text for applicability statements.
-h, -?, --help	Show help/usage message.
-k, --keep	When generating display text, do not overwrite existing display text on statements, only generate display text for statements which have none.
-l, --list	Treat input (stdin or arguments) as lists of filenames of objects, rather than objects themselves.
-N, --omit-issue	Assume that the filenames for the ACT and CCT do not include issue info, i.e. they were created using the -N option of the s1kd-newdm tool.
-p, --presentation	Preprocess applicability statements to produce "presentation" applicability statements which are simpler to parse in an

	XSLT stylesheet. The applicability in the resulting XML is no longer semantically correct.
-r, --recursive	Search for ACT/CCT data modules recursively.
-v, --verbose	Verbose output.
--version	Show version information.
<object>...	The object(s) to preprocess. This can include both individual objects and combined files such as those produced by s1kd-flatten(1).

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Examples

4.1 Generating display text

The built-in XSLT for generating display text follows the guidance in Chap 7.8 of the S1000D 5.0 specification. For example, given the following:

```
<applic>
<assert applicPropertyIdent="prodversion"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</applic>
```

The resulting XML would contain:

```
<applic>
<displayText>
<simplePara>prodversion: A</simplePara>
</displayText>
<assert applicPropertyIdent="prodversion"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</applic>
```

If ACTs or CCTs are supplied which define display names for a property, this will be used instead of the ident. For example, the ACT defines the display name for the "`prodversion`" product attribute:

```
<productAttribute id="prodversion">
<name>Product version</name>
<displayName>Version</displayName>
<descr>The version of the product.</descr>
<enumeration applicPropertyValues="A|B|C"/>
</productAttribute>
```

When supplied with the -A option:

```
$ s1kd-aspp -g -A <ACT> <DM>
```

The resulting XML would instead contain:

```
<applic>
<displayText>
<simplePara>Version: A</simplePara>
<assert applicPropertyIdent="prodversion"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</displayText>
</applic>
```

The methods for generating display text can be changed by supplying a custom XSLT script with the -G option. The -D option can be used to dump the built-in XSLT as a starting point for a custom script. An identity template is automatically added to the script, equivalent to the following:

```
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()"/>
</xsl:copy>
</xsl:template>
```

This means any elements or attributes not matched by a more specific template in the script are copied.

4.2 Display text format string (-F)

The -F option allows for simple customizations to generated display text without needing to create a custom XSLT script (-G). The string determines the format of the display text of each `<assert>` element in the annotation.

The following variables can be used within the format string:

%name%	The name of the property.
%values%	The applicable value(s) of the property.

For example:

```
$ s1kd-aspp -g <DM>
...
<applic>
<displayText>
<simplePara>Version: A</simplePara>
</displayText>
<assert applicPropertyIdent="version" applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
...
```

```
$ s1kd-aspp -F '%name% = %values%' -g <DM>
...
<applic>
<displayText>
<simplePara>Version = A</simplePara>
</displayText>
<assert applicPropertyIdent="version" applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
...
```

4.3 Creating presentation applicability statements

Given the following:

```
<dmodule>
<identAndStatusSection>
<dmAddress>...</dmAddress>
<dmStatus>
...
<applic>
<displayText>
<simplePara>A or B</simplePara>
</displayText>
</applic>
...
</dmStatus>
</identAndStatusSection>
<content>
<referencedApplicGroup>
<applic id="app-B">
<displayText>
<simplePara>B</simplePara>
</displayText>
</applic>
</referencedApplicGroup>
<procedure>
<preliminaryRqmts>...</preliminaryRqmts>
<mainProcedure>
<proceduralStep>
<para>This step is applicable to A or B.</para>
</proceduralStep>
<proceduralStep applicRefId="app-B">
<para>This step is applicable to B only.</para>
</proceduralStep>
<proceduralStep applicRefId="app-B">
<para>This step is also applicable to B only.</para>
</proceduralStep>
<proceduralStep>
```

```
<para>This step is also applicable to A or B.</para>
</proceduralStep>
</mainProcedure>
<closeRqmts>...</closeRqmts>
</procedure>
</content>
</dmodule>
```

Applicability statements should be displayed whenever applicability changes:

1 This step is applicable to A or B.

2 **Applicable to: B**

This step is applicable to B only.

3 This step is also applicable to B only.

4 **Applicable to: A or B**

This step is also applicable to A or B.

There are two parts which are difficult to do in an XSLT stylesheet:

– No statement is shown on Step 3 despite having attribute `applicRefId` because the applicability has not changed since the last statement on Step 2.

– A statement is shown on Step 4 despite not having attribute `applicRefId` because the applicability has changed back to that of the whole data module.

Using the s1kd-aspp tool, the above XML would produce the following output:

```
<dmodule>
<identAndStatusSection>
<dmAddress>...</dmAddress>
<dmStatus>
...
<applic>
<displayText>
<simplePara>A or B</simplePara>
</displayText>
</applic>
...
</dmStatus>
</identAndStatusSection>
<content>
<referencedApplicGroup>
<applic id="app-B">
<displayText>
<simplePara>B</simplePara>
</displayText>
</applic>
```

```
<applic id="app-0000">
<displayText>
<simplePara>A or B</simplePara>
</displayText>
</applic>
</referencedApplicGroup>
<procedure>
<preliminaryRqmts>...</preliminaryRqmts>
<mainProcedure>
<proceduralStep>
<para>This step is applicable to A or B.</para>
</proceduralStep>
<proceduralStep applicRefId="app-B">
<para>This step is applicable to B only.</para>
</proceduralStep>
<proceduralStep>
<para>This step is also applicable to B only.</para>
</proceduralStep>
<proceduralStep applicRefId="app-0000">
<para>This step is also applicable to A or B.</para>
</proceduralStep>
</mainProcedure>
</procedure>
</content>
</dmodule>
```

With attribute `applicRefId` only on those elements where a statement should be shown, and an additional inline applicability to represent the whole data module's applicability. This XML is semantically incorrect but easier for a stylesheet to transform for page-oriented output.

s1kd-flatten

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-flatten** tool combines a publication module and the data modules it references in to a single file for use with a publishing system.

Data modules are by default searched for in the current directory using the data module code, language and/or issue info provided in each reference.

2 Usage

```
s1kd-flatten [-d <dir>] [-I <path>] [-cDfimNpqRruvx] <PM> [<DM>...]
```

3 Options

-c, --containers — Flatten referenced container data modules by copying the references inside the container directly in to the publication

	module. The copied references will also be flattened, unless the -m option is specified.
-D, --remove	Remove unresolved references.
-d, --dir <dir>	Directory to start search in. By default, the current directory is used.
-f, --overwrite	Overwrite input publication module instead of writing to stdout.
-h, -?, --help	Show help/usage message.
-I, --include <path>	Add <path> to the list of directories that the tool will search when resolving references.
-i, --ignore-issue	Always match the latest issue of an object found, regardless of the issue specified in the reference.
-m, --modify	Modify the references in the publication module without flattening them.
-N, --omit-issue	Assume that the files representing the referenced data modules do not include the issue info in their filenames, i.e. they were created using the -N option of the s1kd-new* tools.
-p, --simple	Instead of the hierarchical PM-based format, use a simpler "flat" format.
-q, --quiet	Quiet mode. Errors are not printed.
-R, --recursively	Recursively flatten referenced publication modules, copying their content in to the "master" publication module.
-r, --recursive	Search directories recursively.
-u, --unique	Remove duplicate references within the PM content.
-v, --verbose	Verbose output. Specify multiple times to increase the verbosity.
-x, --use-xinclude	Use XInclude rather than copying each data module's contents directly inside the publication module. DTD entities in data modules will only be carried over to the final publication when using this option, otherwise they do not carry over when copying the data module.
--version	Show version information.
<DM>...	When using the -p option, the filenames to include can be specified manually as additional arguments instead of searching for them in the current directory. When not using the -p option, additional arguments are ignored.
<PM>	The publication module to flatten.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.

--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Exit status

0	No errors.
1	The publication module specified is malformed.
2	An encoding error occurred.

5 Example

```
$ s1kd-flatten -x PMC-EX-12345-00001-00_001-00_EN-CA.XML > Book.xml
```

s1kd-fmgen

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-fmgen** tool generates the content section for front matter data modules from either a standard publication module, or the combined format of the s1kd-flatten(1) tool. Some front matter types require the use of the combined format, particularly those that list information not directly found in the publication module, such as the highlights (HIGH) type.

2 Usage

```
s1kd-fmgen [-D <TYPE>] [-F <FMTYPES>] [-I <date>] [-P <PM>]
           [-p <name>=<val> ...] [-t <TYPE>] [-x <XSL>]
           [-,.flqvh?] [<DM>...]
```

3 Options

-,, --dump-fmtypes-xml	Dump the built-in `.fmtypes` XML format.
-., --dump-fmtypes	Dump the built-in `.fmtypes` simple text format.
-D, --dump-xsl <TYPE>	Dump the built-in XSLT used to generate the specified type of front matter.
-F, --fmtypes <FMTYPES>	Specify a custom `.fmtypes` file.
-f, --overwrite	Overwrite the specified front matter data module files after generating their content.
-h, -?, --help	Show usage message.
-I, --date <date>	Set the issue date of the generated front matter data modules. This can be a specific date in the form of "YYYY-MM-DD", "-" for the current date, or "pm" to use the issue date of the publication module.
-l, --list	Treat input (stdin or arguments) as lists of front matter data modules to generate content for, rather than data modules themselves. If reading list from stdin, the -P option must be used to specify the publication module.
-P, --pm <PM>	Publication module or s1kd-flatten(1) PM format file to generate contents from. If none is specified, the tool will read from stdin.
-p, --param <name>=<value>	Pass a parameter to the XSLT stylesheets used to generate the front matter content. Multiple parameters can be specified by using this option multiple times. The following parameters are automatically supplied to any stylesheet, and therefore their names should be considered reserved: – `"type"` - The front matter type name (e.g., HIGH) that was matched in the `.fmtypes` file or specified by the user with the -t option.
-q, --quiet	Quiet mode. Do not print errors.
-t, --type <TYPE>	Generate content for this type of front matter. Supported types are: – HIGH - Highlights – LOA - List of abbreviations – LOASD - List of applicable specifications and documentation – LOEDM - List of effective data modules – LOI - List of illustrations – LOS - List of symbols

- LOT - List of terms
- LOTBL - List of tables
- TOC - Table of contents
- TP - Title page

-v, --verbose	Verbose output. Specify multiple times to increase the verbosity.
-x, --xsl <XSL>	Use the specified XSLT script to generate the front matter contents instead of the built-in XSLT or the user-configured XSLT from the `.fmtypes` file.
--version	Show version information.
<DM>...	Front matter data modules to generate content for. If no front matter type can be determined for a data module, it will be ignored.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.fmtypes` file

This file specifies a list of info codes to associate with a particular type of front matter.

Optionally, a path to an XSLT script can be given for each info code, which will be used to generate the front matter instead of the built-in XSLT. The path to an XSLT script will be interpreted relative to the location of the `.fmtypes` file (typically, the top directory of the CSDB). The -D option can be used to dump the built-in XSLT for a type of front matter as a starting point for a custom script.

By default, the program will search for a file named `.fmtypes` in the current directory and parent directories, but any file can be specified using the -F option.

Example of simple text format:

```
001    TP
005    LOA
006    LOT
007    LOS
009    TOC
00A    LOA
00S    LOEDM
00U    HIGH    fm/high.xsl
00V    LOASD
00Z    LOTBL
```

Example of XML format:

```
<fmtypes>
<fm infoCode="001" type="TP"/>
<fm infoCode="005" type="LOA"/>
<fm infoCode="006" type="LOT"/>
<fm infoCode="007" type="LOS"/>
<fm infoCode="009" type="TOC"/>
<fm infoCode="00A" type="LOI"/>
<fm infoCode="00S" type="LOEDM"/>
<fm infoCode="00U" type="HIGH" xsl="fm/high.xsl"/>
<fm infoCode="00V" type="LOASD"/>
<fm infoCode="00Z" type="LOTBL"/>
</fmtypes>
```

The info code of each entry in the `.fmtypes` file may also include an info code variant. This allows different transformations to be used based on the variant:

```
<fmtypes>
<fm infoCode="00UA" type="HIGH" xsl="fm/high.xsl"/>
<fm infoCode="00UB" type="HIGH" xsl="fm/high-updates.xsl"/>
<fm infoCode="00U"  type="HIGH"/>
</fmtypes>
```

In the example above, a highlights data module (00U) with info code variant A will use an XSL transformation that creates a simple highlights, while a highlights data module with info code variant B will use an XSL transformation that creates a highlights with update instructions. All other variants will use the built-in XSLT.

Entries are chosen in the order they are listed in the `.fmtypes` file. An info code which does not specify a variant matches all possible variants.

3.2 Optional title page elements

When re-generating the front matter content for a title page data module, optional elements which cannot be derived from the publication module (such as the product illustration or bar code) will be copied from the source data module when updating it.

4 Exit status

0	No errors.
1	The date specified with -I is invalid.
2	No front matter types were specified.
3	An unknown front matter type was specified.
4	The resulting front matter content could not be merged in to a data module.

5 Example

Generate the content for a title page front matter data module and overwrite the file:

```
$ s1kd-flatten PMC-EX-12345-00001-00_001-00_EN-CA.XML |
> s1kd-fmgen -f DMC-EX-A-00-00-00-00A-001A-D_001-00_EN-CA.XML
```

s1kd-icncatalog

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-icncatalog** tool is used to manage a catalog of ICNs for a project, and to resolve ICNs using this catalog. Resolving an ICN means placing the actual filename of the ICN in to the SYSTEM ID of the ENTITY declaration within CSDB objects.

2 Usage

```
s1kd-icncatalog [options] [<object>...]
```

3 Options

-a, --add <ICN>	Add an ICN to the catalog. Follow with the -u and -n options to specify the URI and notation to use for this ICN. The -m option specifies a media group to add the ICN to.
-C, --create	Create a new empty catalog.
-c, --catalog <catalog>	Specify the catalog file to manage or resolve against. By default, the file `.icncatalog` in the current directory is used. If the current directory does not contain this file, the parent directories will be searched.
-d, --del <ICN>	Delete an ICN from the catalog. The -m option specifies a media group to delete the ICN from.
-f, --overwrite	Overwrite the input CSDB objects when resolving ICNs, or overwrite the catalog file when modifying it. Otherwise, output is written to stdout.
-h, -?, --help	Show help/usage message.
-l, --list	Treat input (stdin or arguments) as lists of filenames of CSDB objects, rather than CSDB objects themselves.
-m, --media <media>	Resolve ICNs for this intended output media. The catalog may contain alternative formats for the same ICN to be used for different output media.
-n, --ndata <notation>	Specify the notation to reference when adding an ICN with the -a option.
-q, --quiet	Quiet mode. Errors are not printed.
-t, --type <type>	Specify the type of catalog entry when adding an ICN with the -a option.
-u, --uri <URI>	Specify the URI when adding an ICN with the -a option.
-v, --verbose	Verbose output.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Examples

4.1 Resolving ICNs to filenames

A CSDB object may reference an ICN as follows:

```
<!NOTATION png SYSTEM "png">
<!ENTITY ICN-12345-00001-001-01 SYSTEM "ICN-12345-00001-001-01.PNG"
NDATA png>
```

The SYSTEM ID of this ENTITY indicates that the ICN file will be in the same directory relative to the CSDB object. However, the ICN files in this example are located in a separate folder called 'graphics'. Rather than manually updating every ENTITY declaration in every CSDB object, a catalog file can be used to map ICNs to actual filenames:

```
<icnCatalog>
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="graphics/ICN-12345-00001-001-01.PNG"/>
</icnCatalog>
```

Then, using this tool, the ICN can be resolved against the catalog:

```
$ s1kd-icncatalog -c <catalog> <object>
```

Producing the following output:

```
<!NOTATION png SYSTEM "png">
<!ENTITY ICN-12345-00001-001-01 SYSTEM
"graphics/ICN-12345-00001-001-01.PNG" NDATA png>
```

4.2 Alternative ICN formats

A catalog can also be used to provide alternative file formats for an ICN depending on the intended output media. For example:

```
<icnCatalog>
<notation name="jpg" systemId="jpg"/>
<notation name="svg" systemId="svg"/>
<media name="pdf">
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="ICN-12345-00001-001-01.JPG" notation="jpg"/>
</media>
<media name="web">
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="ICN-12345-00001-001-01.SVG" notation="svg"/>
</media>
</icnCatalog>
```

The -m option allows for specifying which type of media to resolve for:

```
<!NOTATION png SYSTEM "png">
<!ENTITY ICN-12345-00001-001-01 SYSTEM "ICN-12345-00001-001-01.PNG"
NDATA png>
```

```
$ s1kd-icncatalog -c <catalog> -m pdf <object>

<!NOTATION png SYSTEM "png">
<!NOTATION jpg SYSTEM "jpg">
<!ENTITY ICN-12345-00001-001-01 SYSTEM "ICN-12345-00001-001-01.JPG"
NDATA jpg>

$ s1kd-icncatalog -c <catalog> -m web <object>

<!NOTATION png SYSTEM "png">
<!NOTATION svg SYSTEM "svg">
<!ENTITY ICN-12345-00001-001-01 SYSTEM "ICN-12345-00001-001-01.SVG"
NDATA svg>
```

4.3 Reconstructing ICN entity declarations

Some processing, such as XSL transformations, may remove the DTD and external entity declarations as part of parsing an XML CSDB object. A catalog can be used to restore the necessary external entity declarations afterwards. For example:

```
$ xsltproc ex.xsl <object>
```

The resulting XML will not include a DTD or the external entity declarations for the ICNs referenced in the object, so it will not be valid according to the S1000D schema:

```
$ xsltproc ex.xsl <object> | s1kd-validate
-:49:element graphic: Schemas validity error: Element 'graphic',
attribute 'infoEntityIdent': 'ICN-12345-00001-001-01' is not a valid
value of the atomic type 'xs:ENTITY'.
```

Passing the result to this tool, with a catalog containing all the ICNs used by the project:

```
$ xsltproc ex.xsl <object> | s1kd-icncatalog -c <catalog>
```

will reconstruct the required external entity declarations in the DTD.

Note

The s1kd-tools will copy the DTD and external entity declarations automatically when performing transformations (such as with the s1kd-transform tool), so this is only necessary when using more generic XML tools.

4.4 ICN pattern rules

By default, each catalog entry matches a single ICN, but multiple ICNs can be resolved with a single entry by using a pattern rule. An entry with attribute `type="pattern"` specifies a regular expression to use to match ICNs and a template used to construct the resolved URI:

```
<icn
type="pattern"
infoEntityIdent="ICN-(.{5})-(.*)"
uri="graphics/\1/ICN-\1-\2.PNG"
notation="PNG"/>
```

The above entry would match a series of CAGE-based ICNs, resolving them to a subfolder of 'graphics' based on their CAGE code. Using this entry, the following input:

```
<!DOCTYPE dmodule [
<!NOTATION PNG SYSTEM PNG>
<!ENTITY ICN-12345-00001-001-01
SYSTEM "ICN-12345-00001-001-01"
NDATA PNG>
<!ENTITY ICN-54321-00001-001-01
SYSTEM "ICN-54321-00001-001-01"
NDATA PNG>
]>
```

would be resolved as follows:

```
<!DOCTYPE dmodule [
<!NOTATION PNG SYSTEM PNG>
<!ENTITY ICN-12345-00001-001-01
SYSTEM "graphics/12345/ICN-12345-00001-001-01.PNG"
NDATA PNG>
<!ENTITY ICN-54321-00001-001-01
SYSTEM "graphics/54321/ICN-54321-00001-001-01.PNG"
NDATA PNG>
]>
```

The regular expressions must conform to the extended POSIX regular expression syntax. Backreferences \1 through \9 can be used in the URI template to substitute captured groups.

5 Catalog schema

The following describes the schema of an ICN catalog file.

5.1 Catalog

Markup element: `<icnCatalog>`

Attributes:

– None

Child elements:

– `<notation>`
– `<media>`
– `<icn>`

5.2 Notation

The element `<notation>` represents a NOTATION declaration.

Markup element: `<notation>`

Attributes:

– `name`, the NDATA name.
– `publicId`, the optional PUBLIC ID of the notation.
– `systemId`, the optional SYSTEM ID of the notation.

Child elements:

– None

5.3 Media

The element <media> groups a set of alternative ICN formats for a particular output media type.

Markup element: <media>

Attributes:

– `name`, the identifier of the output media.

Child elements:

– <icn>

5.4 ICN

The element <icn> maps an ICN to a filename and optionally a notation. When this element occurs as a child of a <media> element, it will be used when that output media is specified with the -m option. When it occurs as a child of <icnCatalog>, it will be used if no media is specified.

Markup element: <icn>

Attributes:

– `type`, the type of ICN entry, with one of the following values:
  • `"single"` (D) - Specifies a single ICN to resolve.
  • `"pattern"` - Specifies a pattern to resolve one or more ICNs.
– `infoEntityIdent`, the ICN, or pattern used to match ICNs.
– `uri`, the filename the ICN will resolve to.
– `notation`, a reference to a previously declared <notation> element.

Child elements:

– None

5.5 Example ICN catalog

```
<icnCatalog>
<notation name="jpg" systemId="jpg"/>
<notation name="png" systemId="png"/>
```

```
<notation name="svg" systemId="svg"/>
<media name="pdf">
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="ICN-12345-00001-001-01.JPG" notation="jpg"/>
</media>
<media name="web">
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="ICN-12345-00001-001-01.SVG" notation="svg"/>
</media>
<icn infoEntityIdent="ICN-12345-00001-001-01"
uri="ICN-12345-00001-001-01.PNG" notation="png"/>
</icnCatalog>
```

s1kd-index

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-index** tool adds index flags to a data module based on a user-defined set of keywords.

2 Usage

```
s1kd-index -h?
s1kd-index [-I <index>] [-filv] [<module>...]
s1kd-index -D [-filv] [<module>...]
```

3 Options

-D, --delete	Remove the current index flags from a data module.
-f, --overwrite	Overwrite input module(s).

-h, -?, --help	Show help/usage message.
-I, --indexflags <index>	Flag the terms in the specified <index> XML file instead of the default `.indexflags` file.
-i, --ignore-case	Ignore case when flagging terms.
-l, --list	Treat input (stdin or arguments) as lists of filenames of data modules to add index flags to, rather than data modules themselves.
-v, --verbose	Verbose output.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.indexflags` file

This file specifies the list of indexable keywords for the project and their level. By default, the program will search for a file named `.indexflags` in the current directory or parent directories, but any file can be specified using the -I option.

Exmaple of `.indexflags` file format:

```
<indexFlags>
<indexFlag indexLevelOne="bicycle"/>
<indexFlag indexLevelOne="bicycle" indexLevelTwo="brake system"/>
</indexFlags>
```

4 Example

Given the following in a data module:

```
<levelledPara>
<title>General</title>
<para>
The s1kd-tools are a set of small tools for manipulating S1000D XML
data.
</para>
</levelledPara>
```

And the following `.indexflags` file:

```
<indexFlags>
<indexFlag indexLevelOne="S1000D"/>
<indexFlag indexLevelTwo="S10000D" indexLevelTwo="s1kd-tools"/>
```

```
<indexFlag indexLevelOne="data"/>
<indexFlag indexLevelOne="data" indexLevelTwo="XML"/>
</indexFlags>
```

Then the s1kd-index command:

```
$ s1kd-index <DM>.XML
```

Would result in the following:

```
<levelledPara>
<title>General</title>
<para>
The s1kd-tools<indexFlag indexLevelOne="S1000D"
indexLevelTwo="s1kd-tools"/> are a set of small tools for
manipulating S1000D<indexFlag indexLevelOne="S1000D"/>
XML<indexFlag indexLevelOne="data" indexLevelTwo="XML"/>
data<indexFlag indexLevelOne="data"/>.
</para>
</levelledPara>
```

s1kd-instance

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-instance** tool produces "instances" of S1000D CSDB objects, derived from "master" (or "source") objects. The tool supports multiple methods of instantiating objects:

– Filtering on user-supplied applicability definitions, so that non-applicable elements and (optionally) unused applicability annotations are removed in the instance. The definitions can be supplied directly or read from a PCT.

– Filtering on skill levels and security classifications to remove sensitive data.

– Using a CIR to produce a standalone instance from a CIR-dependent master.

Any combination of these methods can be used when producing an instance.

The applications for this tool include:

– Delivering customized data modules or publications to different customers.
– Creating customized instances of CSDB objects which are maintained within the CSDB.
– As a backend to filter content or resolve CIR dependencies at runtime in an electronic viewer application.

2 Usage

```
s1kd-instance [options] [<object>...]
```

3 Options

-A, --simplify	Simplify inline applicability annotations, and remove annotations which are unambiguously valid or invalid.
-a, --reduce	Remove applicability annotations which are unambiguously valid or invalid.
-C, --comment <comment>	Add an XML comment to an instance. Useful as another way of identifying an object as an instance aside from the source address or extended code, or giving additional information about a particular instance. By default, the comment is inserted at the top of the document, but this can be customized with the -X option.
-c, --code <code>	Specify a new data module code (DMC) or publication module code (PMC) for the instance.
-D, --dump <CIR>	Dumps the built-in XSLT used to resolve dependencies for <CIR> CIR type to stdout. This can be used as a starting point for a custom XSLT script to be specified with the -x option. The following types currently have built-in XSLT and can therefore be used as values for <CIR>: – accessPointRepository – applicRepository – cautionRepository – circuitBreakerRepository – controlIndicatorRepository – enterpriseRepository – functionalItemRepository – illustratedPartsCatalog

	– partRepository – supplyRepository – toolRepository – warningRepository – zoneRepository
-d, --dir <dir>	Directory to start searching for referenced objects in. By default, the current directory will be searched. This applies for the ACT and PCT data modules when a product is specified (-p) without specifying the PCT explicitly (-P), or when searching for source objects (-@).
-E, --no-extension	Remove the extension from an instance produced from an already extended object.
-e, --extension <ext>	Specify an extension on the data module code (DME) or publication module code (PME) for the instance.
-F, --flatten-alts	After filtering, "alts" elements containing only one child element will be "flattened" by replacing them with the applicable child element. Alts elements with multiple child elements are left untouched.
-f, --overwrite	Force overwriting of files. By itself, this will cause the source object(s) to be overwritten instead of being printed to stdout. When used with the -o or -O options, if a file exists with the same name as the one specified (-o) or automatically generated by the tool (-O), this will force it to be overwritten. Otherwise, a warning will be printed and the existing file will not be overwritten.
-G, --custom-orig <CODE>/<NAME>	Similar to the -g option, but instead of the default enterprise code and name, use the values <CODE> and <NAME>, which are separated by a slash (/). To only include a code, specify <CODE> with no slash. To only include a name, specify <NAME> prefixed by a slash.
-g, --set-orig	Set the originator of the instance. When this option is specified, the code "`S1KDI`" and the name "`s1kd-instance tool`" are used by default to identify that the instance was produced by this tool. A different code and name can be specified with the -G option.
-H, --list-properties <method>	Create an XML report of all the applicability properties used in the specified CSDB objects. <method> determines how to include values in the report: – "standalone" - Only include the values that are explicitly used in the object.

	– "all" - Include all values as defined in the ACT and CCT.
-h, -?, --help	Show help/usage message.
-I, --date <date>	Set the issue date of the instance. By default, the issue date is taken from the source. If - is given for <date>, the current date will be used.
-i, --infoname <infoName>	Give the data module instance a different infoName.
-J, --clean-display-text	Remove display text from annotations which are simplified in -A or -9 mode.
-j, --clean-ents	After filtering, remove external entities (such as ICNs) which are no longer used from the resulting instances.
-K, --skill-levels <levels>	Filter the object on the specified skill levels. Elements which are marked with skill levels not contained in the string <levels> are removed in the resulting instance.
-k, --skill <level>	Set the skill level of the instance.
-L, --list	Source is a list of object filenames to create instances of, rather than an object itself.
-l, --language <lang>	Set the language and country of the instance. For example, to create an instance for US English, lang would be "en-US".
-m, --remarks <remarks>	Set the remarks for the instance.
-N, --omit-issue	Omit issue/inwork numbers from automatically generated filenames.
-n, --issue <iss>	Set the issue and inwork numbers of the instance. By default, the issue and inwork number are taken from the source. When updating an instance (-@), if + is given for <iss>, the updated instance will have the same issue number with an inwork number incremented by one.
-O, --outdir <dir>	Output instance(s) in <dir>, automatically naming them based on: – the extension specified with -e – the code specified with -c – The issue info specified with -n – the language and country specified with -L If any of the above are not specified, the information is copied from the source object. If <dir> does not exist, it will be created. If a file exists with the same name in the specified directory, a warning will be display and the file will not be overwritten, unless the -f option is specified.

	When using this option, non-XML files, such as external publications, may be specified as objects. They will be copied to <dir>.
-o, --out <file>	Output instance to file instead of stdout.
-P, --pct <PCT>	PCT file to read product definitions from (-p). If a product is specified but no PCT is given, the tool will attempt to use the ACT reference of each source data module to find the ACT and PCT data modules in the current directory.
-p, --product <product>	The ID or primary key of a product in the specified PCT data module (-P), the PCT referenced by the ACT data module specified with -1, or the PCT data module referenced by the source data module itself. A primary key is given in the same form as the -s option and should match a unique assign of a product instance, e.g., "`serialno:prodattr=12345`". Multiple keys can be used by specifying this option multiple times.
-Q, --resolve-containers	Resolve references to container data modules, selecting the appropriate reference for the specified applicability. If zero or more than one references are applicable, the reference to the container will be left untouched. Additionally, if the object being filtered is itself a container data module, the applicability of the referenced data modules will be copied in to it as inline annotations prior to filtering.
-q, --quiet	Quiet mode. Errors are not printed.
-R, --cir <CIR> ...	Use a CIR to resolve external dependencies in the master object, making the instance object standalone. Additional CIRs can be used by specifying the -R option multiple times. The following CIRs have some built-in support: – Access points – Applicability – Cautions – Circuit breakers – Controls/indicators – Enterprises – Functional items – Illustrated parts data – Parts – Supplies – Tools

	– Warnings – Zones The methods of resolving the dependencies for a CIR can be changed by specifying a custom XSLT script with the -x option. The built-in XSLT used for the above CIR data modules can be dumped with the -D option. If "*" is given for <CIR>, the tool will search for CIR data modules automatically.
-r, --recursive	Search for referenced objects recursively. This applies for the ACT and PCT data modules when a product is specified (-p) without specifying the PCT explicitly (-P), when searching for source objects (-@), or when searching for CIR data modules (-R).
-S, --no-source-ident	Do not include <sourceDmIdent>/<sourcePmIdent>/ <repositorySourceDmIdent> in the instance.
-s, --assign <applic>	An applicability definition in the form of "`<ident>:<type>=<value>`". Any number of values can be defined by specifying this option multiple times.
-T, --tag	Tag non-applicable elements with the processing instruction <?notApplicable?> instead of removing them.
-t, --techname <techName>	Give the instance a different techName/pmTitle.
-U, --security-classes <classes>	Filter the object on the specified security classes. Elements marked with security classes not contained in the string <classes> are removed in the resulting instance.
-u, --security <sec>	Set the security classification of the instance. An instance may have a lower security classification than the source if classified information is removed for a particular customer.
-V, --infoname-variant <variant>	Give the instance a different info name variant.
-v, --verbose	Verbose output.
-W, --set-applic	Set the applicability for the whole object, overwriting the current applicability with the user-defined applicability values.
-w, --whole-objects	Check the applicability, skill level, and security classification of the whole object against the user-defined applicability, skill levels, and security classifications. If the whole object is not applicable, then no instance is created.
-X, --comment-xpath <path>	The XPath expression indicating where the comment specified with -C will be inserted. This should be the path to an element where the comment will be inserted as the first child node. By default, this is the top of the document.
-x, --xsl <XSL>	Use a custom XSLT script to resolve CIR dependencies. If this option follows -R, the specified XSLT script will only

	be used for the last specified CIR. If it precedes any -R, the specified XSLT script will be used for all CIRs that do not override it with a following -x.
-Y, --applic <text>	Update the applicability for the whole object using the user-defined applicability values, and using <text> as the new display text.
-y, --update-applic	Update the applicability for the whole object using the user-defined applicability values.
-Z, --add-required	Fix certain elements automatically after filtering. For example, if all support equipment is removed due to filtering, a `<noSupportEquips>` element will be inserted automatically.
-z, --issue-type <type>	Set the issue type of the instance.
-1, --act	Specify the ACT to use to find the CCT and/or PCT.
-2, --cct	Specify the CCT to read dependency tests from (-~).
-4, --flatten-alts-refs	Same as the -F option, but in addition to flattening alts elements, the `internalRefTargetType` of cross-references to them will be changed to the appropriate type (e.g., `"irtt01"` for a `<figure>` in a `<figureAlts>`). This is specifically useful for S1000D Issue 4.1, where the Default BREX does not allow the standard `internalRefTargetType` values to be used with the alts elements.
-5, --print	When -O is used, print the automatically generated file name of the instance.
-6, --clean-annotations	Remove unused applicability annotations.
-7, --dry-run	Do not actually create or update any instances, only print the file names of instances that would be created or updated.
-8, --reapply	Automatically reapply the applicability of the source object when filtering.
-9, --prune	Remove only invalid parts of applicability annotations.
-@, --update-instances	Rather than source objects, the objects specified are existing instances that will be updated.
-%, --read-only	Make instance objects read-only.
-!, --no-infoname	Do not include an infoName in the instance.
-~, --dependencies	Add dependency tests from the CCT to assertions that use the dependant values.
--version	Show version information.
<object>...	Source CSDB objects to instantiate.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.

--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 Identifying the source of an instance

The resulting data module instances will contain the element `<sourceDmIdent>`, which will contain the identification elements of the source data modules used to instantiate them. Publication module instances will contain the element `<sourcePmIdent>` instead.

Additionally, the data module instance will contain an element `<repositorySourceDmIdent>` for each CIR specified with the -R option.

If the -S option is used, neither the `<sourceDmIdent>`/`<sourcePmIdent>` elements or `<repositorySourceDmIdent>` elements are added. This can be useful when this tool is not used to make an "instance" per se, but more generally to make a module based on an existing module.

3.2 Removing/simplifying/pruning applicability annotations

By default, filtering on applicability will remove invalid elements from the resulting instance. In some cases, though, it may be desirable to remove redundant applicability annotations on valid elements. The -a (--reduce), -A (--simplify) and -9 (--prune) options provide different methods of doing this.

The -a (--reduce) option will remove applicability annotations (`applicRefId`) from elements which are deemed to be unambiguously valid or invalid (their validity does not rely on applicability values left undefined by the user). The unused occurrences of the corresponding `<applic>` elements are removed as well.

The -A (--simplify) option will do the same as the -a option, but will also attempt to simplify unused parts of applicability annotations. It simplifies an annotation by removing `<assert>` elements determined to be either unambiguously valid or invalid given the user-defined values, and removing unneeded `<evaluate>` elements when they contain only one remaining `<assert>`.

The -9 (--prune) option works similarly to the -A option, except that only invalid parts of applicability annotations are removed.

For example, given the following input:

```
<referencedApplicGroup>
<applic id="app-0001">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
<applic id="app-0002">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
```

```
applicPropertyValues="B"/>
</applic>
<applic id="app-0003">
<evaluate andOr="or">
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="normal"/>
</evaluate>
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="icy"/>
</evaluate>
</evaluate>
</applic>
</referencedApplicGroup>
<!-- snip -->
<para applicRefId="app-0001">This applies to version A.</para>
<para applicRefId="app-0002">This applies to version B.</para>
<para applicRefId="app-0003">
This applies to version A if the weather is normal, or version B if
the weather is icy.
</para>
```

If this data is filtered for version A, without specifying a value for the weather, and the -a, -A or -9 options are not used, the following will be the result:

```
<referencedApplicGroup>
<applic id="app-0001">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
<applic id="app-0002">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
```

```
applicPropertyValues="B"/>
</applic>
<applic id="app-0003">
<evaluate andOr="or">
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="normal"/>
</evaluate>
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="icy"/>
</evaluate>
</evaluate>
</applic>
</referencedApplicGroup>
<!-- snip -->
<para applicRefId="app-0001">This applies to version A.</para>
<para applicRefId="app-0003">
This applies to version A if the weather is normal, or version B if
the weather is icy.
</para>
```

The second paragraph is removed, because it only applies to version B.

If the -a option is used, the following would be the result:

```
<referencedApplicGroup>
<applic id="app-0003">
<evaluate andOr="or">
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
```

```
applicPropertyValues="normal"/>
</evaluate>
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="icy"/>
</evaluate>
</evaluate>
</applic>
</referencedApplicGroup>
<!-- snip -->
<para>This applies to version A.</para>
<para applicRefId="app-0003">
This applies to version A if the weather is normal, or version B if
the weather is icy.
</para>
```

The applicability annotation reference for the first paragraph is removed because, given that the version is A, it must be true. The corresponding applicability annotations, which are no longer referenced, are also removed. The applicability on the third paragraph remains, however, because it is only true if the version is A **and** the weather is normal, and no value has been given for the weather.

If the -A option is used, the following would be the result:

```
<referencedApplicGroup>
<applic id="app-0003">
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="normal"/>
</applic>
</referencedApplicGroup>
<!-- snip -->
<para>This applies to version A.</para>
<para applicRefId="app-0003">
This applies to version A if the weather is normal, or version B if
the weather is icy.
</para>
```

The annotation is now simplified to remove resolved assertions. Because the version must be A, any assertions restating this can be removed as redundant, and any portions of the annotation in which the version is **not** A can be removed as invalid. This leaves only the assertion about the weather.

If the -9 option is used, the following would be the result:

```
<referencedApplicGroup>
<applic id="app-0001">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
<applic id="app-0003">
<evaluate andOr="and">
<assert
applicPropertyIdent="version"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert
applicPropertyIdent="weather"
applicPropertyType="condition"
applicPropertyValues="normal"/>
</evaluate>
</applic>
</referencedApplicGroup>
<!-- snip -->
<para applicRefId="app-0001">This applies to version A.</para>
<para applicRefId="app-0003">
This applies to version A if the weather is normal, or version B if
the weather is icy.
</para>
```

The first annotation is kept because it is entirely valid. The third annotation is simplified by removing the invalid assertions, but the valid assertions are preserved.

Note

The -A and -9 options may change the **meaning** of certain applicability annotations without changing the **display text**. Display text is always left untouched, so using this option may cause display text to be technically incorrect.

These options are best used when display text will be automatically generated after filtering, such as with the s1kd-aspp tool. The -J option of this tool can be combined with the -k option of the s1kd-aspp tool to only generate display text for annotations which are modified.

3.3 Applicability of an instance (-W, -Y, -y)

The applicability of an instance may change as a result of filtering. For example, a source data module which is applicable to two versions of a product may produce two instances which are each only applicable to one version. There are three options which control how the applicability of the whole instance object is updated.

The -W option will create an applicability annotation for the instance using only the user-defined applicability values. This means, for example, that given the following command:

```
$ s1kd-instance -s version:prodattr=A -W ...
```

The instance would contain the following annotation:

```
<dmStatus>
<!-- snip -->
<applic>
<assert applicPropertyIdent="version"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</applic>
<!-- snip -->
</dmStatus>
```

regardless of what the applicability of the source object was.

The -y option will create an applicability annotation for the instance by combining the user-defined applicability with the applicability of the source object. For example, given the following annotation in the source object:

```
<dmStatus>
<!-- snip -->
<applic>
<assert applicPropertyIdent="version"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</applic>
<!-- snip -->
</dmStatus>
```

and the following command:

```
$ s1kd-instance -s weather:condition=icy -y ...
```

The annotation for the instance would be as follows:

```
<dmStatus>
<!-- snip -->
<applic>
<evaluate andOr="and">
<assert applicPropertyIdent="version"
applicPropertyType="prodattr" applicPropertyValues="A"/>
<assert applicPropertyIdent="weather"
applicPropertyType="condition" applicPropertyValues="icy"/>
</evaluate>
</applic>
<!-- snip -->
</dmStatus>
```

The -Y option by itself works the same as the -y option, but allows custom display text to be set for the annotation. It can also be combined with the -W option to add custom display text to the overwriting annotation:

```
$ s1kd-instance -s version:prodattr=A -WY "Version A" ...
```

```
<dmStatus>
<!-- snip -->
<applic>
<displayText>
<simplePara>Version A</simplePara>
</displayText>
<assert applicPropertyIdent="version"
applicPropertyType="prodattr" applicPropertyValues="A"/>
</applic>
<!-- snip -->
</dmStatus>
```

3.4 Filtering for multiple values of a single property

Though not usually the case, it is possible to create an instance which is filtered on multiple values of the same applicabilty property. Given the following:

```
<referencedApplicGroup>
<applic id="apA">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
<applic id="apB">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="B"/>
</applic>
<applic id="apC">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="C"/>
</applic>
</referencedApplicGroup>
<!-- ... -->
<para applicRefId="apA">Applies to A</para>
<para applicRefId="apB">Applies to B</para>
<para applicRefId="apC">Applies to C</para>
```

filtering can be applied such that the instance will be applicable to both A and C, but not B. This is done by specifying a property multiple times in the applicability definition arguments. For example:

```
$ s1kd-instance -A -Y "A or C" -s attr:prodattr=A -s attr:prodattr=C ...
```

This would produce the following in the instance:

```
<dmStatus>
<!-- ... -->
<applic>
<displayText>
<simplePara>A or C</simplePara>
</displayText>
<evaluate andOr="or">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="C"/>
</evaluate>
</applic>
<!-- ... ->
</dmStatus>
<!-- ... -->
<referencedApplicGroup>
<applic id="apA">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="A"/>
</applic>
<applic id="apC">
<assert applicPropertyIdent="attr"
applicPropertyType="prodattr"
applicPropertyValues="C"/>
</applic>
</referencedApplicGroup>
<!-- ... -->
<para applicRefId="apA">Applies to A</para>
<para applicRefId="apC">Applies to C</para>
```

3.5 Resolving CIR dependencies with a custom XSLT script (-x)

A CIR contains more information about an item than can be captured in a data module's reference to it. If this additional information is required, there are two methods to include it:

– Distribute the CIR with the data module so the extra information can be linked to
– "Flatten" the information to fit in the data module's schema.

A custom XSLT script can be supplied with the -x option, which is then used to resolve the CIR dependencies of the last CIR specified with -R. For example:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
```

```
<xsl:template match="functionalItemRef">
<xsl:variable name="fin" select"@functionalItemNumber"/>
<xsl:variable name="spec" select="//functionalItemSpec[
functionalItemIdent/@functionalItemNumber = $fin]"/>
<xsl:value-of select="$spec/name"/>
</xsl:template>
</xsl:stylesheet>
```

This script would resolve a `functionalItemRef` by "flattening" it to the value of the `name` element obtained from the CIR.

The example CIR would contain a specification like:

```
<functionalItemSpec>
<functionalItemIdent functionalItemNumber="ABC"
functionalItemType="fit01"/>
<name>Hydraulic pump</name>
<functionalItemAlts>
<functionalItem/>
</functionalItemAlts>
</functionalItemSpec>
```

The source data module would contain a reference:

```
<para>
The
<functionalItemRef functionalItemNumber="ABC"/>
is an item in the system.
</para>
```

The command would resemble:

```
$ s1kd-instance -R <CIR> -x <custom XSLT> <src>
```

And the resulting XML would be:

```
<para>The Hydraulic pump is an item in the system.</para>
```

The source data module and CIR are combined in to a single XML document which is used as the input to the XSLT script. The root element `mux` contains two `dmodule` elements. The first is the source data module, and the second is the CIR data module specified with the corresponding -R option. The CIR data module is first filtered on the defined applicability.

An "identity" template is automatically inserted in to the custom XSLT script, equivalent to the following:

```
<xsl:template match="@*|node()">
<xsl:copy>
<xsl:apply-templates select="@*|node()"/>
</xsl:copy>
</xsl:template>
```

This means any elements or attributes which are not matched with a more specific template in the custom XSLT script are automatically copied.

The set of built-in XSLT scripts used to resolve dependencies can be dumped using the -D option.

3.6 Updating instances (-@)

The -@ option is used to automatically update instance objects from their source objects.

The tool will use the `<sourceDmIdent>`/`<sourcePmIdent>` in each instance to find the source object they were derived from, and filter it based on the instance's metadata in order to produce an updated version of the instance. CIRs identified by `<repositorySourceDmIdent>` elements in the instance will also be used to update it.

Only objects which identify a source object will be processed in this mode. All other non-instance objects specified are ignored. The elements `<sourceDmIdent>`, `<sourcePmIdent>` and `<repositorySourceDmIdent>` identify a specific issue of an object that the instance was last updated from, but this is ignored and the latest issue found of a source object will be used instead.

This feature is primarily useful when instances of objects are stored in the CSDB, rather than only being generated during publication or dynamically in a viewer. For example, imagine you have a descriptive data module:

```
DMC-EX-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
```

and you deliver to two customers, C1 and C2. The data module contains information for both:

```
<description>
<para>This text applies to all customers.</para>
<para applicRefId="app-C1">This only applies to Customer 1.</para>
<para applicRefId="app-C2">This only applies to Customer 2.</para>
</description>
```

Neither customer wants to see information that applies only to the other, so you can create two customized instances of this data module, identified with the extended code:

```
DMC-EX-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
DME-12345-C1-EX-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
DME-12345-C2-EX-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
```

Each instance data module identifies the original data module as its source:

```
<sourceDmIdent>
<dmCode modelIdentCode="EX" systemDiffCode="A" systemCode="00"
subSystemCode="0" subSubSystemCode="0" assyCode="00" disassyCode="00"
disassyCodeVariant="A" infoCode="040" infoCodeVariant="A"
itemLocationCode="D"/>
<language languageIsoCode="en" countryIsoCode="CA"/>
<issueInfo issueNumber="001" inWork="00"/>
</sourceDmIdent>
```

and is set to apply only to the correct customer:

```
<dmStatus>
...
<applic>
<assert applicPropertyIdent="customer" applicPropertyType="prodattr"
applicPropertyValues="1"/>
</applic>
...
</dmStatus>
```

Note

The assertions in the applicability of an instance must use single values in order to work in this mode. Ranges (~) and sets (|) are not supported.

Now, when a change is made to the master data module, this tool can be used to update these instances automatically:

```
$ s1kd-instance -@ -f DME-*.XML
```

4 Exit status

0	No errors.
1	Missing or incomplete argument.
2	Specified file does not exist.
3	Source object for an instance could not be found.
4	Malformed applicability definition.
6	XML was invalid or does not conform to S1000D.
7	Value given for an argument was malformed.
8	Issue date specified with -I is invalid.
9	The number of CIR data modules found when searching exceeded the available memory.

5 Examples

Filtering a data module on specified applicability and writing to stdout:

```
$ s1kd-instance -s version:prodattr=A <DM>
```

Filtering a data module on a specified product instance and writing to stdout:

```
$ s1kd-instance -P <PCT> -p versionA <DM>
```

Filtering a data module on specified skill levels and writing to stdout:

```
$ s1kd-instance -k sk01/sk02 <DMs>
```

Filtering data modules for a particular customer and outputting with extended identification:

```
$ s1kd-instance -s version:prodattr=A -e 12345-54321 -O . <DMs>
```

Writing out a data module from stdin to a directory with automatic naming:

```
$ s1kd-transform -s <xsl> <DM> | s1kd-instance -SO <dir>
```

s1kd-neutralize

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

Generates neutral metadata for the specified CSDB objects. This includes:

– XLink attributes for references, using the S1000D URN scheme.

– RDF and Dublin Core metadata.

2 Usage

```
s1kd-neutralize [-o <file>] [-Dflnvh?] [<object>...]
```

3 Options

-D, --delete	Remove neutral metadata from the CSDB object.
-f, --overwrite	Overwrite specified CSDB object(s) automatically.
-h, -?, --help	Show usage message.

-l, --list	Treat input (stdin or arguments) as lists of CSDB objects to neutralize, rather than CSDB objects themselves.
-n, --namespace	Include the IETP namespaces for data module and publication module elements.
-o, --out <file>	Output neutralized CSDB object XML to <file> instead of stdout.
-v, --verbose	Verbose output.
--version	Show version information.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Example

```
$ DMOD=DMC-XLINKTEST-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
$ xmllint --xpath "//description/dmRef" $DMOD
<dmRef>
<dmRefIdent>
<dmCode modelIdentCode="XLINKTEST" systemDiffCode="A"
systemCode="00" subSystemCode="0" subSubSystemCode="0" assyCode="01"
disassyCode="00" disassyCodeVariant="A" infoCode="040"
infoCodeVariant="A" itemLocationCode="D"/>
</dmRefIdent>
<dmRefAddressItems>
<dmTitle>
<techName>XLink test</techName>
<infoName>Referenced data module</infoName>
</dmTitle>
</dmRefAddressItems>
</dmRef>

$ s1kd-neutralize $DMOD | xmllint --xpath "//description/dmRef" -
<dmRef xlink:type="simple"
xlink:href="URN:S1000D:DMC-XLINKTEST-A-00-00-01-00A-040A-D"
xlink:title="XLink test - Referenced data module">
[...]
</dmRef>
```

s1kd-syncrefs

Description

Table of contents

Page

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-syncrefs** tool copies all external references (dmRef, pmRef, externalPubRef) within the content of a data module and uses them to generate the <refs> element. Each unique reference is copied, sorted, and placed in to the <refs> element. If a <refs> element already exists, it is overwritten.

2 Usage

```
s1kd-syncrefs [-dflvh?] [-o <out>] [<data module>...]
```

3 Options

-d, --delete	Delete the <refs> element.
-f, --overwrite	Overwrite the data modules automatically.

-h, -?, --help	Show help/usage message.
-l, --list	Treat input (stdin or arguments) as lists of data modules to synchronize references in, rather than data modules themselves.
-o, --out <out>	The resulting XML is written to <out> instead of stdout.
-v, --verbose	Verbose output.
--version	Show version information.
<data module>...	The data module(s) to synchronize references in. Default is to read from stdin.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

4 Exit status

0	No errors.
1	Invalid data module.
2	Number of references in a data module exceeded the available memory.

5 Example

```
$ s1kd-syncrefs -f DMC-EX-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
```

s1kd-uom

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1 General

The **s1kd-uom** tool converts between specified units of measure in quantity data, for example, to automatically localize units of measure in data modules.

2 Usage

```
s1kd-uom [-dflv,.h?] [-D <fmt>] [-F <fmt>]
         [-u <uom> -t <uom> [-e <expr>] [-F <fmt>] ...]
         [-s <name>|-S <path> ...] [-U <path>] [-p <fmt> [-P <path>]]
         [<object>...]
```

3 Options

-D, --duplicate-format <fmt>	Specify a custom format for duplicating quantities (-d). The '%' character acts as a placeholder for the duplicate quantity value. The default format for -d is equivalent to `-D ' (%)'`.
-d, --duplicate	When converting, instead of overwriting the original quantity, include the converted quantity after the original in parenthesis. For example, "200 mm" when converting mm to in would become "200 mm (7.87 in)".
-e, --formula <expr>	Specify the formula for a conversion, given as an XPath expression.
-F, --format <fmt>	Specify the format for quantity values. When used before -u, this specifies the default format for all conversions. Otherwise, this specifies the format for each individual conversion. Formats specified for individual conversions override the default format set for all conversions.
-f, --overwrite	Overwrite input CSDB objects.
-h, -?, --help	Show help/usage message.
-l, --list	Treat input (stdin or arguments) as lists of filenames of CSDB objects to list references in, rather than CSDB objects themselves.
-P, --uomdisplay <path>	Use a custom `.uomdisplay` file.
-p, --preformat <fmt>	Preformat quantity data to the specified decimal format. The built-in formats are: – SI - comma for decimal separator, space for grouping – euro - comma for decimal separator, full-stop for grouping – imperial - full-stop for decimal separator, comma for grouping
-S, --set <path>	Apply a set of conversions defined in an XML file.

-s, --preset <name>	Apply a set of predefined conversions. The available presets are: – SI - convert imperial/US customary units to SI units. – imperial - convert SI units to British imperial units. – US - convert SI units to US customary units.
-t, --to <uom>	Unit of measure to convert to.
-U, --uom <path>	Use a custom `.uom` file.
-u, --from <uom>	Unit of measure to convert from.
-v, --verbose	Verbose output.
-,, --dump-uom	Dump the default `.uom` file.
-., --dump-uomdisplay	Dump the default `.uomdisplay` file.
--version	Show version information.
<object>	CSDB objects to convert quantities in.

In addition, the following options enable features of the XML parser that are disabled as a precaution by default:

--dtdload	Load the external DTD.
--net	Allow network access to load external DTD and entities.
--noent	Resolve entities.
--xinclude	Do XInclude processing.

3.1 `.uom` file

This file contains the rules for converting units of measure. If no specific conversions are given with the -u and -t options, this file also acts as a list of all conversions to perform.

By default, the program will search the current directory and parent directories for a file named `.uom`, but any file can be specified by using the -U option.

Example of a `.uom` file:

```
<uom>
<convert from="degF" to="degC" formula="($value - 32) * (5 div 9)"/>
<convert from="in" to="cm" formula="$value * 2.54"/>
<convert from="lbm" to="kg" formula="$value div 2.205"/>
</uom>
```

The tool contains a default set of rules for common units of measure. This can be used to create a default `.uom` file by use of the -, option:

```
$ s1kd-uom -, > .uom
```

To select only certain common rules when generating a `.uom` file, the -u and -t options can be used:

```
$ s1kd-uom -, -u in -t cm -u degF -t degC > .uom
```

This will generate a `.uom` file containing rules to convert inches to centimetres, and degrees Fahrenheit to degrees Celsius.

The same file format is used with the -S option to specify a set of conversions to perform. In this case, the attribute `formula` is optional, as the default formula or the formula in the `.uom` file will be used if it is not specified.

3.2 Conversion formula variables (-e)

When specifying a formula for conversion, the following variables can be used:

`$pi`	The constant π
`$value`	The original quantity value

For example, the formula to convert degrees to radians can be given as follows:

```
$value * ($pi div 180)
```

3.3 Preformatting UOMs (-p) and the `.uomdisplay` file

The tool can also convert semantic quantity data to presentation quantity data. The -p option specifies which conventions to use for formatting quantity values. For example:

```
<para>Tighten the
<quantity>
<quantityGroup>
<quantityValue quantityUnitOfMeasure="cm">6.35</quantityValue>
</quantityGroup>
</quantity>
bolt.</para>

$ s1kd-uom -p SI <DM>

<para>Tighten the 6,35 cm bolt.</para>
```

This can also be combined with UOM conversions:

```
$ s1kd-uom -u cm -t in -p imperial <DM>

<para>Tighten the 2.5 in bolt.</para>
```

Custom formats for values or UOMs can be defined in the `.uomdisplay` file. By default, the tool will search the current directory and parent directories for a file named `.uomdisplay`, but any file can be specified by using the -P option.

Example of a `.uomdisplay` file:

```
<uomDisplay>
<format name="custom" decimalSeparator="," groupingSeparator="."/>
<uoms>
<uom name="cm"> cm</uom>
<uom name="cm2"> cm<superScript>2</superScript></uom>
</uoms>
```

```
<currencies>
<currency name="CAD">
<prefix>$</prefix>
<postfix> CAD</postfix>
</currency>
<currency name="GBP">
<prefix>£</prefix>
<postfix> GBP</postfix>
</currency>
</currencies>
</uomDisplay>
```

Units of measure and currencies that are not defined will be presented as their name (e.g., "cm2") separated from the value by a space.

More complex UOM display, such as pluralization of units of measure, can be accomplished with embedded XSLT in the `.uomdisplay` file:

```
<uoms
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:variable name="value" select="parent::*/>
<uom name="in">
<xsl:text> </xsl:text>
<xsl:choose>
<xsl:when test="$value = 1">inch</xsl:when>
<xsl:otherwise>inches</xsl:otherwise>
</xsl:choose>
</uom>
<uom name="ft">
<xsl:text> </xsl:text>
<xsl:choose>
<xsl:when test="$value = 1">foot</xsl:when>
<xsl:otherwise>feet</xsl:otherwise>
</xsl:choose>
</uom>
</uoms>
```

The context for the embedded XSLT is the unit of measure attribute on the value, tolerance or group. XSLT elements in the `<uoms>` element will be processed for all units of measure, while XSLT elements in `<uom>` elements will only apply to an individual unit of measure.

The tool contains a default set of formats and displays. These can be used to create a default `.uomdisplay` file by use of the -. option:

```
$ s1kd-uom -. > .uomdisplay
```

4 Examples

4.1 Common units of measure

Input:

```
<quantity>
<quantityGroup>
<quantityValue quantityUnitOfMeasure="cm">15</quantityValue>
</quantityGroup>
</quantity>
```

Command:

```
$ s1kd-uom -u cm -t in <DM>
```

Output:

```
<quantity>
<quantityGroup>
<quantityValue quantityUnitOfMeasure="in">5.91</quantityValue>
</quantityGroup>
</quantity>
```

4.2 Using a custom formula and format

Input:

```
<quantity
quantityType="qty02"
quantityTypeSpecifics="CAD">10.00</quantity>
```

Command:

```
$ s1kd-uom -u CAD -t USD -e '$value div 1.31' -F '0.00'
```

Output:

```
<quantity
quantityType="qty02"
quantityTypeSpecifics="USD">7.36</quantity>
```

5 UOM file schema

5.1 UOM

Markup element: `<uom>`

Attributes:

– `format` (O), the number format for all rules.

Child elements:

– `<convert>`

5.2 Conversion rule

The element `<convert>` defines a rule to convert one unit of measure to another.

Markup element: `<convert>`

Attributes:

- `format` (O), the number format for this specific rule.
- `formula` (M), the expression used to convert the quantity value.
- `from` (M), unit of measure to convert from.
- `to` (M), unit of measure to convert to.

Child elements:

- None

6 UOM display file schema

6.1 UOM display

Markup element: `<uomDisplay>`

Attributes:

- None

Child elements:

- `<format>`
- `<groupTypePrefixes>`
- `<wrapInto>`
- `<uoms>`
- `<currencies>`

6.2 Quantity value format

Markup element: `<format>`

Attributes:

- `name` (M), the name of the format
- `decimalSeparator` (M), the decimal separator
- `groupingSeparator` (M), the grouping separator

Child elements:

- None

6.3 Group type prefixes

The element `<groupTypePrefixes>` specifies prefixes which are added for specific group types.

Markup element: `<groupTypePrefixes>`

Attributes:

– None

Child elements:

– `<nominal>`, text placed before a nominal group.
– `<minimum>`, text placed before a minimum group.
– `<minimumRange>`, text placed before a minimum group that is followed by a maximum group to specify a range.
– `<maximum>`, text placed before a maximum group.
– `<maximumRange>`, text placed before a maximum group that is preceded by a minimum group to specify a range.

6.4 Wrap into element

Markup element: `<wrapInto>`

Attributes:

– None

Child elements:

The element `<wrapInto>` contains one child element of any type, which quantities will be wrapped in to after formatting.

6.5 Units of measure

Markup element: `<uoms>`

Attributes:

– None

Child elements:

– `<uom>`

The element `<uoms>` may also contain arbitrary XSLT elements which will be processed for all units of measure.

6.6 Display of a unit of measure

Markup element: `<uom>`

Attributes:

– `name` (M), the name of the UOM.

Child elements:

The element `<uom>` may contain mixed content, which will be used for the display of the unit of measure. This can include XSLT elements, which allows for handling complex cases of UOM display, such as pluralization.

6.7 Currencies

Markup element: `<currencies>`

Attributes:

– None

Child elements:

– `<currency>`

The element `<currencies>` may also contain arbitrary XSLT elements which will be processed for all currencies.

6.8 Display of a currency

Markup element: `<currency>`

Attributes:

– `name` (M), the name of the currency.

Child elements:

– `<prefix>`, text placed before the currency value.

– `<postfix>`, text placed after the currency value.

The child elements of `<currency>` may contain mixed content, which will be used for the display of the unit of measure. This can include XSLT elements, which allows for handling complex cases of currency display.