S1000D tools

Documentation

S1000DTOOLS-KHZAE-00000-00

Issue No. 005, 2018-08-31

Publisher:
khzae.net

s1kd-tools

List of effective data modules

The listed documents are included in issue 005, dated 2018-08-31, 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	S1000DTOOLS-A-00-00-00-00A-001A-D	C	2018-08-31	1	All
List of effective data modules	S1000DTOOLS-A-00-00-00-00A-00SA-D	C	2018-08-31	3	All
Highlights	S1000DTOOLS-A-00-00-00-00A-00UA-D	C	2018-08-31	1	All
Table of contents	S1000DTOOLS-A-00-00-00-00A-009A-D	C	2018-08-31	3	All
List of abbreviations	S1000DTOOLS-A-00-00-00-00A-005A-D	C	2018-08-31	1	All
s1kd-tools - Description	S1000DTOOLS-A-00-00-00-00A-040A-D	N	2017-08-14	1	All
s1kd-tools - Introduction	S1000DTOOLS-A-00-00-00-00A-018A-D	C	2018-08-31	3	All
s1kd-tools - Usage examples	S1000DTOOLS-A-00-00-00-00A-130A-D	C	2018-08-31	10	All
s1kd-defaults - Description	S1000DTOOLS-A-00-0U-00-00A-040A-D	C	2018-08-17	3	All
s1kd-dmrl - Description	S1000DTOOLS-A-00-0M-00-00A-040A-D	C	2018-08-31	2	All
s1kd-newcom - Description	S1000DTOOLS-A-00-0G-00-00A-040A-D	C	2018-08-31	2	All
s1kd-newddn - Description	S1000DTOOLS-A-00-0H-00-00A-040A-D	C	2018-08-31	3	All
s1kd-newdm - Description	S1000DTOOLS-A-00-07-00-00A-040A-D	C	2018-08-31	6	All
s1kd-newdml - Description	S1000DTOOLS-A-00-0L-00-00A-040A-D	C	2018-08-31	3	All

Document title	Data module code		Issue date	No. of pages	Applicable to
s1kd-newimf - Description	S1000DTOOLS-A-00-0D-00-00A-040A-D	C	2018-08-31	2	All
s1kd-newpm - Description	S1000DTOOLS-A-00-0C-00-00A-040A-D	C	2018-08-31	3	All
s1kd-newsmc - Description	S1000DTOOLS-A-00-0Z-00-00A-040A-D	N	2018-08-31	3	All
s1kd-newupf - Description	S1000DTOOLS-A-00-0V-00-00A-040A-D	C	2018-08-31	2	All
s1kd-addicn - Description	S1000DTOOLS-A-00-0R-00-00A-040A-D	C	2018-08-31	2	All
s1kd-ls - Description	S1000DTOOLS-A-00-06-00-00A-040A-D	C	2018-08-31	2	All
s1kd-ref - Description	S1000DTOOLS-A-00-08-00-00A-040A-D	C	2018-08-31	4	All
s1kd-metadata - Description	S1000DTOOLS-A-00-09-00-00A-040A-D	C	2018-08-31	3	All
s1kd-sns - Description	S1000DTOOLS-A-00-0Y-00-00A-040A-D	N	2018-06-15	2	All
s1kd-transform - Description	S1000DTOOLS-A-00-0F-00-00A-040A-D	C	2018-08-31	2	All
s1kd-upissue - Description	S1000DTOOLS-A-00-05-00-00A-040A-D	C	2018-08-31	3	All
s1kd-brexcheck - Description	S1000DTOOLS-A-00-04-00-00A-040A-D	C	2018-08-31	5	All
s1kd-checkrefs - Description	S1000DTOOLS-A-00-0J-00-00A-040A-D	C	2018-08-17	3	All
s1kd-refls - Description	S1000DTOOLS-A-00-0P-00-00A-040A-D	C	2018-08-31	2	All
s1kd-validate - Description	S1000DTOOLS-A-00-02-00-00A-040A-D	C	2018-08-31	3	All
s1kd-acronyms - Description	S1000DTOOLS-A-00-0K-00-00A-040A-D	C	2018-08-17	3	All
s1kd-aspp - Description	S1000DTOOLS-A-00-0Q-00-00A-040A-D	C	2018-08-31	6	All

s1kd-tools

Document title	Data module code	Issue date	No. of pages	Applicable to
s1kd-flatten - Description	S1000DTOOLS-A-00-0N-00-00A-040A-D	C 2018-08-31	2	All
s1kd-fmgen - Description	S1000DTOOLS-A-00-0X-00-00A-040A-D	C 2018-08-31	3	All
s1kd-icncatalog - Description	S1000DTOOLS-A-00-0W-00-00A-040A-D	C 2018-08-31	5	All
s1kd-index - Description	S1000DTOOLS-A-00-0S-00-00A-040A-D	C 2018-08-17	3	All
s1kd-instance - Description	S1000DTOOLS-A-00-03-00-00A-040A-D	C 2018-08-31	14	All
s1kd-neutralize - Description	S1000DTOOLS-A-00-0E-00-00A-040A-D	C 2018-08-31	2	All
s1kd-syncrefs - Description	S1000DTOOLS-A-00-01-00-00A-040A-D	C 2018-08-31	2	All

s1kd-tools

Highlights

The listed changes are introduced in issue 005, dated 2018-08-31, of this publication.

Data module code	Reason for update
S1000DTOOLS-A-00-05-00-00A-040A-D	Add -c, -t and -H options.
S1000DTOOLS-A-00-06-00-00A-040A-D	Add -N option.
S1000DTOOLS-A-00-07-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0C-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0D-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0G-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0H-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0L-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0M-00-00A-040A-D	Add -% option.
S1000DTOOLS-A-00-0V-00-00A-040A-D	Add -~ option.
S1000DTOOLS-A-00-0Z-00-00A-040A-D	New tool for creating SCORM content packages.

s1kd-tools

Table of contents

The listed documents are included in issue 005, dated 2018-08-31, of this publication.

Document title	Document identifier	Issue date	No. of pages	Applicable to
s1kd-newdml - Description	S1000DTOOLS-A-00-0L-00-00A-040A-D	2018-08-31	3	All
s1kd-newimf - Description	S1000DTOOLS-A-00-0D-00-00A-040A-D	2018-08-31	2	All
s1kd-newpm - Description	S1000DTOOLS-A-00-0C-00-00A-040A-D	2018-08-31	3	All
s1kd-newsmc - Description	S1000DTOOLS-A-00-0Z-00-00A-040A-D	2018-08-31	3	All
s1kd-newupf - Description	S1000DTOOLS-A-00-0V-00-00A-040A-D	2018-08-31	2	All
Tools for authoring and managing data				
s1kd-addicn - Description	S1000DTOOLS-A-00-0R-00-00A-040A-D	2018-08-31	2	All
s1kd-ls - Description	S1000DTOOLS-A-00-06-00-00A-040A-D	2018-08-31	2	All
s1kd-ref - Description	S1000DTOOLS-A-00-08-00-00A-040A-D	2018-08-31	4	All
s1kd-metadata - Description	S1000DTOOLS-A-00-09-00-00A-040A-D	2018-08-31	3	All
s1kd-sns - Description	S1000DTOOLS-A-00-0Y-00-00A-040A-D	2018-06-15	2	All
s1kd-transform - Description	S1000DTOOLS-A-00-0F-00-00A-040A-D	2018-08-31	2	All
s1kd-upissue - Description	S1000DTOOLS-A-00-05-00-00A-040A-D	2018-08-31	3	All
Tools for validating data				
s1kd-brexcheck - Description	S1000DTOOLS-A-00-04-00-00A-040A-D	2018-08-31	5	All
s1kd-checkrefs - Description	S1000DTOOLS-A-00-0J-00-00A-040A-D	2018-08-17	3	All
s1kd-refls - Description	S1000DTOOLS-A-00-0P-00-00A-040A-D	2018-08-31	2	All
s1kd-validate - Description	S1000DTOOLS-A-00-02-00-00A-040A-D	2018-08-31	3	All

Document title	Document identifier	Issue date	No. of pages	Applicable to
Tools for delivering data				
s1kd-acronyms - Description	S1000DTOOLS-A-00-0K-00-00A-040A-D	2018-08-17	3	All
s1kd-aspp - Description	S1000DTOOLS-A-00-0Q-00-00A-040A-D	2018-08-31	6	All
s1kd-flatten - Description	S1000DTOOLS-A-00-0N-00-00A-040A-D	2018-08-31	2	All
s1kd-fmgen - Description	S1000DTOOLS-A-00-0X-00-00A-040A-D	2018-08-31	3	All
s1kd-icncatalog - Description	S1000DTOOLS-A-00-0W-00-00A-040A-D	2018-08-31	5	All
s1kd-index - Description	S1000DTOOLS-A-00-0S-00-00A-040A-D	2018-08-17	3	All
s1kd-instance - Description	S1000DTOOLS-A-00-03-00-00A-040A-D	2018-08-31	14	All
s1kd-neutralize - Description	S1000DTOOLS-A-00-0E-00-00A-040A-D	2018-08-31	2	All
s1kd-syncrefs - Description	S1000DTOOLS-A-00-01-00-00A-040A-D	2018-08-31	2	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

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. Together, they form a "suite" of tools for producing and maintaining S1000D publications within a CSDB.

They may also be used individually, to support other S1000D "suites" (usually from an S1000D "vendor") 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-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-validate** tool validates CSDB objects according to their S1000D schema and general correctness as XML documents.

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

The **s1kd-checkrefs** tool validates references between CSDB objects.

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.

s1kd-tools

Usage examples

Table of contents

List of tables

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.

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.

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, giving the data module code, title, security classification and responsible entity:

```
<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
```

s1kd-tools

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

When a data module is in the inwork state, the s1kd-upissue tool is called without any additional arguments:

```
$ 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

–   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   Quality assurance verification

In contrast to the first three 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 and 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 -is rinstate-status ...
```

The data module is once again upissued to the next official issue, and the issue type is set to one of the "rinstate-..." 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, create a directory for the customized delivery and use the -O option of the s1kd-instance tool:

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

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

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, since these systems typically track changes based on filenames.

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 -Ni 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's history, rather than as individual files.

s1kd-tools

s1kd-defaults

Description

Table of contents

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>...]
```

Produced by: khzae.net

3 Options

-b \<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	Convert a `.dmtypes` file.
-d	Convert a `.defaults` file.
-F	Convert a `.fmtypes` file.
-f	Overwrite the existing file after conversion.
-i	Initialize a new CSDB by generating the `.defaults`, `.dmtypes` and `.fmtypes` files in the current directory.
-J	Dump the default `.brexmap` file to stdout.
-j \<map\>	Use a custom `.brexmap` file to map a BREX DM to a `.defaults` or `.dmtypes` file.
-s	Sort the entries alphabetically for either file/output format.
-t	Output using the simple text format. Otherwise, the XML format is used by default.
-h -?	Show help/usage message.
--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.

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, 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

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 [-$ <issue>] [-% <dir>] [-FfNqsvh?] <DML>...
```

3   Options

-$ <issue>	Specify which issue of S1000D to use when creating objects.
-% <dir>	Use XML templates in the specified directory instead of the built-in templates of each of the s1kd-new* tools.
-F	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 existing CSDB objects.
-h -?	Show help/usage message.
-N	Omit issue/in-work numbers from the filenames of created CSDB objects.
-q	Do not report errors when any of the CSDB objects already exist.
-s	Do not create CSDB objects, only output the s1kd-new* commands to create them.
-v	Print the filenames of newly created CSDB objects.
--version	Show version information.
<DML>...	One or more S1000D data management lists.

4   Example

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

s1kd-tools

s1kd-newcom

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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>   The code of the comment, in the form of
                             MODELIDENTCODE-SENDERIDENT-YEAR-SEQ-TYPE.

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

                             –   4.2 (default)

	–	4.1
	–	4.0
	–	3.0
	–	2.3
	–	2.2
	–	2.1
	–	2.0

-@ <filename>	Save the new comment as <filename> instead of an automatically named file in the current directory.
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-b <BREX>	BREX data module code.
-C <country>	The country ISO code of the new comment.
-c <sec>	The security classification of the new comment.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-I <date>	The issue date of the new comment in the form of YYYY-MM-DD.
-L <lang>	The language ISO code of the new comment.
-m <remarks>	Set the remarks for the new comment.
-o <orig>	The enterprise name of the originator of the comment.
-p	Prompt the user for values left unspecified.
-q	Do not report an error when the file already exists.
-r <type>	The response type of the new comment.
-t <title>	The title of the new comment.
-v	Print the file name of the newly created comment.
--version	Show version information.

3.1 `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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-tools

s1kd-newddn

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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>	The code of the new data dispatch note, in the form of MODELIDENTCODE-SENDER-RECEIVER-YEAR-SEQUENCE.
-$ <issue>	Specifiy which issue of S1000D to use. Currently supported issues are:

	–	4.2 (default)
	–	4.1
	–	4.0
	–	3.0
	–	2.3
	–	2.2
	–	2.1
	–	2.0

-@ <filename>	Save the new DDN as <filename> instead of an automatically named file in the current directory.
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-a <auth>	Specify the authorization.
-b <BREX>	BREX data module code.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-h -?	Show help/usage message.
-I <date>	The issue date of the new DDN in the form of YYYY-MM-DD.
-m <remarks>	Set the remarks for the new data dispatch note.
-N <country>	The receiver's country.
-n <country>	The sender's country.
-o <sender>	The enterprise name of the sender.
-p <showprompts>	Prompt the user for values left unspecified.
-q	Do not report an error when the file already exists.
-r <receiver>	The enterprise name of the receiver.
-T <city>	The receiver's city.
-t <city>	The sender's city.
-v	Print the file name of the newly created DDN.
--version	Show version information.

3.1   `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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
None	

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

-# <DMC>   The data module code of the new data module.

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

- – 4.2 (default)
- – 4.1
- – 4.0
- – 3.0
- – 2.3
- – 2.2
- – 2.1
- – 2.0

-@ <filename>	Save the new data module as <filename> instead of an automatically named file in the current directory.
-% <dir>	Use XML templates in the specified directory instead of the built-in templates.
-~ <dir>	Dump the built-in XML templates to the specified directory.
-,	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.
-.	Dumps the simple text form of the built-in default `.dmtypes`.
-!	Do not include an info name for the new data module.
-B	When creating a new BREX data module, use the `.defaults` and `.dmtypes` files to add a basic set of context rules.
-b <BREX>	BREX data module code.
-C <country>	The country ISO code of the new data module.
-c <sec>	The security classification of the new data module.
-D <dmtypes>	Specify the `.dmtypes` file name.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-I <date>	Issue date of the new data module in the form of YYYY-MM-DD.
-i <info>	The info name of the new data module.
-j <map>	Use a custom `.brexmap` file when using the -B option.
-k <skill>	The skill level code of the new data module.
-L <language>	The language ISO code of the new data module.
-M <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>	Set remarks for the new data module.
-N	Omit issue/inwork numbers from filename. The "omitIssueInfo" .defaults file key can also be set to control this option.
-n <issue>	The issue number of the new data module.
-O <CAGE>	The CAGE code of the originator.
-o <orig>	The originator enterprise name of the new data module.
-P	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	Prompts the user for any values left unspecified.
-q	Do not report an error when the file already exists.
-R <CAGE>	The CAGE code of the responsible partner company.
-r <RPC>	The responsible partner company enterprise name of the new data module.
-S <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>	The schema URL.
-T <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 <tech>	The tech name of the new data module.
-v	Print the file name of the newly created data module.
-w <inwork>	The inwork number of the new data module.
--version	Show version information.

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 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 type (schema) for data modules based on their info code. By default, the program will search the current directory 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>
```

Info code variants can also be given specific default schema and info names. To do this, include the variant with the info code:

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

The two forms of info codes (with and without variant) can be mixed. Defaults are chosen in the order they are listed in the `.dmtypes` file. An info code with no variant matches all possible variants.

3.4 `.brexmap` file

Refer to the documentation for s1kd-defaults(1) 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 4.2). 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 -# S1000DTOOLS-A-00-07-00-00A-040A-D
```

s1kd-tools

s1kd-newdml

Description

Table of contents

List of tables

References

Table 1   References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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] [<datamodules>]
```

3   Options

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

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

                             –   4.2 (default)

Applicable to: All   **S1000DTOOLS-A-00-0L-00-00A-040A-D**

	– 4.1
	– 4.0
	– 3.0
	– 2.3
	– 2.2
	– 2.1
	– 2.0
-@ <filename>	Save new DML to <filename> instead of an automatically named file in the current directory.
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-b <BREX>	BREX data module code.
-c <sec>	The security classification of the new data module.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-h -?	Show usage message.
-I <date>	The issue date of the new DML in the form of YYYY-MM-DD.
-i <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.
-m <remarks>	Set the remarks for the new data management list.
-N	Omit the issue/inwork numbers from filename.
-n <issue>	The issue number of the new data module.
-p	Prompts the user for any values left unspecified.
-q	Do not report an error when the file already exists.
-R <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 <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>	Create a DMRL using the specified SNS rules.
-v	Print the file name of the newly created DML.
-w <inwork>	The inwork number of the new data module.

--version	Show version information.
<datamodules>	Any number of data module file names to automatically add to the list.

3.1 `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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-tools

s1kd-newimf

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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

-% <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 (4.2).

-~ <dir>   Dump the built-in XML template to the specified directory.

Applicable to: All   **S1000DTOOLS-A-00-0D-00-00A-040A-D**

-b \<BREX\>	BREX data module code.
-c \<sec\>	The security classification of the new ICN metadata file.
-d \<defaults\>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-I \<date\>	The issue date of the new ICN metadata file in the form of YYYY-MM-DD.
-m \<remarks\>	Set the remarks for the new ICN metadata file.
-n \<issue\>	The issue number of the new ICN metadata file.
-O \<CAGE\>	The CAGE code of the originator.
-o \<orig\>	The originator enterprise name of the new ICN metadata file.
-p	Prompts the user for any values left unspecified.
-q	Do not report an error when the file already exists.
-R \<CAGE\>	The CAGE code of the responsible partner company.
-r \<RPC\>	The responsible partner company enterprise name of the new ICN metadata file.
-t \<title\>	The ICN title (if creating multiple ICNs, they will all use this title).
-v	Print the file name of the newly created IMF.
-w \<inwork\>	The inwork issue of the new ICN metadata file.
--version	Show version information.

3.1 `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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-tools

s1kd-newpm

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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

-# <PMC>	The publication module code of the new publication module.
-$ <issue>	Specify which issue of S1000D to use. Currently supported issues are:

    –   4.2 (default)

Produced by: khzae.net

	–	4.1
	–	4.0
	–	3.0
	–	2.3
	–	2.2
	–	2.1
	–	2.0

-@ <filename>	Save new publication module as <filename> instead of an automatically named file in the current directory.
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-b <BREX>	BREX data module code.
-C <country>	The country ISO code of the new publication module.
-c <sec>	The security classification of the new publication module.
-D	Include issue date in referenced data modules.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-I <date>	The issue date of the new publication module in the form of YYYY-MM-DD.
-i	Include issue information in referenced data modules.
-L <language>	The language ISO code of the new publication module.
-l	Include language information in referenced data modules.
-m <remarks>	Set remarks for the new publication module.
-n <issue>	The issue number of the new publication module.
-p	Prompt the user for any values left unspecified.
-q	Do not report an error when the file already exists.
-R <CAGE>	The CAGE code of the responsible partner company.
-r <RPC>	The responsible partner company enterprise name of the new publication module.
-s <title>	The short title of the new publication module.
-T	Include titles in referenced data modules.
-t <title>	The title of the new publication module.
-v	Print the file name of the newly created publication module.
-w <inwork>	The inwork number 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.

3.1   `.defaults` file

Refer to [S1000DTOOLS-A-00-07-00-00A-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

List of tables

References

Table 1   References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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

-# <SMC>   The SCORM content package code of the new SCORM content package.

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

	– 4.2 (default)
	– 4.1
-@ <filename>	Save new SCORM content package as <filename> instead of an automatically named file in the current directory.
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-b <BREX>	BREX data module code.
-C <country>	The country ISO code of the new SCORM content package.
-c <sec>	The security classification of the new SCORM content package.
-D	Include issue date in referenced data modules.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-I <date>	The issue date of the new SCORM content package in the form of YYYY-MM-DD.
-i	Include issue information in referenced data modules.
-k <skill>	The skill level code of the new data module.
-L <language>	The language ISO code of the new SCORM content package.
-l	Include language information in referenced data modules.
-m <remarks>	Set remarks for the new SCORM content package.
-n <issue>	The issue number of the new SCORM content package.
-p	Prompt the user for any values left unspecified.
-q	Do not report an error when the file already exists.
-R <CAGE>	The CAGE code of the responsible partner company.
-r <RPC>	The responsible partner company enterprise name of the new SCORM content package.
-T	Include titles in referenced data modules.
-t <title>	The title of the new SCORM content package.
-v	Print the file name of the newly created SCORM content package.
-w <inwork>	The inwork number 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.

3.1   `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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

List of tables

References

Table 1 References

Data module/Technical publication	Title
S1000DTOOLS-A-00-07-00-00A-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

-@ <filename>   Save the new UPF as <filename> instead of an automatically named file in the current directory.

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

	– 4.2 (default)
	– 4.1
-% <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 (4.2).
-~ <dir>	Dump the built-in XML template to the specified directory.
-d <defaults>	Specify the `.defaults` file name.
-f	Overwrite existing file.
-q	Do not report an error when the file already exists.
-v	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.

3.1 `.defaults` file

Refer to S1000DTOOLS-A-00-07-00-00A-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-tools

s1kd-addicn

Description

Table of contents

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 [-s <src>] [-o <out>] [-fh?] <ICN>...
```

3   Options

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

-s \<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.

4   Example

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

s1kd-ls

Description

Table of contents

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 [-0CDIiLlMNoPrwX] [<object>|<dir> ...]
```

3   Options

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

Applicable to: All   **S1000DTOOLS-A-00-06-00-00A-040A-D**

-C, -D, -L, -M, -P, -X	List comments, data modules, data management lists, ICN metadata files, publication modules, and data dispatch notes respectively. If none are specified, -CDLMPX is assumed.
-h -?	Show the usage message.
-I	Show only inwork issues of objects (inwork != 00).
-i	Show only official issues of objects (inwork = 00).
-l	Show only the latest official/inwork issue of objects.
-N	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.
-o	Show only old official/inwork issues of objects.
-r	Recursively descend in to directories.
-w	Show only writable object files.
--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.

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
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
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

$ 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-tools

s1kd-ref

Description

Table of contents

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 [-filrtdh?] [-$ <issue>] [-s <src>] [-o <dst>]
         [<code>|<filename>]...
```

3   Options

-$ <issue>	Output XML for the specified issue of S1000D.
-d	Include the issue date in the reference (target must be a file)
-f	Overwrite source data module instead of writing to stdout.

-h -?	Show the usage message.
-i	Include the issue information in the reference (target must be a file)
-l	Include the language information in the reference (target must be a file)
-o <dst>	Output to <dst> instead of stdout.
-r	Add the generated reference to the source data module's `refs` table rather than printing the XML to stdout.
-s <src>	Specify a source data module <src> to add references to when using the -r option.
-t	Include the title in the reference (target must be a file).
--version	Show version information.
<code>|<filename>	Either a code, including the prefix (DMC, PMC, etc.), or the filename of a CSDB object.

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>
```

s1kd-tools

s1kd-metadata

Description

Table of contents

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	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 <file>	Use <file> to edit metadata files. <file> consists of lines starting with a metadata name, followed by whitespace,

---

Applicable to: All   **S1000DTOOLS-A-00-09-00-00A-040A-D**

2018-08-31   Page 1

followed by the new value for the metadata (the program uses this same format when outputting all metadata if no <name> is specified).

-e	When showing all metadata, only list editable items. This is useful when creating a file for use with the -c option.
-F <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	When editing metadata, overwrite the object. The default is to output the modified object to stdout.
-H	Lists all available metadata with a short description of each. Specify specific metadata to describe with the -n option.
-l	Treat input as a list of object filenames to read or edit metadata on, rather than an object itself.
-n <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 mode. Non-fatal errors such as a missing piece of optional metadata in an object will not be printed to stderr.
-T	Do not format columns in output.
-t	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>	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 <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.
-w <name>	Show or edit metadata only on objects where the value of <name> is equal to the value specified in the following -v option.
--version	Show version information.
<object>...	The object(s) to show/edit metadata on. The default is to read from stdin.

s1kd-tools

4   Example

```
$ ls
DMC-S1000DTOOLS-A-00-09-00-00A-040A-D_EN-CA.XML
DMC-S1000DTOOLS-A-00-0Q-00-00A-040A-D_EN-CA.XML

$ DMOD=DMC-S1000DTOOLS-A-00-09-00-00A-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                    http://www.s1000d.org/S1000D_4-2/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                    S1000DTOOLS-A-00-09-00-00A-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-tools

s1kd-sns

Description

Table of contents

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>] [-npsh?] [<BREX> ...]
```

3   Options

-d <dir>   The root directory of the new SNS structure. By default, the tool will use the name "SNS" in the current directory.

-h -?   Show usage message.

Applicable to: All   **S1000DTOOLS-A-00-0Y-00-00A-040A-D**

-n	Use only the SNS codes when naming directories. By default, each directory will be named in the form of "snsCode - snsTitle".
-p	Print the SNS structure only.
-s	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.

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-tools

s1kd-transform

Description

Table of contents

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>] [-filh?] [<object> ...]
```

3   Options

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

Produced by: khzae.net

-i	Includes an "identity" template in to each specified stylesheet.
-l	Treat input (stdin or arguments) as lists of CSDB objects to transform, rather than CSDB objects themselves.
-o \<file\>	Output to \<file\> instead of stdout. This option only makes sense when the input is a single CSDB object.
-p \<name\>=\<value\>	Pass a parameter to the last specified stylesheet.
-s \<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.
--version	Show version information.
\<object\> ...	Any number of CSDB objects to apply all specified stylesheets to.

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-tools

s1kd-upissue

Description

Table of contents

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 [-dfHIilNqRrv] [-1 <type>] [-2 <type>]
             [-c <reason>] [-s <status>] [-t <urt>] [<file>...]
```

3   Options

-1 <type>	Set first verification type (tabtop, onobject, ttandoo).
-2 <type>	Set second verification type (tabtop, onobject, ttandoo).
-c <reason>	Add a reason for update to the upissued objects. Multiple RFUs can be added by specifying this option multiple times.
-d	Do not actually create or modify any files, only print the name of the file that would be created or modified.
-f	Overwrite existing upissued CSDB objects.
-H	Mark the last specified reason for update (-c) as a highlight.
-I	Do not change issue date. Normally, when upissuing to the next inwork or official issue, the issue date is changed to the current date. This option will keep the date of the previous inwork or official issue.
-i	Increase the issue number of the CSDB object. By default, the in-work issue is increased.
-l	Treat input (stdin or arguments) as lists of CSDB objects to upissue, rather than CSDB objects themselves.
-N	Omit issue/inwork numbers from filename.
-q	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.
-R	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 old RFUs. 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.
-s <status>	Set the status of the new issue. Default is 'changed'.
-t <urt>	Set the updateReasonType of the last specified reason for update (-c).
-v	Print the file name of the upissued CSDB object.
--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.

4 Examples

4.1 Data module with issue/inwork in filename

```
$ ls
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML

$ s1kd-upissue DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
$ ls
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML

$ s1kd-upissue \
  -i DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
$ ls
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-02_EN-CA.XML
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_001-00_EN-CA.XML
```

4.2 Data module without issue/inwork in filename

```
$ ls
DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_EN-US.XML

$ s1kd-metadata DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_EN-CA.XML \
  -n issueInfo
000-01
$ s1kd-upissue -N DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_EN-CA.XML
$ s1kd-metadata DMC-S1000DTOOLS-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-S1000DTOOLS-KHZAE-FOOBAR_000-01_EN-CA.TXT

$ s1kd-upissue TXT-S1000DTOOLS-KHZAE-00001_000-01_EN-CA.TXT
$ ls
TXT-S1000DTOOLS-KHZAE-FOOBAR_000-01_EN-CA.TXT
TXT-S1000DTOOLS-KHZAE-FOOBAR_000-02_EN-CA.TXT
```

s1kd-tools

s1kd-brexcheck

Description

Table of contents

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>] [-I <path>] [-w <severities>]
               [-BcDfLlpqSstuVvxh?] [<object>...]
```

3 Options

-B	Check each input object against the appropriate built-in S1000D default BREX only. The actual BREX reference of each object is ignored.
-b <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	When a context rule defines values for an object (objectValue), check if the value of each object is within the allowed set of values.
-f	Output only the filenames of CSDB objects with BREX/SNS errors.
-h -?	Show the help/usage message.
-I <path>	Add a search path for BREX data modules. By default, only the current directory is searched.
-L	Treat input as a list of object filenames to check, rather than an object itself.
-l	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	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.
-p	Display a progress bar.
-S[tu]	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	Use shortened, single-line messages to report BREX errors instead of multiline indented messages.
-v -V -q -D	Verbosity of the output.
-w <severities>	Specify a list of severity levels for business rules.
-x	Output an XML report instead of a plain-text one.

--version	Show version information.

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 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 Return value

The number of BREX errors encountered is returned in the exit status code.

5 Example

```
$ DMOD=DMC-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
$ BREX=DMC-S1000D-F-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-S1000DTOOLS-A-00-00-00-00A-040A-D_000-01_EN-CA.XML
  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 53:
    ELEMENT internalRef
      ATTRIBUTE internalRefId
        TEXT
          content=stp-0001
      ATTRIBUTE internalRefTargetType
        TEXT
          content=irtt08
```

s1kd-tools

s1kd-checkrefs

Description

Table of contents

List of tables

References

Table 1   References

Data module/Technical publication	Title
None	

Description

1   General

The **s1kd-checkrefs** tool takes a list of S1000D data modules and pub modules, and lists any invalid references to data/pub modules within them (references to modules not included in the list). It can also update the address items (title, issueDate if applicable) of all valid references using the corresponding address items of the given modules.

2   Usage

```
s1kd-checkrefs [-d <dir>] [-m <object>] [-s <source>] [-t <target>]
               [-ceFfLlnquvh?] <modules>...
```

3   Options

-c   Only check/update references within the content section of modules.

-d <dir>	Check references between data modules in the specified directory. Additional data modules can still be specified with -s.
-e	Check/update external publication references against a pre-defined list of publications.
-F	Fail on first invalid reference and return an error code.
-f	List files which contain invalid references.
-h -?	Show help/usage message
-L	Treat input as a list of data module filenames, rather than a data module itself.
-l	List all invalid references found.
-m <object>	Change all references to the source object specified with -s into references that point to <object>.
-n	When listing invalid references (-l), include the source filenames of the objects in which the invalid references occur.
-q	Do not display error messages for invalid references.
-s <source>	Use only the specified module as the source of address items. Only references to this module will be checked and/or updated in all other modules.
-t <target>	Only check and/or update references within this module. All other modules will only be used as sources.
-u	Update the address items of all valid references found within the specified modules.
-v	Verbose output.
--version	Show version information.

3.1 External publication list (-e)

Since external publications can be of any format, in order to check references to them, their metadata must be specified in an XML format for the s1kd-checkrefs tool to read.

The root element of the XML file is the `externalPubs` element. Each external publication is represented by an element `externalPubAddress`. The identifying elements of the publication are stored in the `externalPubIdent` element (corresponding with the `externalPubRefIdent` element). The address items are stored in the `externalPubAddress` element (corresponding with the `externalPubRefAddressItems` element).

Example:

```
<?xml version="1.0"?>
<externalPubs>
<externalPubAddress>
<externalPubIdent>
```

```
<externalPubCode>s1kd-checkrefs</externalPubCode>
<externalPubTitle>s1kd-checkrefs manual</externalPubTitle>
</externalPubIdent>
<externalPubAddressItems>
<externalPubIssueDate year="2017" month="08" day="14"/>
</externalPubAddressItems>
</externalPubAddress>
</externalPubs>
```

4   Examples

Validate all references in all data modules:

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

Validate references in a single data module:

```
$ s1kd-checkrefs -t <DM> DMC-*.XML
```

Update all references in all data modules:

```
$ s1kd-checkrefs -u DMC-*.XML
```

Change references from one data module to another in all data modules:

```
$ s1kd-checkrefs -s <old DM> -m <new DM> DMC-*.XML
```

s1kd-tools

s1kd-refls

Description

Table of contents

List of tables

References

Table 1 References

Data module/Technical publication	Title
None	

Description

1   General

The **s1kd-refls** tool lists external references to other CSDB objects (dmRef, pmRef), optionally matching them to a filename in the current directory. This makes it easy to see what a given CSDB object "depends" on.

2   Usage

```
s1kd-refls [-acflNqh?] [<object>...]
```

3   Options

-a   List all references, not attempting to match them to an actual filename.

-c   List references in the `content` section of a CSDB object only.

-f	Include the filename of the source object where each reference was found in the output.
-h -?	Show help/usage message.
-l	Treat input (stdin or arguments) as lists of filenames of CSDB objects to list references in, rather than CSDB objects themselves.
-N	Assume filenames of referenced CSDB objects omit the issue info, i.e. they were created with the -N option to the s1kd-new* tools.
-q	Quiet mode. Errors are not printed.
--version	Show version information.
<object>...	CSDB object(s) to list references in. If none are specified, the tool will read from stdin.

4   Example

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

s1kd-tools

s1kd-validate

Description

Table of contents

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>] [-X <URI>] [-flqvx] [<object>...]
```

3   Options

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

Produced by: khzae.net

	this option will force searching to be performed only in the specified directory. This can also be accomplished through the use of XML catalogs.
-f	List invalid files.
-l	Treat input as a list of object names to validate, rather than an object itself.
-v -q	Set the verbosity of the output, verbose or quiet. Verbose will explictly indicate success, rather than simply not displaying any errors. Quiet will not output anything.
-X <URI>	Exclude an XML namespace from the validation. Elements in the namespace specified by <URI> are ignored.
-x	Do XInclude processing before validation.
--version	Show version information.
<object>...	Any number of CSDB objects to validate. If none are specified, input is read from stdin.

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...]
```

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.

s1kd-tools

4   Example

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

s1kd-acronyms

Description

Table of contents

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 [-dlptx] [-n <#>] [-o <file>] [-T <types>]
                [<dmodule>...]
s1kd-acronyms [-fl] [-i|-I|-!] [-m|-M <acr>] [-o <file>]
```

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

3   Options

-D	Remove acronym markup, flattening it to the acronym term.
-d	Format XML output as an S1000D `<definitionList>`.
-f	When marking up acronyms with the -m option, overwrite the input data modules instead of writing to stdout.
-h -?	Show help/usage message.
-i -I -!	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.
	The -I option prompts for all acronyms, not just those with multiple definitions. This can be useful if some occurrences of the acronym term should be ignored.
	The -! option will not prompt for acronyms, instead it will markup 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.
-l	Treat input (stdin or arguments) as lists of filenames of data modules to find or markup acronyms in, rather than data modules themselves.
-M <list>	Like the -m option, but use a custom list of acronyms instead of the default `.acronyms` file.
-m	Instead of listing acronyms in the specified data modules, automatically markup acronyms in the data module using the `.acronyms` file.
-n <#>	Minimum number of spaces after the term in pretty-printed text output.
-o <file>	Output to <file> instead of stdout.
-p	Pretty print text/XML acronym list output.
-T <types>	Only search for acronyms with an attribute `acronymType` whose value is contained within the string <types>.
-t	Format XML output as an S1000D `<table>`.
-x	Use XML output instead of plain text.

--version	Show version information.
<dmodule>...	Data modules to find acronyms in. If none are specified, input is taken from stdin.

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, 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-tools

s1kd-aspp

Description

Table of contents

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.

Applicable to: All   **S1000DTOOLS-A-00-0Q-00-00A-040A-D**

2 Usage

```
s1kd-aspp [-g [-A <ACT>]... [-C <CCT>]... [-G <XSL>]]
          [-p [-a <ID>]] [-cdflxh?] [<modules>...]
```

3 Options

-A <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>   The ID to use for the inline applicability annotation
                        representing the whole data module's applicability. Default is
                        "applic-0000".

-C <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 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 the built-in XSLT used to generate display text for
                        applicability statements.

-f   Overwrite input data module(s) rather than outputting to
                        stdout.

-G <XSLT>   Use custom XSLT to generate display text for applicability
                        statements.

-g   Generate display text for applicability statements.

-l   Treat input (stdin or arguments) as lists of filenames of
                        modules, rather than modules themselves.

-p   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.

-x   Process the modules using the XInclude specification.

--version   Show version information.

<modules>...   The module(s) to preprocess. This can include both individual
                        modules and combined files such as those produced by s1kd-
                        flatten(1).

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

s1kd-tools

```
<para>This step is applicable to A or B.</para>
</proceduralStep>
<proceduralStep applicRefId="applic-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="applic-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-tools

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. Additional directories can be searched using the -I option.

2   Usage

```
s1kd-flatten [-I <path>] [-cdfNpx] <PM> [<DM>...]
```

3   Options

-c   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 -d option is specified.
-d	Remove unresolved references, but do not flatten resolved ones.
-f	Overwrite input publication module instead of writing to stdout.
-h -?	Show help/usage message.
-I <path>	Add <path> to the list of directories that the tool will search when resolving references.
-N	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	Instead of the "flat" PM format, use a "publication" XML format, where the root element `publication` contains XInclude references to the publication module and the referenced data modules.
-x	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.

4   Example

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

s1kd-fmgen

Description

Table of contents

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 [-F <FMTYPES>] [-P <PM>] [-X <XSL> [-p <name>=<val> ...]]
           [-,.fhx?] (-t <TYPE>|<DM>...)
```

3   Options

-,	Dump the built-in `.fmtypes` XML format.
-.	Dump the built-in `.fmtypes` simple text format.

Applicable to: All   **S1000DTOOLS-A-00-0X-00-00A-040A-D**

-h -?	Show usage message.
-F <FMTYPES>	Specify a custom `.fmtypes` file.
-f	Overwrite the specified front matter data module files after generating their content.
-l	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>	Publication module or s1kd-flatten(1) PM format file to generate contents from. If none is specified, the tool will read from stdin.
-p <name>=<value>	Pass a parameter to the XSLT specified with the -X option.
-t <TYPE>	Generate content for this type of front matter when no data modules are specified. Supported types are:

- HIGH - Highlights
- LOEDM - List of effective data modules
- TOC - Table of contents
- TP - Title page

-X <XSL>	Transform the front matter contents after generating them using the specified XSLT. This can be used, for example, to generate content for a descriptive schema data module instead, to support older issues of the specification, or for types of generated front matter not covered by the frontmatter schema.
-x	Do XInclude processing.
--version	Show version information.
<DM>...	Front matter data modules to generate content for.

3.1 `.fmtypes` file

This file specifies a list of info codes to associate with a particular type of front matter. By default, the program will search for a file named `.fmtypes` in the current directory, but any file can be specified using the -F option.

Example of simple text format:

```
001    TP
009    TOC
00S    LOEDM
00U    HIGH
```

Example of XML format

```
<fmtypes>
```

```
<fm infoCode="001" type="TP"/>
<fm infoCode="009" type="TOC"/>
<fm infoCode="00S" type="LOEDM"/>
</fmtypes>
```

4   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-tools

s1kd-icncatalog

Description

Table of contents

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>...]
```

Applicable to: All   **S1000DTOOLS-A-00-0W-00-00A-040A-D**

3   Options

-a <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 <catalog>

Specify the catalog file to manage or resolve against. By default, the file `.icncatalog` in the current directory is used.

-d <ICN>

Delete an ICN from the catalog. The -m option specifies a media group to delete the ICN from.

-f

Overwrite the input CSDB objects when resolving ICNs, or overwrite the catalog file when modifying it. Otherwise, output is written to stdout.

-l

Treat input (stdin or arguments) as lists of filenames of CSDB objects, rather than CSDB objects themselves.

-m <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 <notation>

Specify the notation to reference when adding an ICN with the -a option.

-t

Create a new empty catalog.

-u <URI>

Specify the URI when adding an ICN with the -a option.

-x

Process input CSDB objects using the XInclude specification.

--version

Show version information.

4   Examples
4.1   Resolving ICNs to filenames
A CSDB object may reference an ICN as follows:

```
<!NOTATION png SYSTEM "png">
<!ENTITY ICN-TEST-00001-001-01 SYSTEM "ICN-TEST-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-TEST-00001-001-01"
uri="graphics/ICN-TEST-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-TEST-00001-001-01 SYSTEM
"graphics/ICN-TEST-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-TEST-00001-001-01"
uri="ICN-TEST-00001-001-01.JPG" notation="jpg"/>
</media>
<media name="web">
<icn infoEntityIdent="ICN-TEST-00001-001-01"
uri="ICN-TEST-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-TEST-00001-001-01 SYSTEM "ICN-TEST-00001-001-01.PNG"
NDATA png>

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

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

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


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

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:

– `infoEntityIdent`, the ICN

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

s1kd-tools

s1kd-index

Description

Table of contents

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>] [-fil] [<module>...]
s1kd-index -D [-fil] [<module>...]
```

3   Options

-D   Remove the current index flags from a data module.

-f   Overwrite input module(s).

Applicable to: All   **S1000DTOOLS-A-00-0S-00-00A-040A-D**

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

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, 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>
```

s1kd-tools

```
<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-tools

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 an "instance" of an S1000D CSDB object, derived from a "master" (or "source") object. The tool supports multiple methods of instantiating an object:

–   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.

Applicable to: All   **S1000DTOOLS-A-00-03-00-00A-040A-D**

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

– Changing various pieces of metadata in the instance.

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

2 Usage

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

3 Options

-A	Simplify inline applicability annotations and remove unused ones.
-a	Remove unused inline applicability annotations.
-C <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>	Specify a new data module code (DMC) or publication module code (PMC) for the instance.
-E	Remove the extension from an instance produced from an already extended object.
-e <ext>	Specify an extension on the data module code (DME) or publication module code (PME) for the instance.
-F	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 existing file with same name as the filename generated automatically with -O, if it exists.
-G <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 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.
-I <date>	Set the issue date of the instance. By default, the issue date is taken from the source.

-i <infoName>	Give the data module instance a different infoName.
-K <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 <level>	Set the skill level of the instance.
-L	Source is a list of object filenames to create instances of, rather than an object itself.
-l <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>	Set the remarks for the instance.
-N	Omit issue/inwork numbers from automatically generated filenames.
-n <iss>	Set the issue and inwork numbers of the instance. By default, the issue and inwork number are taken from the source.
-O <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.

-o <file>	Output instance to file instead of stdout.
-P <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>	The ID or primary key of a product in the the specified PCT data module (-P) or the PCT data module referenced by the source data module. 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`
-R <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 -r option. The built-in XSLT used for the above CIR data modules can be dumped with the -x option.

-r <XSL>	Use a custom XSLT script to resolve CIR dependencies for the last specified CIR.
-S	Do not include <sourceDmIdent>/<sourcePmIdent>/ <repositorySourceDmIdent> in the instance.
-s <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 <techName>	Give the instance a different techName/pmTitle.
-U <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 <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	When -O is used, print the automatically generated file name of the instance.
-W	Set the applicability for the whole object, overwriting the current applicability with the user-defined applicability values.
-w	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 <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 <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 -r 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
-Y <text>	Update the applicability for the whole object using the user-defined applicability values, and using <text> as the new display text.
-y	Update the applicability for the whole object using the user-defined applicability values.
--version	Show version information.
<object>...	Source CSDB objects to instantiate.

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.

s1kd-tools

3.2 Instance module code (-c) vs extension (-e)

When creating a data module or publication module instance, the instance should have the same data module/publication module code as the master, with an added extension code, the DME/PME. However, in cases where a vendor does not support this extension or possibly when this tool is used to create "instances" which will from that point on be maintained as normal standalone data modules/publication modules, it may be desirable to change the data module/ publication module code instead. These two options can be used together as well to give an instance a new DMC/PMC as well an extension.

3.3 Removing/simplifying applicability annotations (-a vs -A)

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 and -A options provide two methods of doing this.

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

The -A 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>.

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 neither the -a
or -A option is 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.

Note

The -A option 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. This option is best used when display text will be automatically generated after filtering, such as with the s1kd-aspp tool.

3.4   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>
```

3.5   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>
```

3.6 Resolving CIR dependencies with a custom XSLT script (-r)

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 -r 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="$cir//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> -r <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 -x option.

4   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 -O <dir>
```

s1kd-neutralize

Description

Table of contents

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>] [-flh?] [<object>...]
```

3   Options

-f	Overwrite specified CSDB object(s) automatically.
-h -?	Show usage message.
-l	Treat input (stdin or arguments) as lists of CSDB objects to neutralize, rather than CSDB objects themselves.

| -o <file> | Output neutralized CSDB object XML to <file> instead of stdout. |
| --version | Show version information. |

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

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 [-dfl] [-o <out>] [<data module>...]
```

3   Options

-d	Delete the <refs> element.
-f	Overwrite the data modules automatically.
-l	Treat input (stdin or arguments) as lists of data modules to synchronize references in, rather than data modules themselves.

-o <out>	The resulting XML is written to <out> instead of stdout.
--version	Show version information.
<data module>...	The data module(s) to synchronize references in. Default is to read from stdin.

4   Example

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