



# Intergraph Smart Data Validator

## Administration User's Guide

Version 2018 (3.0)  
September 2017



## Copyright

Copyright © 2015-2017 Hexagon AB and/or its subsidiaries and affiliates. All rights reserved.

This computer program, including software, icons, graphic symbols, documentation, file formats, and audio-visual displays; may be used only as pursuant to applicable software license agreement; contains confidential and proprietary information of Hexagon AB and/or third parties which is protected by patent, trademark, copyright law, trade secret law, and international treaty, and may not be provided or otherwise made available without proper authorization from Hexagon AB and/or its subsidiaries and affiliates.

Portions of the user interface copyright 2012-2014 Telerik AD.

## U.S. Government Restricted Rights Legend

Use, duplication, or disclosure by the government is subject to restrictions as set forth below. For civilian agencies: This was developed at private expense and is "restricted computer software" submitted with restricted rights in accordance with subparagraphs (a) through (d) of the Commercial Computer Software - Restricted Rights clause at 52.227-19 of the Federal Acquisition Regulations ("FAR") and its successors, and is unpublished and all rights are reserved under the copyright laws of the United States. For units of the Department of Defense ("DoD"): This is "commercial computer software" as defined at DFARS 252.227-7014 and the rights of the Government are as specified at DFARS 227.7202-3.

Unpublished - rights reserved under the copyright laws of the United States.

Intergraph Corporation  
305 Intergraph Way  
Madison, AL 35758

## Documentation

Documentation shall mean, whether in electronic or printed form, User's Guides, Installation Guides, Reference Guides, Administrator's Guides, Customization Guides, Programmer's Guides, Configuration Guides and Help Guides delivered with a particular software product.

## Other Documentation

Other Documentation shall mean, whether in electronic or printed form and delivered with software or on Intergraph Smart Support, SharePoint, or box.net, any documentation related to work processes, workflows, and best practices that is provided by Intergraph as guidance for using a software product.

## Terms of Use

- a. Use of a software product and Documentation is subject to the Software License Agreement ("SLA") delivered with the software product unless the Licensee has a valid signed license for this software product with Intergraph Corporation. If the Licensee has a valid signed license for this software product with Intergraph Corporation, the valid signed license shall take precedence and govern the use of this software product and Documentation. Subject to the terms contained within the applicable license agreement, Intergraph Corporation gives Licensee permission to print a reasonable number of copies of the Documentation as defined in the applicable license agreement and delivered with the software product for Licensee's internal, non-commercial use. The Documentation may not be printed for resale or redistribution.
- b. For use of Documentation or Other Documentation where end user does not receive a SLA or does not have a valid license agreement with Intergraph, Intergraph grants the Licensee a non-exclusive license to use the Documentation or Other Documentation for Licensee's internal non-commercial use. Intergraph Corporation gives Licensee permission to print a reasonable number of copies of Other Documentation for Licensee's internal, non-commercial use. The Other Documentation may not be printed for resale or redistribution. This license contained in this subsection b) may be terminated at any time and for any reason by Intergraph Corporation by giving written notice to Licensee.

## Disclaimer of Warranties

Except for any express warranties as may be stated in the SLA or separate license or separate terms and conditions, Intergraph Corporation disclaims any and all express or implied warranties including, but not limited to the implied warranties of merchantability and fitness for a particular purpose and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of such disclaimer. Intergraph believes the information in this publication is accurate as of its publication date.

The information and the software discussed in this document are subject to change without notice and are subject to applicable technical product descriptions. Intergraph Corporation is not responsible for any error that may appear in this document.

The software, Documentation and Other Documentation discussed in this document are furnished under a license and may be used or copied only in accordance with the terms of this license. THE USER OF THE SOFTWARE IS EXPECTED TO MAKE THE FINAL EVALUATION AS TO THE USEFULNESS OF THE SOFTWARE IN HIS OWN ENVIRONMENT.

Intergraph is not responsible for the accuracy of delivered data including, but not limited to, catalog, reference and symbol data. Users should verify for themselves that the data is accurate and suitable for their project work.

## Limitation of Damages

IN NO EVENT WILL INTERGRAPH CORPORATION BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL INCIDENTAL, SPECIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO, LOSS OF USE OR PRODUCTION, LOSS OF REVENUE OR PROFIT, LOSS OF DATA, OR CLAIMS OF THIRD PARTIES, EVEN IF INTERGRAPH CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

UNDER NO CIRCUMSTANCES SHALL INTERGRAPH CORPORATION'S LIABILITY EXCEED THE AMOUNT THAT INTERGRAPH CORPORATION HAS BEEN PAID BY LICENSEE UNDER THIS AGREEMENT AT THE TIME THE CLAIM IS MADE. EXCEPT WHERE PROHIBITED BY APPLICABLE LAW, NO CLAIM, REGARDLESS OF FORM, ARISING OUT OF OR IN CONNECTION WITH THE SUBJECT MATTER OF THIS DOCUMENT MAY BE BROUGHT BY LICENSEE MORE THAN TWO (2) YEARS AFTER THE EVENT GIVING RISE TO THE CAUSE OF ACTION HAS OCCURRED.

IF UNDER THE LAW RULED APPLICABLE ANY PART OF THIS SECTION IS INVALID, THEN INTERGRAPH LIMITS ITS LIABILITY TO THE MAXIMUM EXTENT ALLOWED BY SAID LAW.

## Export Controls

Intergraph Corporation's commercial-off-the-shelf software products, customized software and/or third-party software, including any technical data related thereto ("Technical Data"), obtained from Intergraph Corporation, its subsidiaries or distributors, is subject to the export control laws and regulations of the United States of America. Diversion contrary to U.S. law is prohibited. To the extent prohibited by United States or other applicable laws, Intergraph Corporation software products, customized software, Technical Data, and/or third-party software, or any derivatives thereof, obtained from Intergraph Corporation, its subsidiaries or distributors must not be exported or re-exported, directly or indirectly (including via remote access) under the following circumstances:

- a. To Cuba, Iran, North Korea, the Crimean region of Ukraine, or Syria, or any national of these countries or territories.
- b. To any person or entity listed on any United States government denial list, including, but not limited to, the United States Department of Commerce Denied Persons, Entities, and Unverified Lists, the United States Department of Treasury Specially Designated Nationals List, and the United States Department of State Debarred List ([https://build.export.gov/main/ecr/eg\\_main\\_023148](https://build.export.gov/main/ecr/eg_main_023148)).
- c. To any entity when Customer knows, or has reason to know, the end use of the software product, customized software, Technical Data and/or third-party software obtained from Intergraph Corporation, its subsidiaries or distributors is related to the design, development, production, or use of missiles, chemical, biological, or nuclear weapons, or other un-safeguarded or sensitive nuclear uses.
- d. To any entity when Customer knows, or has reason to know, that an illegal reshipment will take place.
- e. Any questions regarding export/re-export of relevant Intergraph Corporation software product, customized software, Technical Data and/or third-party software obtained from Intergraph Corporation, its subsidiaries or distributors, should be addressed to PPM's Export Compliance Department, 305 Intergraph Way, Madison, Alabama 35758 USA or at [exportcompliance@intergraph.com](mailto:exportcompliance@intergraph.com). Customer shall hold harmless and indemnify PPM and Hexagon Group Company for any causes of action, claims, costs, expenses and/or damages resulting to PPM or Hexagon Group Company from a breach by Customer.

## Trademarks

Intergraph®, the Intergraph logo®, Intergraph Smart®, SmartPlant®, SmartMarine, SmartSketch®, SmartPlant Cloud®, PDS®, FrameWorks®, I-Route, I-Export, ISOGEN®, SPOOLGEN, SupportManager®, SupportModeler®, SAPPHIRE®, TANK, PV Elite®, CADWorx®, CADWorx DraftPro®, GTSTRUDL®, and CAESAR II® are trademarks or registered trademarks of Intergraph Corporation or its affiliates, parents, subsidiaries. Hexagon and the Hexagon logo are registered trademarks of Hexagon AB or its subsidiaries. Microsoft and Windows are registered trademarks of Microsoft Corporation. MicroStation is a registered trademark of Bentley Systems, Inc. Other brands and product names are trademarks of their respective owners.

# Contents

<b>Preface.....</b>	<b>9</b>
Smart Data Validator Product Documentation .....	9
Customer Support .....	10
<b>What's New in Smart Data Validator Administration? .....</b>	<b>11</b>
<b>Welcome to Smart Data Validator .....</b>	<b>16</b>
Smart Data Validator process architecture .....	16
Workflow .....	18
Auto-generation of export mapping .....	19
Auto-generation of rules for validation.....	19
Attaching a workflow to exported objects .....	19
Smart Data Validator work process .....	20
<b>Using Smart Data Validator .....</b>	<b>21</b>
<b>Set Smart Data Validator Default Options.....</b>	<b>23</b>
Set illegal characters for import mapping .....	25
<b>Define Target Systems.....</b>	<b>26</b>
Create a new target system .....	26
Edit a target system .....	27
<b>Define Import Mapping.....</b>	<b>28</b>
Import mapping .....	28
Auto-generate export mapping .....	29
Auto-generate validation rules .....	29
Create an import definition.....	30
Edit an import definition .....	31
Copy an import definition .....	31
View mapping summary.....	32
<b>Define Column Headers .....</b>	<b>33</b>
Column types .....	34
Create a new column header.....	35
Connect to a target system .....	36
Search using a local search .....	36
Create a new object or property using free text .....	37
Create column header mappings.....	37
Object mapping .....	37
Property mapping .....	39
Relationship mapping .....	39

Relationship property mapping.....	40
Summary map .....	40
Edit column headers .....	41
Manage column header order.....	42
Define a raw attribute map.....	42
Raw Attribute Map .....	43
Edit a raw attribute map.....	44
Generate raw attribute property mappings for export .....	44
<b>Define Export Mapping .....</b>	<b>47</b>
Create an export mapping .....	48
Edit an export mapping .....	48
Copy an export mapping .....	48
Define stage object to target object mappings.....	49
Create a stage object to target object mapping .....	49
Edit a stage object to target object mapping .....	50
Define stage properties and relationships to target object mappings.....	50
Stage property to target property mappings.....	51
Stage relationship to target property mapping .....	52
Stage relationship to target relationship mappings .....	53
Stage relationship property to target relationship property mappings.....	54
<b>Define Validation Rules and Rulesets .....</b>	<b>56</b>
Create a new validation rule set.....	57
Edit a validation rule set.....	57
Copy a validation rule set.....	58
Create a validation rule .....	58
Edit a validation rule.....	59
Manage validation rules .....	59
Configure the automatic generation of validation rules .....	59
Rule auto-generation model .....	60
Naming convention for rule auto-generation .....	63
Smart Data Validator rule severity levels .....	63
Create a validation rule relationship .....	63
<b>Define Implicit Delete Rules .....</b>	<b>64</b>
Create an implicit delete rule .....	65
Edit an implicit delete rule .....	66
<b>Define Job Definitions.....</b>	<b>67</b>
Create a job definition .....	68
Edit a job definition.....	70
Delete a job definition .....	70
<b>Use Function Column Type Mappings .....</b>	<b>71</b>
Use the Computed column .....	71
Use the Prompted API column.....	75

<b>Triggering a Job External to the Job Management Module .....</b>	<b>76</b>
Using silent workflows and job scheduler .....	77
Understanding the control file format .....	77
Trigger the job .....	80
<b>Converting a SmartPlant Foundation Object to a Smart Data Validator Job .....</b>	<b>82</b>
Load files in SmartPlant Foundation Desktop Client using Loader .....	83
Create a new job details item object .....	84
Create a new job details object .....	85
Create a submittal .....	86
Attach the Convert Import Validate Export workflow automatically .....	86
Monitor the submittal .....	88
View the job .....	88
Use planned revisions of documents with an MDR submittal .....	88
<b>Functions .....</b>	<b>89</b>
Computed column functions .....	89
Prompted API column functions .....	100
Document functions .....	100
Target system computed columns .....	101
Non-target system computed columns .....	104
Document function examples .....	105
Date and time functions .....	106
ConvertToHash function .....	106
Computed column .NET methods .....	107
<b>Unique Keys and Query Definitions .....</b>	<b>110</b>
Target system unique keys .....	110
Target system query definitions .....	111
<b>Validation Rules and Actions .....</b>	<b>113</b>
Validation rule description .....	114
Actions .....	117
Export mapping actions .....	119
Relationship cardinality rules .....	121
Rename action in Smart Data Validator .....	121
DEVTag rename mapping example .....	122
Design document master rename mapping .....	122
<b>Understand Implicit Delete Functionality .....</b>	<b>123</b>
Deleting objects .....	123
Deleting objects by property grouping .....	123
Deleting objects by object and relationship grouping .....	124
Deleting relationships .....	125
<b>Environment Variables .....</b>	<b>127</b>
Environment variable examples .....	129

<b>Document Mappings .....</b>	<b>131</b>
Import mapping examples.....	134
CSV file example .....	134
Object mappings example .....	134
Property mappings example.....	134
UID Definitions mappings example .....	135
Relationships mappings example.....	135
Documents with concurrent engineering .....	135
Exporting multiple revisions .....	136
Reservation and activation of document masters.....	138
Mappings for published documents .....	138
Tool signature mappings .....	139
Published document mappings .....	140
Export mappings example .....	146
<b>Export File Mappings .....</b>	<b>147</b>
CSV mapping example .....	148
CSV file example .....	148
Object mappings example .....	148
Property mappings example.....	148
UID Definitions mappings example .....	148
Relationships mappings example.....	149
Files with custom target systems .....	149
Creating new versions and copying files from the previous version .....	150
Mappings for supported files .....	150
CSV file example .....	152
Object mappings example .....	152
Property mappings example.....	153
Relationships mappings example.....	153
<b>Smart Data Validator Mapping Export.....</b>	<b>155</b>
<b>Delete Job Definitions Using SmartPlant Foundation .....</b>	<b>156</b>
<b>Supported CSV File Formats.....</b>	<b>157</b>
CSV files not in invariant format.....	158

<b>Clash Detection .....</b>	<b>160</b>
<b>Export Objects configured with ENS in Target System .....</b>	<b>162</b>
<b>Case-Insensitive Support for Oracle Database .....</b>	<b>166</b>
<b>Job Reports.....</b>	<b>171</b>
<b>EPC Database Registration After Project Completion .....</b>	<b>174</b>
<b>Glossary .....</b>	<b>175</b>
<b>Index .....</b>	<b>179</b>



# Preface

This document contains information and procedural instructions for the administration and operation of Intergraph Smart® Data Validator. The content is the same as the online Help, delivered as part of the standard installation. This document is intended for all system administrators and administration users who are working with the validation, transformation, and loading of data into SmartPlant® Foundation or any other environment.

★ **IMPORTANT** You may not see all the commands and functionality documented in this *Smart Data Validator Administration User's Guide*, depending on your security configuration, role, and access permissions. Only users with specific administration role and access permissions are able to create, edit, or delete validation rules or job definitions. For role or access changes, see the *SmartPlant® Foundation Desk Top Client User's Guide* or contact your system administrator.

## Smart Data Validator Product Documentation

Smart Data Validator documentation is available as Help and as PDF files. To view printable guides for Smart Data Validator, click **Help > Printable Guides** in the software.

Hexagon PPM gives its customers permission to print as many copies of the delivered PDF files as they need for their non-commercial use. Do not print the PDF files for resale or redistribution.

### Release Bulletin

- *Smart Data Validator Release Bulletin* - Provides information on Smart Data Validator features for the current release.

### Installation and Overviews

- *Smart Data Validator Installation and Setup Guide* - Provides installation and setup instructions for Smart Data Validator.
- *Smart Data Validator Getting Started Guide* - Provides overview information to help users know about the Smart Data Validator functionalities.

### Administration User's Guide

- *Smart Data Validator Administration User's Guide* - Provides instructions for administering the configuration of Smart Data Validator.

### Job Management User's Guide

- *Smart Data Validator Job Management User's Guide* - Provides instructions on the management of jobs that process imported data to a specified workflow and export that data to a target system or target systems.

### Customization Guide

- *Smart Data Validator Customization Guide* - Provides instructions for creating and modifying the various areas of customization possible in Smart Data Validator and details on provided samples included with the release.

## Troubleshooting Guide

- *Smart Data Validator Troubleshooting Guide* - Provides information about troubleshooting the installation and configuration of Smart Data Validator.

## Customer Support

For the latest support information for this product, use a web browser to connect to <http://hexagonppm.com/ppm-support> (<http://hexagonppm.com/ppm-support>). Also, you can submit any documentation comments or suggestions you might have on our support site.

To access the Technical User Forum, go to

<http://www.intergraph.com/ppm/customers/tuf/foundation.aspx>  
(<http://www.intergraph.com/ppm/customers/tuf/foundation.aspx>)

## SECTION 1

# What's New in Smart Data Validator Administration?

### Version 2018

- A new process step, **SDVMDRProcessStep**, is now available in the **Convert Import Validate Export** workflow and allows you to relate the MDR document master with other document masters in the submittal. For more information, see *Attach the Convert Import Validate Export workflow automatically* (on page 86). (CR-AM-116012)
- The import mappings delivered with Smart Data Validator to process the Master Document Registry submittal are now enhanced to support the relationship between the MDR document master and other document masters in the submittal. For more information, see *Load files in SmartPlant Foundation Desktop Client using Loader* (on page 83). (CR-AM-111304)
- You can now use the **Check for Unmapped Properties** option in the **Edit Rule** dialog box to report the unmapped properties in the validation report. For more information, see *Validation rule description* (on page 114). (CR-AM-116148)
- You can now create a new job details item using the SmartPlant Foundation Desktop Client. You can use the job details item to specify the required details to process the CSV files attached to the submittal, for converting it into a job. For more information, see *Create a new job details item object* (on page 84). (CR-AM-114447)
- You can now import, validate, and export published documents using Smart Data Validator. For more information, see *Mappings for published documents* (on page 138). (CR-AM-117430, CR-AM-117431)
- You can now specify a version number for a document during import mapping using the **SPFDocVersion** property. For more information, see *Document Mappings* (on page 131). (CR-AM-117428)
- A new validation rule, **Check Duplicate Issue Dates**, checks to see if there are revisions with duplicate issue dates for the same master document and prevents the revisions from being loaded. For more information, see *Validation Rules and Actions* (on page 113). (CR-AM-111891)
- You can now find the class definition of a parent object using a new computed column, **GetParentObjectClassDef**. For more information, see *Use the computed column* (on page 71). (CR-AM-114222)
- You can now use the **ConvertToHash** computed column to generate a unique hash code for objects when you have case-sensitive data in your CSV file but the database you are exporting to is case-insensitive. For more information, see *Use the computed column* (on page 71) and *ConvertToHash function* (on page 106). (CR-AM-123252)
- The **Date Time**, **Double**, **Integer**, **Regular Expression**, **String Does Not Start With**, **String Not Equal To**, and **Check Properties Against Schema** rules are now enhanced to

support the validation of relationship properties. For more information, see *Validation Rules and Actions* (on page 113). (CR-AM-109129, CR-AM-110983)

- The **Check Properties Against Schema** rule can now be auto-generated when configuring relationship property mappings. For more information, see *Rule auto-generation model* (on page 60). (CR-AM-110983)
- Smart Data Validator now automatically changes the status of an existing planned revision of a document from RESERVED to RESERVEDSUPERSEDED when a newer revision is loaded. For more information, see *Use planned revisions of documents with an MDR submittal* (on page 88). (CR-AM-110845)
- The planned revision status of a document is now automatically changed from RESERVEDSUPERSEDED to RESERVED if all the actual revisions are deleted for any reason. For more information, see *Use planned revisions of documents with an MDR submittal* (on page 88). (CR-AM-112195)
- You can now unzip any zipped files uploaded with a submittal object if they are not referenced in the attached CSV file. For more information, see *Attach the Convert Import Validate Export workflow automatically* (on page 86). (CR-AM-113029)
- A new validation rule, **Check File Exists for Document Revision**, validates the document revision and checks if any file is attached to it, unless the document revision is a planned revision. In the case of a planned revision, the rule does not check for attached files. For more information, see *Validation rule description* (on page 114). (CR-AM-113436)
- Smart Data Validator now supports multiple values in columns when processing CSV files. You can create a column header with multiple values by selecting the **Multiple Values Column** checkbox in the **Create a new column header** dialog box in Smart Data Validator Administration. For more information, see *Create a new column header* (on page 35). (CR-AM-113407)
- You can specify a default separator for CSV columns with multiple values in the **System Options** wizard in Smart Data Validator Administration. For more information, see *Set Smart Data Validator Default Options* (on page 23). (CR-AM-113407)
- A new validation rule, **Check Is Claimable**, checks to see if a data object is claimable into a Smart Digital Asset | Collaboration project and prevents the object from being loaded. For more information, see *Validation rule description* (on page 114). (CR-AM-113608)
- A new validation rule, **Check Is Revisable**, checks to see if a document is revisable to a Smart Digital Asset | Collaboration project and prevents the document revision from being loaded. For more information, see *Validation rule description* (on page 114). (CR-AM-113608)
- You can now delete a job definition using Smart Data Validator Administration. For more information, see *Delete a job definition* (on page 70). (CR-AM-107489)
- You can now create validation rules for relationship properties using Smart Data Validator Administration. For more information, see *Create a validation rule* (on page 58). (CR-AM-107788)
- You can now export relationship properties to a target system using Smart Data Validator Administration. For more information, see *Relationship property mapping* (on page 40) and *Stage relationship property to target relationship property mappings* (on page 54). (CR-AM-107486)
- A new workflow, **Convert Import Validate Export**, is now delivered in the product installation folder. With this workflow, you can convert a SmartPlant Foundation object such

as a submittal into a Smart Data Validator job and process it further for loading data into Smart Digital Asset | Collaboration. For more information, see *Converting a SmartPlant Foundation Object to a Smart Data Validator Job* (on page 82). (CR-AM-107675)

- You can now create a job details object using the SmartPlant Foundation Desktop Client, and relate it to a submittal classification type. For more information, see *Create a new job details object* (on page 85). (CR-AM-107675)
- You can now extract the value of any property in a job using a new computed column function, **GetJobDetails**. For more information, see *Use the computed column* (on page 71). (CR-AM-107676)
- You can now attach a workflow to documents after they are exported. For more information, see *Attach the Convert Import Validate Export workflow automatically* (on page 86). (CR-AM-108543)
- A new property definition, **VTLJobNumber**, is now available on the IVTLJob interface definition for creating staging tables with the prefix, 'VTL'. (CR-AM-107673)
- You can now split the CSV file attached to a SmartPlant Foundation object by specifying the column name and values in it. For more information, see *Create a new job details object* (on page 85). (CR-AM-108544)
- The **Import Definition Copy** and **Export Mappings Copy** commands now copy the relationship property mappings. (CR-AM-108451)
- The inconsistencies in the relationship properties are now detected and reported when you create a job definition using Smart Data Validator Administration. (CR-AM-108453)
- Termination or deletion of target system relationships using implicit delete process now takes into consideration both ends of the relationship given in the implicit delete rule. (TR-AM-109074)
- Import mappings, export mappings, and rule sets to process Master Document Registry, Master Tag Registry, and Tag Document Registry submittals are now delivered with Smart Data Validator. They are available at *[Installation folder]\SDV\2016\ProjectData\Mappings*. For more information see *Load files in SmartPlant Foundation Desktop Client using Loader* (on page 83). (CR-AM-107692, CR-AM-107677, CR-AM-107678)
- The job details object is now associated with the one or more classification types of SmartPlant Foundation objects by creating a relationship between the job details object and the classification type. For more information, see *Create a new job details object* (on page 85). (CR-AM-109189)
- When the staging and target systems are the same, conflicts can occur while you are creating export mappings for the properties of an object in a raw attribute map. You can now resolve the conflicts by adding a suffix to the property name in the Raw Attribute Map which will ensure that the staging and target system properties have different names. For more information, see *Define a raw attribute map* (on page 42). (CR-AM-109442)

### Version 2016

- You can now evaluate the value of the property using the **InvertedFilter** computed column when the property name in the inverted CSV file matches the property name configured in the computed column. For more information, see *Generate raw attribute property mappings for export* (on page 44), *Use the Computed column* (on page 71), and *Functions* (on page 89). (CR-AM-86324)

- Export functionality is extended to allow the export of markup files, navigation files, graphics files, and alternate rendition files. For more information, see *Mappings for supported files* (on page 150). (CR-AM-104720)
- You can now save the job details by selecting the **Record Job Details** option in the **Create Job Definition** dialog box. The job details data is stored in the staging system, and is available even if the job has been deleted in Smart Data Validator. For more information, see *Define Job Definitions* (on page 67), *Create a job definition* (on page 68), and *Job Reports* (on page 171). (CR-AM-78996)
- Prompted columns now appear in numerical order in the **Prompted values** page in the **Create Job** dialog box. (CR-AM-83126)
- You can now prevent exporting the objects that have no value against a property which is used for UID creation. For more information, see *Set Smart Data Validator Default Options* (on page 23). (CR-AM-91357)
- You can now set the options for creation of new document versions for the same revision in the **Job Definition** dialog box. For more information, see *Create a job definition* (on page 68) and *Creating new versions and copying files from the previous version* (on page 150). (CR-AM-99691)
- The object, property, relationship, and relationship property mapping names can include underscore and alphanumeric characters. For more information, see *Create column header mappings* (on page 37) and *Define Export Mapping* (on page 47). (TR-AM-102440)
- You can now reserve document masters in the target system, and later activate the reserved documents. For more information, see *Reservation and activation of document masters* (on page 138). (CR-AM-97647)
- You can now verify if the UOM of the object property is the same as the UOM configured in the target system by using the **UOM** validation rule. For more information, see *Validation rule description* (on page 114). (CR-AM-89679)
- You can now retrieve the number of revisions available for a document from the target system by using the computed column **GetDocumentRevisionsCountFromTargetSystem**. For more information, see *Use the Computed column* (on page 71) and *Target system computed columns* (on page 101). (CR-AM-88517)
- You can now use the **Allow Disabled Enum Entries** option in the **Edit Rule** dialog box to report the disabled enums in the validation report. For more information, see *Validation rule description* (on page 114). (CR-AM-86626)
- You can now validate an object and its properties in the target system by using the **Validate Only** action. For more information, see *Actions* (on page 117). (CR-AM-82757)
- The case-sensitivity of database interactions with SQL Server databases is based on the collation settings. For more information, see *Case-Insensitive Support for Oracle Database* (on page 166). (CR-AM-98874)
- In the **Regular Expression** rule, you can now include object properties to create regular expression patterns. For more information, see *Validation rule description* (on page 114). (CR-AM-89678)
- Deleting and terminating required properties is now restricted in Smart Data Validator. The validation report informs users of such failures. For more information, see *Property mapping* (on page 39). (CR-AM-85941)

- Database interactions with Oracle are not case-sensitive anymore in Smart Data Validator. To take advantage of this feature, you must configure export mapping using query definitions to search for objects other than UIDs, unique keys, and schema items. For more information, see *Case-Insensitive Support for Oracle Database* (on page 166). (CR-AM-95022)
- You can now select optional interfaces that are available for a selected class definition in the target system to instantiate them on the objects during export. For more information, see *Object mapping* (on page 37). (CR-AM-79090)

### Version 2015 R1

- You can now invoke **Check Exists** action for the **Staging System Cardinality** and the **Target System Cardinality** rules. For more information, see *Validation Rules and Actions* (on page 113). (CR-AM-90002)
- You can now filter properties by selecting an object column from the **Select Object Mapping to Filter Properties** list while creating a property mapping. For more information, see *Property mapping* (on page 39). (CR-AM-90577)
- The **Date Time**, **Double**, **Integer**, and **String Length** rule forms are now enhanced to validate that the minimum value is less than or equal to the maximum value. The **Regular Expression** rule form is now enhanced to validate that the regular expression value is valid. For more information, see *Validation Rules and Actions* (on page 113). (CR-AM-85162)
- You can select **Exclude Claimed Items** on the **Create Implicit Delete Rule** dialog box to stop the deletion of objects and relationships that are claimed to lower configuration. For more information, see *Create an implicit delete rule* (on page 65). (CR-AM-86764)
- You can now delete crashed jobs irrespective of status, after the wait time (in days) listed in **System Options** has elapsed. For more information, see *Set Smart Data Validator Default Options* (on page 23). (CR-AM-88168)
- The confirmation message for forms closed without any modifications has been removed to improve user experience. (CR-AM-78103)
- You can now export objects that are Engineering Numbering System (ENS) configured in the target system. For more information, see *Export Objects configured with ENS in Target System* (on page 162). (CR-AM-94142)
- You can now trigger a job external to the Smart Data Validator Job Management module. Prior to triggering the job, you must create the import mapping, validation rules, export mapping, implicit delete rules, and job definition in the Smart Data Validator Administrator module. For more information, see *Triggering a Job external to the Job Management module* (on page 76). (CR-AM-90580, CR-AM-95678)

## SECTION 2

# Welcome to Smart Data Validator

Smart Data Validator is a comprehensive solution that enables you to significantly reduce the time and costs associated with the validation of data to be exported into your target system databases. Smart Data Validator manages the data for quality assurance purposes, and provides reports that can be used for both greenfield and brownfield environment data to help minimize quality issues. By increasing the accuracy of imported data using validation rules, you can improve the quality of the data exported to a new database system or an existing brownfield plant. Smart Data Validator helps manage this migration of data from projects and contractors into new projects, databases, and legacy database systems in easily configurable steps, which can help accelerate any handover.

Smart Data Validator manages the acquisition of data from multiple sources and verifies the quality of the incoming data before it is loaded into target projects or operational database systems. Data that you load into Smart Data Validator is subject to rigorous quality control, before being exported into the target systems, such as Hexagon PPM's SmartPlant® Enterprise suite or other database systems. Smart Data Validator improves the range, quality, consistency, and traceability of the validation performed, which enables you to check the quality of information in deliverables before you send to your customers.

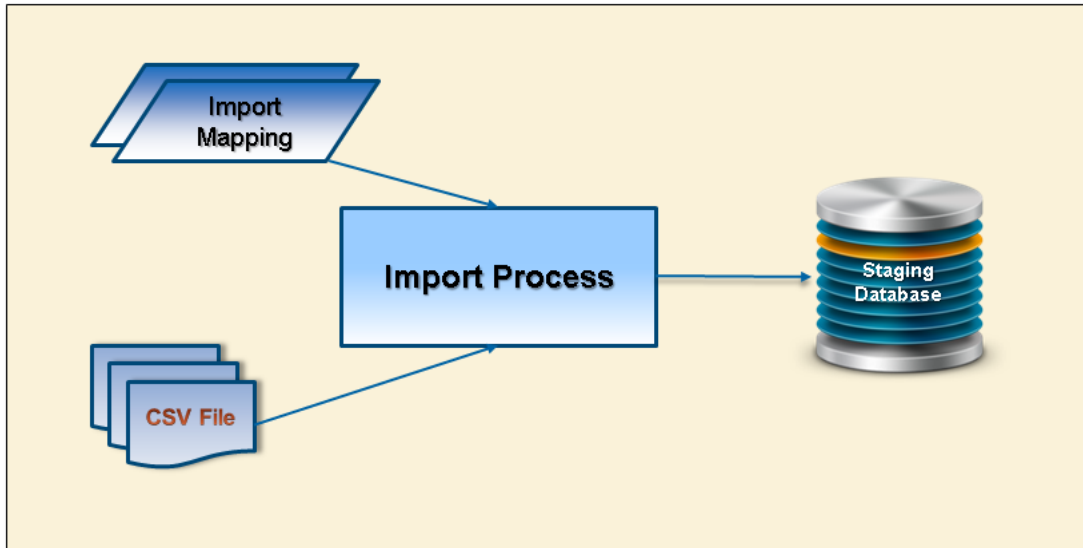
Smart Data Validator has been designed so that you no longer need to write any code for the validation and export processes. These have been replaced by an automated rules based system for the validation and mapping of data objects. A consistent SmartPlant Foundation workflow operation ensures that the requirements for any imported data are met before export to a specified target system.

## Smart Data Validator process architecture

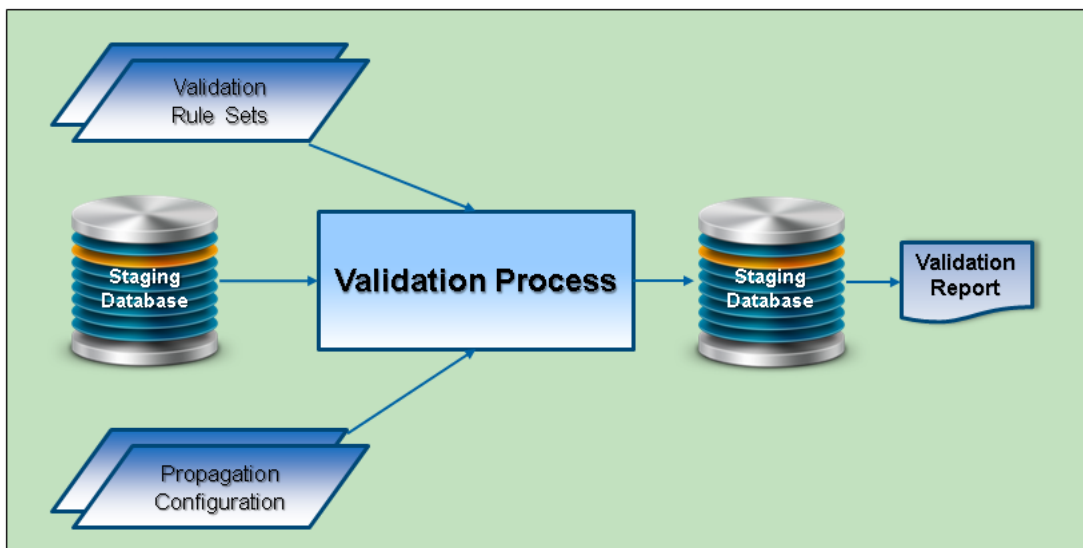
The following diagrams illustrate the architecture of the Smart Data Validator process.



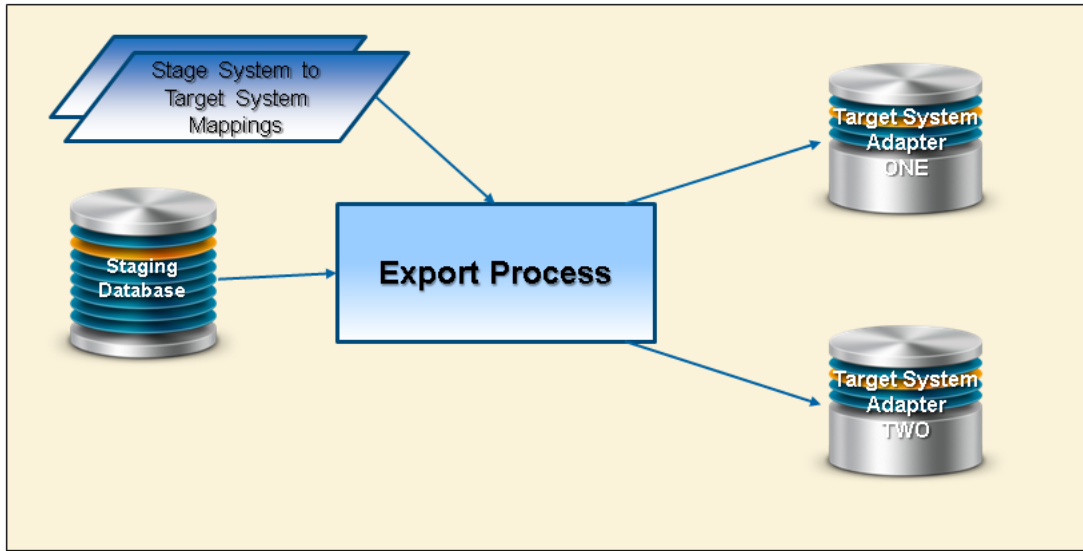
### Import process



### Validation process



## Export process



## Workflow

A key feature of Smart Data Validator is that the validation of data imported into the staging area and the export of objects to one or more target systems are configured in a workflow run using Smart Data Validator Job Management. Smart Data Validator is delivered with three out-of-the-box workflow templates:

- **Import Validate Export**
- **Import Validate Implicit Delete Export**
- **Import Validate Export Delete Job**

The workflow drives the behavior of Smart Data Validator by bringing each component together as a separate process step, each of which is approved as the data progresses through the workflow:

- The import of information is defined using mapped column headers that can be defined from a specified target system configuration.
- Rule sets are defined for the validation of the imported data, both locally in the staging area and against a target system.
- Data is validated with or without a step that produces an error report.
- Implicit delete functionality is configured by setting up rules that determine what data is deleted or terminated. The deleted and terminated objects are included in an optional deletes report before you actually delete the objects during the export process.
- Export data is filtered and mapped to the intended target system (or systems) with or without a report or a confirmation step.
- A workflow can be attached to the exported objects using a process step, `SDVAttachWorkflowStep`. For example, if you want to attach a review workflow to a large

number of documents, it is easy and automatic using the process step, instead of manually attaching the workflow to each document.

**NOTE** Smart Data Validator does not support reviving an object after it has been terminated. This includes reviving any data that was created as the result of a claim.

**TIP** If you do not have a SmartPlant Foundation target system to drive the import mapping and validation, the imported data can be mapped to the staging area using free text or by using existing schema items in the staging area. The validation can then be run against the staging area. For more information on customizing the workflow, see *Workflow customization* in the *Smart Data Validator Customization Guide*.

## Auto-generation of export mapping

When defining import mapping, Smart Data Validator can automatically generate the export mapping required to transfer data from the staging area columns to the target system columns. This is possible only if a target system is selected on the import mapping. The export mapping is used during the export process, where the imported data has been mapped to match the existing architecture of the target system.

## Auto-generation of rules for validation

Rule sets are used to enforce levels of quality. Some properties may be missing or have values outside of the standard operational scope. Smart Data Validator manages the validation of this data and lets you determine if the issue is serious enough to prevent the export of the data or not. Rules are automatically generated when you define your import mapping and saves you time when creating rules. This allows the automatic definition and execution of any rules, for example, ensuring that a check exists for an object in the target system or asking the target system and definitions for the type of property and its set of valid property values.

## Attaching a workflow to exported objects

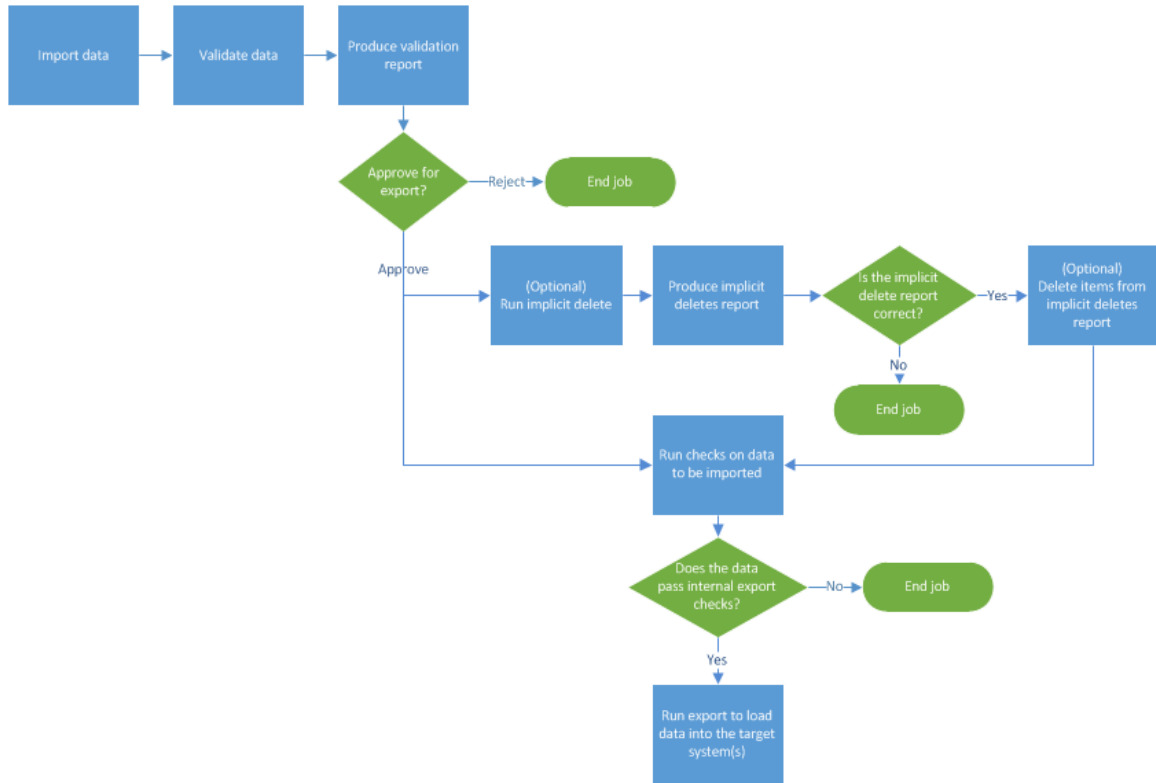
A workflow can be attached to the exported objects in the target system by configuring a process step, `SDVAttachWorkflowStep`, in any of the three delivered workflows. This step uses two arguments: the name of the workflow you are attaching to the exported objects, and the relationship definitions that relate the objects with the SDV job. Example:  
`SDVAttachWorkflowStep,  
Document_Review_And_Sign_Off,SCLBSubmittalDocumentVersions,VTLJobForObjAbstract.`  
The relationship definitions must be mapped using the environment variable, `JOBNAME`, as a computed column type while creating the import definition. For example, if you are attaching a workflow to a document, then you must create a relationship definition using `JOBNAME` at one end and the document name at the other end.

### NOTES

- The staging and target systems must be configured on the same site.
- It is not required to model the relationship definitions if they are already available in the schema.

## Smart Data Validator work process

The following diagram illustrates the main steps for using Smart Data Validator components to import data and load it into the target system:



## SECTION 3

# Using Smart Data Validator

With Smart Data Validator Administration, you can control how supplied data is imported, validated, and loaded into a specific target system, such as a SmartPlant Enterprise database. Smart Data Validator Administration consists of a set of components that manage the import of CSV files and validate the data for loading into the target system database.

**⚠ CAUTION** We recommend using a text editor like Notepad to modify your CSV files to avoid any unexpected results.

After the data is imported into the staging area, the remaining Smart Data Validator components are fully configurable and optional. For example, a job might include just the import and validation of the new data and provide a report on the quality of the imported data. You can control the export of the data so that the export process is not initiated until the quality of the imported data is approved.

Data files are imported using an import definition, which specifies how the data is mapped to the target system using existing class definitions, property definitions, and relationship definitions. The imported data is validated against defined validation rule sets that can use the target system as their basis, and the export mapping ensures that the data is mapped correctly into the target system format. Job definitions combine all these components into one set of steps.

**★ IMPORTANT** Smart Data Validator does not process reference files. For example, if a document contains a reference to another file and the document is imported into Smart Data Validator, the software does not maintain any relationships to the reference file when exported to a target system. If you want to manage reference files, this must be done in SmartPlant Foundation.

There are several components that are defined in Smart Data Validator Administration and used to manage the movement of data:

- Import process - Uses a definition to map data files to the staging area.
- Validation process - Evaluates imported data against a defined set of rules to ensure the validity of the data.
- Implicit delete process - Allows you to manage the deletion or termination of objects from a target system.
- Export process - Uses a defined mapping, based on the structure of the target system, to manage loading data into the final destination system.
- Job definition - Links together various importing, validating, and loading components to define how a set of data will be processed by Smart Data Validator Job Management.
- Target system - Destination where the validated data is loaded.

### NOTES

- The UID definitions (unique identifiers) in the staging area must match the UID definitions in the target system. UID definitions must be set on all objects that Smart Data Validator exports from the staging area. When you create your mappings, you must set the correct UID in the import mapping by setting up the UID definitions for any object mapping. For more information, see *Define Column Headers* (on page 33).

- Smart Data Validator Job Management runs the complete process using all the components to import, validate, and load the data into a target system database. For more information on Job Management, see Smart Data Validator Job Management User's Guide.
- 


### **What do you want to do?**

- *Set Smart Data Validator Default Options* (on page [23](#))
  - *Define Target Systems* (on page 26)
  - *Define Import Mapping* (on page 28)
  - *Define Column Headers* (on page 33)
  - *Define Export Mapping* (on page 47)
  - *Define Validation Rules and Rulesets* (on page 56)
  - *Define Implicit Delete Rules* (on page 64)
  - *Define Job Definitions* (on page 67)
  - *Use function column type mappings* (on page 71)
-

## SECTION 4


# Set Smart Data Validator Default Options

The following options can be set at the site level and will be inherited by all objects created in Smart Data Validator. The options can be changed, as needed, when configuring and mapping the objects required for Smart Data Validator Administration.

1. Click **System Options**  on the main Smart Data Validator window.
2. On the **Import** page, set the defaults for character encoding, illegal characters, validation of column header text and order, and the separator for multiple values. Each import mapping object created in Smart Data Validator Administration inherits these defaults, which you can keep or change, as necessary.

For more information on illegal characters, see *Set illegal characters for import mapping* (on page 25).

3. On the **Export** page, select default error severity settings in the following boxes:
  - **Block object on export minimum severity** - Set the minimum error severity level at which object export will be prevented. Select **Error** to block the export of objects that return validation errors, or **Warning** to block the export of objects that return validation warnings or errors.
  - **Block property on export minimum severity** - Set the default minimum severity level at which the property export will be prevented. Select **Error** to block the export of properties that return validation errors, or **Warning** to block the export of properties that return validation warnings or errors.

 **NOTE** This default value can be overridden on the stage property to target property map.

4. On the **Job Management** page:
  - Select the column set that will be used when displaying job information in SmartPlant Foundation Desktop Client. For more information about columns sets, refer to *Create column sets* in the *SmartPlant Foundation Desktop Client Help*.
  - Set the wait time to the number of days before a crashed job can be deleted from Smart Data Validator Job Management. The minimum wait time value is 1 and the maximum value is 30. For example, if you have created a job today, and set the wait time to 1 day, then you can delete the job anytime next day.
5. On the **Reports** page, specify the:
  - Maximum number of rows to be included in a validation report.
  - Default validation report definition, which determines the type of validation report generated: Excel, text, or a custom format.

 **NOTE** A default Excel format is delivered with Smart Data Validator.

6. On the set of **Validation** pages, select the default rules for the auto-generation of rules:
  - In the **Auto Generated Object Rules** page, select the default object rules to be automatically generated for each object action.

- In the **Auto Generated Property Rules** page, select the default property rules to be automatically generated for each property action.
- In the **Auto Generate Relationship Rules** page, select the default relationship rules to be automatically generated for each relationship action.
- In the **Auto Generate Relationship Property Rules** page, select the default relationship property rules to be automatically generated for each relationship property action.

★ **IMPORTANT** Before the relationship property rule is auto-generated, at least one relationship property mapping must be configured. One rule will be generated for all the properties of a relationship.

For more information, see *Actions* (on page 117).

📘 **NOTE** There is a naming convention for the validation rules that are automatically generated. For more information, see *Naming convention for rule auto-generation* (on page 63).

7. On the **Inconsistency Severity** page, select the default error severity for inconsistent property values, invalid action combinations, and conflicts between multiple actions. By default, inconsistency checks run for all jobs, but they can be disabled on the job definition.
  8. On the **Severity Level** page:
    - Choose the colors used for each severity level in the validation report. The defaults are red for errors and yellow for warnings.
    - Create additional severity levels or remove any unused levels.
  9. On the **Validation Defaults** page, select default severity triggers for the following boxes:
    - **Property error propagation minimum severity** - The minimum severity level at which the property error will be propagated to the object. Any error of higher severity, such as warnings, will also propagate up to the object.
- 📘 **NOTE** This default can be overridden on the stage property to target property mapping.
- **Missing mandatory and unique identifier generation columns error severity** – The default severity level for an object or property that comes from a mandatory column but has no a value.
  - **Missing UID columns error severity** – The default severity level when one of the properties for the UID creation has no value.
  - **Auto-generate rule error severity**- The default severity level for an auto-generated rule that fails validation. For more information about auto-generated rules, see *Auto-generate validation rules* in *Define Import Mapping* (on page 28).




## Set illegal characters for import mapping

### What illegal characters can I specify in import mappings?

When creating or updating import definitions, you can specify a set of illegal characters that cannot be used in imported data. Illegal characters can be any characters that you decide are illegal in your CSV file.

### What happens if illegal characters are found in an imported file?

If Smart Data Validator encounters any illegal characters during the import process, the import operation fails with an error indicating that illegal characters were found.

1. Click **Add**  next to the **Illegal Characters** box.
2. Click a text character or symbol, and click **Select Character** to add it to the **Illegal Characters** box.

 **TIP** Matching character types are grouped into subsets, such as **Currency Symbol**, which displays just the currency symbols. Click **Select Character Subset** to view all the subsets.

## SECTION 5

# Define Target Systems

A target system is a defined system or database that is the destination for validated data imported into Smart Data Validator. Target systems can be created in Smart Data Validator Administration with the URL address for the target system adapter. For instructions on how to setup a target system adapter, see *Smart Data Validator Installation and Setup Guide* and the *Smart Data Validator Customization Guide*.

When your target system is based on SmartPlant Foundation, you can use it to drive your import mapping, validation, and export mapping. Other target systems can be used just for export mapping.

**NOTE** When creating a target system, you must define the access groups that will have permission to select that target system during job creation.


Select **Target Systems** from the main toolbar list box to view a list of all the target systems configured for use in Smart Data Validator.

---

### What do you want to do?

- *Create a new target system* (on page 26)
  - *Edit a target system* (on page 27)
- 

## Create a new target system

1. From the main toolbar, click **Create New Target System**  to open the **Create New Target System** dialog box.
2. In the **Basic Details** page, enter the name and description of the new target system in the **Name** and **Description** boxes.
3. In the **Target System URL** box, enter a valid URL address location for the target system.
4. Click the **Supports Validation** option if you want the target system to support the validation of the data.
5. Click the **Implicit Delete** option if you want the target system to support the implicit deletion of data.

**NOTE** For more information on implicit delete functionality, see *Define Implicit Delete Rules* (on page 64).

6. Click the **Is Enabled** option to enable the target system to be used to create new job definitions.

★ **IMPORTANT** If the target system is disabled, jobs running using this target system are still allowed to complete.

7. Click **Next**.
8. In the **Access Groups** page, select an access group and click **Add** to add the access group to the **Selected Access Groups** list.


📘 **NOTE** Relating one or more of the access groups allows you to control who has permissions to create and edit the target system.

9. Click **Next**. View the **Summary**, and check the details.



📘 **NOTE** You can click any page heading to edit its properties, as required.

10. Click **Finish** to save the target system.

## Edit a target system

1. Select a target system from the **Target systems** list.
2. From the main toolbar, click **Edit Target System** .
3. In the **Edit Target System Definition** dialog box, click **Basic Details**. Edit the details for the target system as necessary, and click **Next**.

📘 **NOTE** If the target system is disabled, running jobs with this target system are still allowed to complete.

4. In the **Access Groups** page, select an access group and use the **Add**  or **Remove**  buttons to add or remove it from the **Selected Access Groups** list.
5. Click **Next**. View the **Summary**, and check the details.

📘 **NOTE** You can click any page heading to edit its properties, as required.

6. Click **Finish** to save the changes made to the target system.

## SECTION 6

# Define Import Mapping

### What is invariant format?

Invariant format is a property that does not change when a specified transformation is applied on a CSV file. For example, Smart Data Validator uses the invariant format for thousand separator “,” and for decimal separator “.”. So a value of 1.001 would always be interpreted as 1 with three decimal places even if your intention may have been 1001, as the decimal point was really a thousand separator. Smart Data Validator will not return any validation errors on properties using the invariant format, but you run the risk of not getting the intended data into the system.

### Import mapping process

Smart Data Validator uses import definitions to map imported CSV file data from existing column headers to the correct column headers for objects, properties, relationships, and relationship properties in the staging area. The import process uses the import definition as a template. The import mapping is built using the existing physical column headers in the input data file, but also supports other column headers that are computed or provided by prompt.

★ **IMPORTANT** CSV files must be supplied in invariant format. If you import a CSV file that is not in invariant format, you must use computed columns to create an import mapping that converts numeric and date information into invariant format. For more information, see *Supported CSV file formats* (on page 157).

## Import mapping

Select **Import Mapping** from the main toolbar list box to view, edit, or create import definitions. The details for import mappings are displayed in three panes:

- **Import Definitions** - This pane displays all the existing import definitions in Smart Data Validator. To select an import definition, click on it in this pane.
- **Column Headers** - This pane displays all the column headers that have been defined for a selected import definition.
- **Import Definition Properties** - This pane displays all the properties defined for each column header in the import definition. A list of all the object, property, relationship, and relationship property mappings is displayed when a specific column header is selected.

★ **IMPORTANT** All UID definitions (unique identifiers) in the staging area must match the UID definitions in the target system. UID definitions must be set on all objects that Smart Data Validator is exporting from the staging system to the target system.

When creating your mappings, you must ensure uniqueness in the Smart Data Validator import mapping by setting up the unique identifier on an object mapping. The system warns you if the number of columns selected to construct the unique identifiers does not match the number of elements in the target system UID definition.

📘 **NOTE** However, a mapping with an incorrect number of UID elements may not be incorrect. For example, if computed columns are concatenated, the UID warning can be safely ignored and the mapping saved with the different number of unique elements.

## Auto-generate export mapping

### When is auto-generation of export mapping recommended?

Always. The auto-generation of export mapping ensures that any errors are taken care of before exporting the data.

The export mappings in an import definition maps the imported objects, classes, and properties to the existing objects, classes, and properties in the target system. This mapping information can be manually created or automatically generated by selecting the **Auto Generate Export Mapping** option.

★ **IMPORTANT** When the **Auto Generate Export Mapping** option is enabled for an import definition, the target system selected is attached to the export mapping. The target system cannot be attached to another export mapping, as the import mapping objects are mapped to the target system's export mapping. A target system can have only one export mapping.

### NOTES

- When the **Auto Generate Export Mapping** option is selected, you cannot retrieve existing local mappings or use the free text option. The automatic generation of export mapping is available only when a target system is selected on the initial import definition form.
- When any import mapping is deleted that contains class definitions, relationship definitions, or property definitions that are no longer used by Smart Data Validator, those schema items are automatically removed from the database.

## Auto-generate validation rules

Validation rules are automatically generated when import mappings are defined, to save you time when creating rules. For example, each time you create an object mapping for **Check Exists**, Smart Data Validator automatically generates a rule. Validation automatically runs the rule to ensure the object exists in the target system.

By default, each action that can be assigned to a column is related to one or more generic rules. When validation rules are automatically generated, each generic rule is automatically configured for and applied to the column. For more information about the rules related to each action, see *Validation Rules and Actions* (on page 113).

---

### What do you want to do?

- *Create an import definition* (on page 30)
  - *Edit an import definition* (on page 31)
  - *Copy an import definition* (on page 31)
  - *View mapping summary* (on page 32)
-


## Create an import definition

### What is the Template File option in the Create New Import Definition dialog box?

The **Template File** option in the **Create New Import Definition** dialog box allows you to select a CSV file (if you have one) that can be used as a template or starting point. Column headers are created automatically based on the data in the first row of the CSV file.

### What is raw property format and when is it preferred over basic format?

A raw property format CSV file, or inverted CSV file, contains a smaller number of columns, and each property for an object is stored in a separate row. Whereas in a basic CSV file, each row contains properties and relationships for a single object. Raw property format is preferred when different objects have widely differing sets of properties.


1. To create an import definition, click **Create Import Definition**  on the **Import Definitions** toolbar.
2. In the **Create New Import Definition** dialog box, type a name and description for the import definition.
 

**NOTE** You cannot create an import definition with the same name as an existing import definition.
3. Choose the **Template File** type by selecting **None** or **CSV**. For more information on CSV file formats, see *Supported CSV file formats* (on page 157).
 

**NOTE** When you click the **CSV** option, the dialog expands so you can browse to a CSV template file that has column headers set up to match the target system. Click **Open**.
4. Select the character encoding to be used from the **Character Encoding** box. For example, **ISO-8859-15**.
5. Using the **Illegal Characters** box, define characters that cannot be used in the imported data. For more information, see *Set illegal characters for import mapping* (on page 25).
6. Select the **Enforce Column Text** option to ensure that the column names exactly match the column names used for creating the Import Definition. The column names in CSV file that is used to create a job must match the Import Definition column headers or the column headers of the CSV that is used to create the Import Definition.
7. Select the **Enforce Column Order** option to use the exact column order as the CSV file that is used for creating the Import Definition. During the execution of the job, the system checks if the order of the column headers is same as the Import Definition or same as the CSV file that is used to create the Import Definition.
8. From the **Target System** box, select the target system for the new import definition. Only target systems that support validation, such as SmartPlant Foundation-based target systems, appear in this list.
9. Select the **Auto Generate Export Mapping** option if you want to automatically generate the export mappings so that the column headers match the objects and properties found in the target system.
10. Select the **Auto Generate Validation Rules** option if you want to automatically generate the validation rules from the mapping set up in the target system. The rules are generated in SmartPlant Foundation Desktop Client. For more information, see *Configure the automatic generation of validation rules* (on page 59).



11. Choose the import definition format from the **Mapping Format** box.
  - **Basic Format** - keeps the standard **Import Mapping** dialog layout.
  - **Raw Property Format** - Creates a CSV raw attribute mapping format dialog layout. For more information, see *Define a raw attribute map* (on page 42).
12. Click **Save**.

## Edit an import definition


1. Select an import definition from the list in the **Import Definitions** pane, click **Edit Import Definition** .
2. In the **Edit Import Definition** dialog box, modify the properties for the selected import definition, as needed. For more information on changing these parameters, see *Create an import definition* (on page 30).

### NOTES


- The **Name** box in the **Edit Import Definition** dialog box is editable. However, you cannot edit an import definition name to be the same name as an existing import definition, because the UID is not updated when the name of an import definition is edited.
  - You cannot change the target system or the mapping format once the import definition is created.
3. Click **Save**.

 **TIP** To delete an existing import definition, including any associated column headers and their mappings, select it and click **Delete**  on the **Import Definitions** toolbar.


## Copy an import definition

1. From the **Import Definitions** list, select an import definition, and click **Copy Import Definition**  in the **Import Definition** toolbar.
2. In the **New Import Definition Name** dialog box, type a unique name for the new import definition.
3. Click **Save**.

## View mapping summary

- To view a mapping summary diagram of the selected import definition, click **View Mapping Summary** .

A summary of the import definition is displayed in the **Mapping Summary** dialog box. These relationships can be simple or complex, depending on the mapping you create for the import definition, and can show all the properties.

 **TIP** You can zoom in on the image, resize and move the elements, fit the image to the page, or print the image by using the standard commands on the **Mapping Summary** dialog box.



## SECTION 7

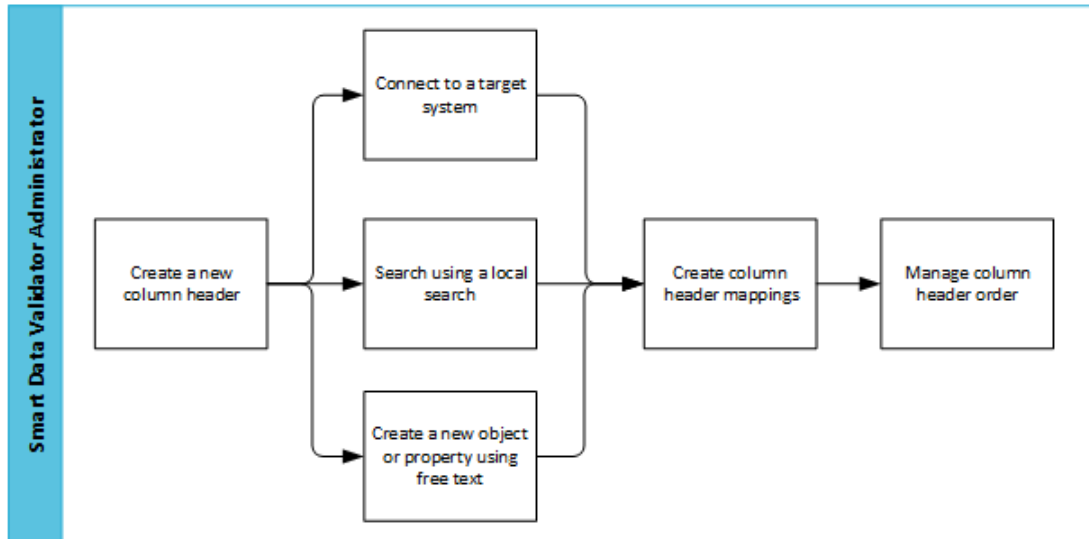
# Define Column Headers

You can define the column headers for each import definition to map the column data from imported CSV files to the existing columns in a target system. The mapping can be driven by the target system database and defined manually using the free text option, or you can use existing mapping in the local staging area. All existing column headers for a selected import definition are displayed in the **Column Headers** pane.

You can create, edit, and modify each column header mapping to match the class definitions, objects, properties, relationships, and relationship properties between the imported data objects and target system objects.

**NOTE** If you selected **Raw Property Format** as the mapping format when creating the import definition, a separate **Raw Attribute Map** pane displays under the **Column Headers** pane. Initially, the Raw Attribute Map fields are empty and only show the column headers after at least one column has been mapped to an object; matching the CSV template file selected in the import definition. The Raw Attribute Map contains a read-only representation of the CSV raw attribute mappings. For information on defining the raw attribute mappings, see *Define a raw attribute map* (on page 42).

Smart Data Validator Administrator



### What do you want to do?

- *Create a new column header* (on page 35)
- *Connect to a target system* (on page 36)
- *Search using a local search* (on page 36)
- *Create a new object or property using free text* (on page 37)

- *Create column header mappings* (on page 37)
  - *Manage column header order* (on page 42)
  - *Define a raw attribute map* (on page 42)
  - *Generate raw attribute property mappings for export* (on page 44)
- 

## Column types

Column headers are used to map the data in the CSV file to the correct locations in the staging area and target systems, and supports a number of different column types. The column type you select when defining the column header determines the appearance of the **Column Header** and **Mapping** dialog boxes that you use later in the import mapping process. Smart Data Validator supports the following types of columns:

- **Physical** - Maps to a specific column in the input data source.
- **Computed** - Generates a new column using data from other columns or sources, such as environment type variables. This provides the ability to use functions. For example, you can enter a function in the **Computed API** box to replace original entries with new entries.

For more information, see *Use the Computed column* (on page 71).

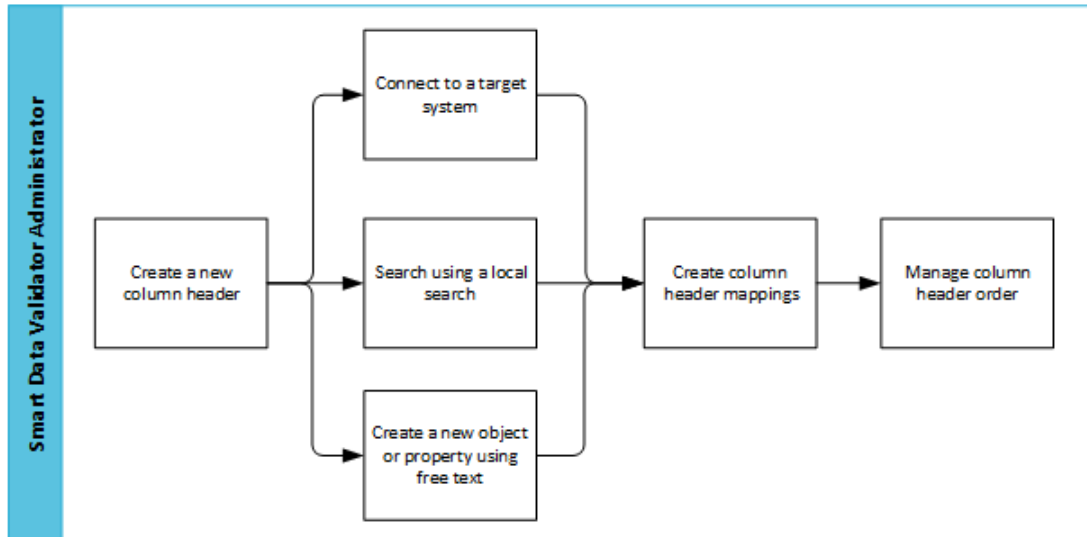
- **Constant** - Allows you to use a string constant.
- **Prompted** - Prompts the user for a value when a job is run, based on the job definition. The provided value is stored with the job definition for later use, if necessary.
- **Prompted API** - Generates a list of prompts generated from other columns or sources, such as environment type variables, and displays this list using functions.

For more information, see *Use the Prompted API column* (on page 75).

- **Prompted Picklist** - Allows the user to pick an entry from a defined enumerated list in the local staging area when running a job.


## Create a new column header

Smart Data Validator Administrator



**How do I decide whether to select Local, Target System, or Free Text option when creating a new mapping for a column header?**

The **Local**, **Target System**, and **Free Text** options are available in the **Create New Column Header** dialog box to create new mappings.

- Use the **Local** option to search for an object, property, or relationship locally in the Smart Plant Foundation system and add it to a new mapping.
  - Use the **Target System** option to connect to an existing target system and add a new mapping.
  - Use the **Free Text** option to create a new object or property that does not exist in the target system and create the mapping.
1. Click **Create New Column Header**  from the **Column Headers** pane toolbar to open the **Create New Column Header** dialog box.
  2. In the **Column header** pane, select the column type required from the **Column Type** list box. The options on this dialog box differ for the column type selected. For more information, see *Column types* (on page 34).
  3. Provide a name and any other details required for the selected column header type, such as in the **Column Header Text** box.
  4. Select the **Multiple Values Column** checkbox if any column in your CSV file contains multiple values.

For example, if a tag has to be related to many secondary disciplines (such as MECH, ELEC, CIVIL), you can separate the values in the CSV file using a comma, and then select this checkbox when creating the column header.

**NOTE** The **Multiple Values Column** option is supported for **Physical** and **Computed Column** types only.

5. From the **Add New Mapping** options, choose whether to map the new column header to an object, property, relationship, or relationship property.

**NOTE** If you do not want to add prompted values for the **Prompted API**, **Prompted**, and **Prompted Picklist** column types, select the **Value for this column can be blank** option. A prompted value is then not needed when you create a job in Smart Data Validator Job Management.



## Connect to a target system

After you specify the column header for the data from the data file, you must map the column to an object, property, relationship, or relationship property in the staging area or target system. To find an object, property, relationship, or relationship property in the target system to map to, you must connect to that target system and search for it.

**NOTE** When you create an import definition using the **Auto Generate Export Mapping** option, click **Connect To target System** and the objects and properties found in the target system are automatically imported into the Smart Data Validator staging database to help you create mappings for the new column headers.

1. Click **Connect to Target System** to connect to the selected target system.



**TIP** If there are multiple target systems associated with the import definition, choose the one for mapping from the list, and click **Connect**.

2. Click **Search**  to find the object, property, relationship, or relationship property in the target system, and get a list of all the matching items.
3. Choose the target system object, property, relationship, or relationship property that you want to use, and click  to create the mapping.

For more information on creating column header mappings, see *Create column header mappings* (on page 37).


## Search using a local search

To search for an object, property, or relationship from SmartPlant Foundation and add to a new mapping:

1. Click **Local** to search for objects and properties found locally in the SmartPlant Foundation system.
2. Click **Search**  to find the object, property, relationship, or relationship property in the target system, and get a list of all the matching items.
3. Choose the target system object, property, relationship, or relationship property that you want to use, and click  to create the mapping.

## Create a new object or property using free text

You can create a new object or property that does not exist in the target system database using free text:

1. Click **Free Text**, and type the name of the new object or property in the text box.
2. Click **Add**  to create the mapping.

## Create column header mappings

You must create column mappings for each column header in the import definition so that imported data matches the corresponding data column format in the staging area or target system. You can search for the objects and properties found locally or add new objects and properties using free text, then create the new mappings to match the target system objects and properties.

There are four types of column header mappings you can create for each column header.



- *Object mapping* (on page 37)
- *Property mapping* (on page 39)
- *Relationship mapping* (on page 39)
- *Relationship property mapping* (on page 40)


### NOTES

- When a column mapping has been added successfully, a check mark displays in the **Has Mapping** column next to the column header mapping in the main window. The options available for the column mapping vary based on the type of column header mapping used.
- The object, property, relationship, and relationship property mapping names can include underscore and alphanumeric characters.


## Object mapping

A column header can only have a single object mapping, but can have multiple mappings of different types. The following is an example of creating the object mapping **Maps to Object PPMTTag with action Create Update**.

1. In the **Mappings** pane, select the **Object Mapping** option in **Add New Mapping**.
2. Select the **Local** tab at the bottom of the **Mappings** pane.
3. Type **\*Tag\*** in the box and click **Search**  to search for all the objects in the local staging area that have Tag in the name.
4. Select the **PPMTTag** object from the results in the list box, and click **Add** . The **Maps to Object PPMTTag with action Create Update** mapping is added to the column header.
5. If necessary, change the **Object Action** to another selection. For a list of object actions, see *Actions* (on page 117).

6. To add unique identifiers (UIDS) that match the existing object, such as **Tag**, **Plant Code**, or **Tag Name**, click **Create Unique Identifier** .

#### **NOTES**

- The unique identifiers (UIDs) must be set on all objects that Smart Data Validator exports from the staging area. You must set the correct UID in the import mapping by setting up the UID definitions for any object mapping.
  - The **Unique Key definition** and **Unique ID Definition** boxes are populated with values that are configured for the class definition currently being mapped to. Once they are propagated, they are not updated again and must be updated manually by loading the class XML file with the new Unique ID Definition and Unique Key Definition.
7. To add optional interfaces for the selected class definition, click **Select Optional Interfaces** . This option is available for object mapping with **Create Update** and **Update** actions.

#### **NOTES**



- The **Select Optional Interfaces** option allows you to select optional interfaces that are available for the selected class definition in the target system. The selected optional interfaces are instantiated on the objects during export.
  - If you do object mapping with the document revision class definition, the **Select Optional Interfaces** dialog box displays the optional interfaces for both the revision and version. If the selected optional interface is available for both the revision and version class definitions, during export the optional interface gets instantiated for both the revision and version objects.
8. Add a **Prefix** for the file type if required, for example **TG**.
9. Select the **Mandatory Object** option to indicate that the column is required to have a value.
10. Click **Save**.


#### **NOTES**

- When any cell used to create the unique identifier is blank, an error is included in the validation report indicating which cell or cells are missing values.
- If an object in the target system already exists in the staging area then Smart Data Validator will ask you to rename the object when saving.
- If the **Mandatory Object** option is selected and the column has no value, an error is included in the validation report.

## Property mapping

The following is an example for creating the property mapping **Maps to Property PPMTagStatus with action Create Update**.

1. In the **Mappings** pane, click the **Property Mapping** option for **Add New Mapping**.
2. Select the **Target System** tab at the bottom of the **Mappings** pane.
3. Type \* in the box, and click **Search**  to search for all properties related to PPMTag.
4. Select **PPMTagStatus** from the results in the list box, and click **Add** . The **Maps to Property PPMTagStatus with action Create Update** mapping is added to the column header.
5. If necessary, change the **Property Action** to another selection. For a list of property actions, see *Actions* (on page 117).
6. In the **Parent Objects** box, select the parent objects required.

 **TIP** You can select each one individually or you can use the **Select All** or **Unselect All** options.



7. Click **Save**.

### NOTES


- When you create a property mapping without an object mapping, you can select an object column from the **Select Object Mapping to Filter Properties** list on the **Target System** tab to view the properties related to the selected object. If you do not select an object column, the entire target schema is searched.
- When you create a property mapping with an object mapped to the column header, your property search displays properties only for that object. If you do a property mapping with the document revision object, then your property search not only displays the properties for the revision but also displays the properties of the version.
- Using Smart Data Validator, you cannot delete or terminate the required properties of an object that exists in the target system.

## Relationship mapping

The following is an example of relationship mapping for **Maps to Relationship PPMTagClassHasPropertyDef with action Check Exists** mapping.



1. In the **Mappings** pane, select the **Relationship Mapping** option for **Add New Mapping**.
2. Select the **Target System** tab at the bottom of the **Mapping** pane.
3. Type **\*PPMTag\*** in the box, and click **Search**  to search for all relationships that have PPMTag in the name.
4. Select **PPMTagClassHasPropertyDef** from the results in the list box, and click **Add** . The **Maps to Relationship PPMTagClassHasPropertyDef with action Check Exists** mapping is added to the column header.
5. If necessary, change the **Relationship Action** to another selection. For a list of relationship actions, see *Actions* (on page 117).

6. In the **Column For Relationship End 1** box, select the end 1 of the relationship, for example, select **TagStatus**.
7. In the **Column For Relationship End 2** box, select the end 2 of the relationship, for example **PPMTagStatus**.
8. Click **Save**.

 **NOTE** Only one end of the relationship mapping can have a multiple values column.

## Relationship property mapping

The following example shows the relationship property mapping for **Maps to Relationship Property SPFFilterByOwnerOrOwningGroup** with the **Create Update** action.

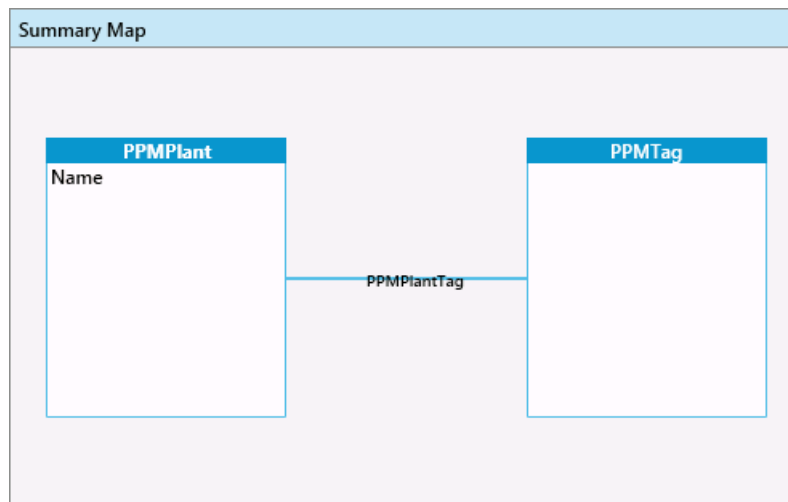
1. In the **Mappings** pane, select **Relationship Property Mapping > Add New Mapping**.
2. Type **\*SPFFilter\*** in the box, and click **Search** .
3. Select **SPFFilterByOwnerOrOwningGroup** in the list, and click **Add** .

The **Maps to Relationship Property SPFFilterByOwnerOrOwningGroup** with action **Create Update** mapping is added to the column header.

4. In the **Parent Relationships** box, select the parent relationships required.
5. Click **Save**.

## Summary map

When you have finished creating each mapping, the **Summary Map** pane displays a graphical representation of the column header mapping. For example, the following image shows the **PPMPlantTag** relationship mapping created between the two object classes, **PPMPlant** and **PPMTag**.










## Edit column headers

**When can I safely ignore the system warning that UID definitions in the staging system do not match those in the target system?**

You can safely ignore the system warning that UID definitions in the staging system do not match those in the target system when you have used computed columns with concatenation. This is because the number of unique elements in the staging and target systems may not match due to the concatenation.

1. Select a column header to edit, and click **Edit Column Header**  in the **Column Headers** toolbar.
2. In the **Edit Column Header** dialog box, update the column type, name, and other available properties, based on the column type.
3. In the **Mappings** pane, click **Expand**  on a specific mapping to edit the details of each of the object, property, relationship, and relationship property mappings for the column header.

### TIPS


- If necessary, you can add a new property or relationship mapping to the column header. For more information on how to update the mapping, see *Create column header mappings* (on page 37).
- Only some of the property values can be edited on existing mappings, as the object cannot be changed once a column header is created.
- Click **Delete**  beside **Expand**  to remove a mapping from the selected column header.
- You can select a column header, and click **Delete**  on the **Column Headers** toolbar to delete the column header and all mappings associated with it.

### NOTES

- A blue check mark next to a column header indicates that it is a component of the raw attribute map. For more information on defining a raw attribute map, see *Define a raw attribute map* (on page 42).
- If the column header has been defined as a component of the raw attribute map, the edit functionality is limited to only the column header type and its applicable properties.
- If you need to edit a column header mapping to a raw attribute map, you must remove the column header type from the existing raw attribute map.
- If you change the class map action on the Parent Object column, a warning displays that you must change the action on the raw attribute map to be compatible with the new class map action.

## Manage column header order

All the columns created in the import definition need to be arranged into the order required to match the existing column headers in the selected target system. Use the following procedure to rearrange the order of the existing column headers:

1. From the **Import Definitions** toolbar, click **View Column Header Management** .
2. Select a column header in the list and promote or demote it using the arrows.
3. Click **Save** when you are finished arranging the column headers.








## Define a raw attribute map

The data in a standard CSV file is typically organized in horizontal rows, where each row contains properties and relationships for the single object. However, it is possible to organize the data as an inverted CSV file.

The inverted CSV file, or CSV raw attribute format file, contains a smaller number of columns, and each property and attribute for an object is on in a separate row. In the case of a CSV raw attribute format file, the first column or columns contain data to identify the object, such as the area, for example, PLANT\_CODE, and the name of the object, TAG\_NAME.

The first column header can be defined to identify the object's attributes in your CSV template file. For information on defining the column headers, see *Create column header mappings* (on page 37).

In the following example, the last three column header mappings for the import definition match the **Property\_Name**, **Property\_Value**, and **Property\_Value\_UOM** in the raw attribute map.

Column Headers 			
  			
Order Value	Name	Column Type	Has Mapping
1	PLANT_CODE	Physical	✓
2	TAG_NAME	Physical	✓
3	PROPERTY_NAME	Physical	
4	PROPERTY_VALUE	Physical	
5	PROPERTY_VALUE_UOM	Physical	

These column header mappings can be predefined in an imported CSV template XML file that you select when creating the import definition for CSV raw attribute mapping. However, these values are set as Property\_Name, Property\_Value, and Property\_Value\_UOM, but you can use different values for the column headers in the import definition. To define the object properties for these columns, see *Create column header mappings* (on page 37).

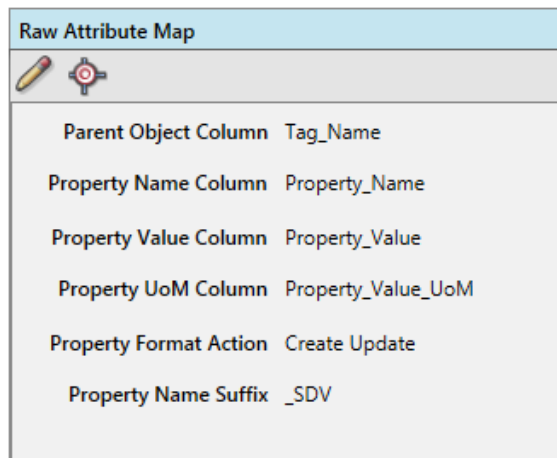
**NOTE** When the CSV raw attribute mappings are complete and the raw attribute map column header has been mapped to the object property, the column is identified by a blue dot in the **Column Headers** pane.

## Raw Attribute Map

When you have created an import definition using the **Raw Property Format** mapping format, the **Raw Attribute Map** dialog box displays under the **Column Headers** pane.

The **Raw Attribute Map** dialog box displays an editable list of CSV raw attribute mappings for inverted format files. The column headers are:

- **Parent Object Column** - Mandatory. Any column with a class map
- **Property Name Column** - Mandatory. Any column without mappings
- **Property Value Column** - Mandatory. Any column without mappings
- **Property UOM Column** - Any column without mappings
- **Property Format Action** - Any of the property actions available for the Class map action on the Parent Object Column. This default value is based on the selected Parent Object Column, for example **Create Update**.
- **Property Name Suffix** - When staging and target systems are the same, you must specify a suffix to avoid conflicts. For example, you can specify **\_SDV** as a suffix.




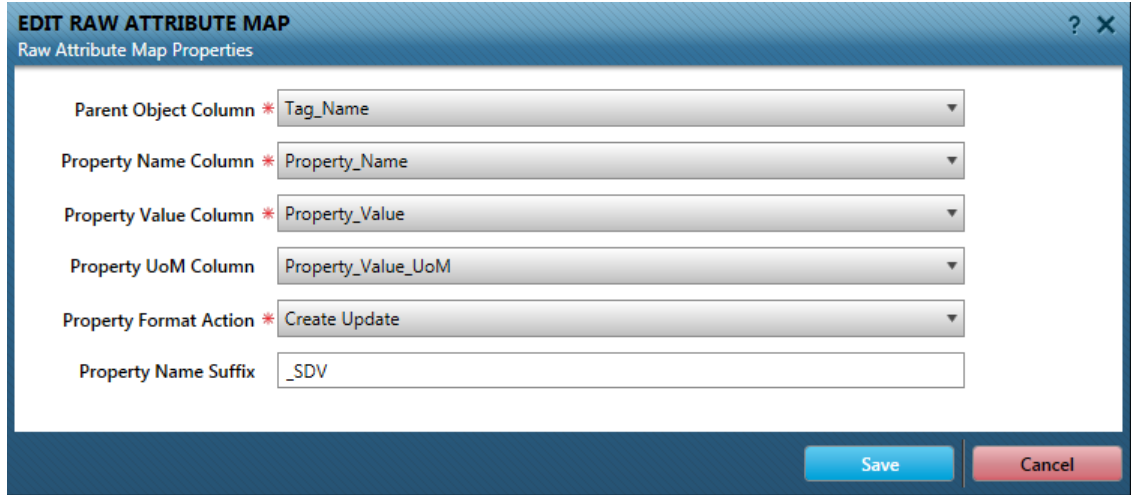
Raw Attribute Map	
Parent Object Column	Tag_Name
Property Name Column	Property_Name
Property Value Column	Property_Value
Property UoM Column	Property_Value_UoM
Property Format Action	Create Update
Property Name Suffix	_SDV

### NOTES

- The Column headers can be defined to identify the object attributes in your CSV template file. For information on editing the column headers, see *Edit a raw attribute map* (on page 44) and *Create column header mappings* (on page 37). To auto generate the raw attribute properties for export mapping, see *Generate raw attribute property mappings for export* (on page 44).
- The property name suffix must be specified only when the staging and target systems are the same.

## Edit a raw attribute map

You use the **Edit Raw Attribute Map** dialog box to define CSV raw attribute mappings. To open the **Edit Raw Attribute Map** dialog box, click **Edit**  on the **Raw Attribute Map** pane.



### NOTES

- The property values in the Raw Attribute Map form must match the column headers as defined in the import definition CSV template file.
- The Property\_Name Column, Property\_Value Column, and Property\_UOM Column, inherit their mapping property values from the Parent Object Column.
- The Property Format Action drop-down menu is defined by the Parent Object Column. The action selected matches the same action as selected by the Parent Object Column mapping.
- All of the mandatory fields are indicated by the red asterisk.
- A blue dot in the **Column Headers** pane indicates that the CSV raw attribute mappings are complete.
- You cannot delete a column header that contains a raw attribute property. You must first deleted the property from the raw attribute map before deleting the column header.

## Generate raw attribute property mappings for export


After you have created a raw attribute map in the **Raw Attribute Map** dialog box, you can generate raw attribute property mappings to export the raw attribute properties from an existing CSV inverted format file to a target system or by creating computed columns for properties.

The **Ignore Property** allows you to ignore the desired properties being loaded into the staging system when importing the data from an existing inverted CSV file. However, you can import the value for the selected property from the inverted CSV file to the staging system by creating a computed column for this property. For more information, refer to *Use the Computed column* (on page 71).

When creating computed columns for the properties, apply the **InvertedFilter** function to check that the property name in the inverted CSV file matches the property name in the computed

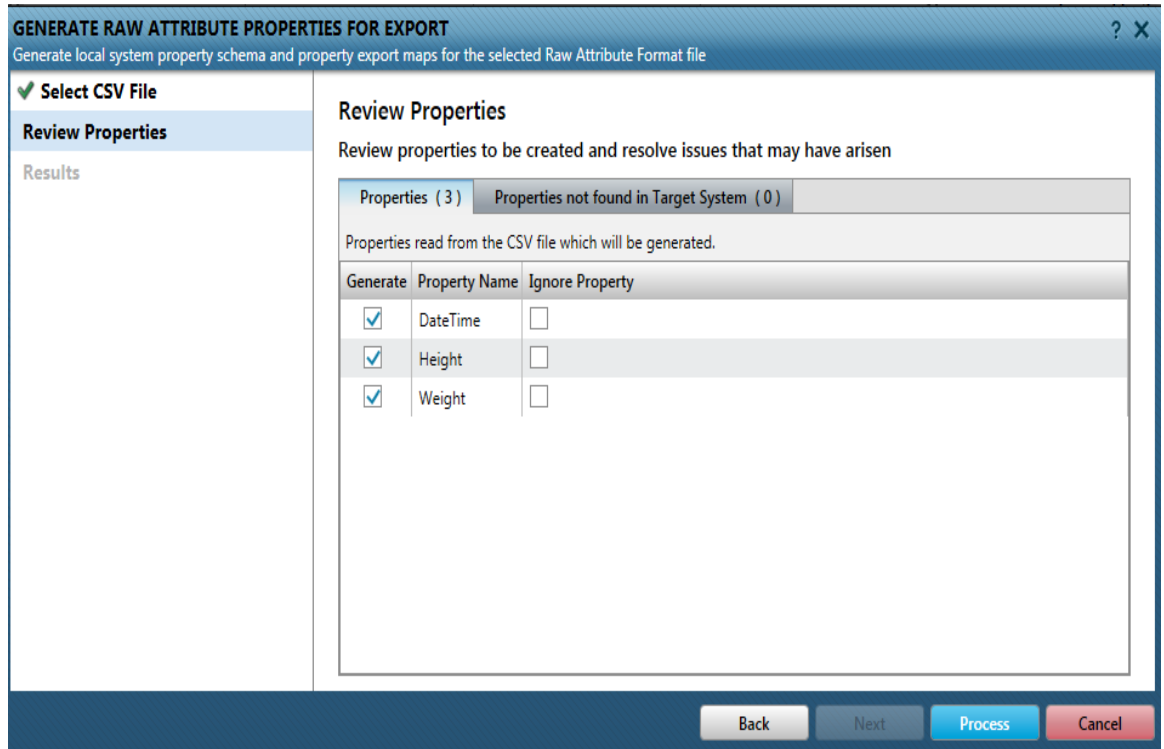
column. The computed column containing the property is evaluated, and the property value is then imported into the staging system.

To generate the raw attribute property mappings for export:

1. Click **Generate Raw Attribute Properties for Export**  on the **Raw Attribute Map** dialog box located under the **Column Headers** pane in **Import Mapping**. The **Generate Raw Attribute Properties for Export** wizard opens.
2. In the **Select CSV File** pane, browse to the CSV inverted format file that contains the raw attribute property names. The path to the CSV file is displayed in the **CSV File** box, and the number of properties read from the CSV file are shown.

**NOTE** Properties read from the CSV file have all illegal characters removed automatically on selection, such as space, (, ), and /. These are excluded due to property name constraints in SmartPlant Foundation.

3. Click **Next** to open the **Review Properties** pane, where you can review the property mappings to be created in two tabs:
  - **Properties** - Lists the number of properties read from the CSV file that generates a raw attribute property mapping.
  - **Properties not found in Target System** - Lists the number of properties read from the CSV file that do not have a raw attribute property mapping and are not found in the target system.
4. On the **Properties** tab, select the **Ignore Property** check box for the properties you want to ignore when importing the data from an existing CSV file.



The screenshot shows the 'GENERATE RAW ATTRIBUTE PROPERTIES FOR EXPORT' wizard. The left pane has 'Select CSV File' and 'Review Properties' (selected). The right pane is titled 'Review Properties' and contains a table of properties to be generated.


Properties ( 3 )			Properties not found in Target System ( 0 )		
Properties read from the CSV file which will be generated.					
Generate	Property Name	Ignore Property			
<input checked="" type="checkbox"/>	DateTime	<input type="checkbox"/>			
<input checked="" type="checkbox"/>	Height	<input type="checkbox"/>			
<input checked="" type="checkbox"/>	Weight	<input type="checkbox"/>			

At the bottom of the wizard are buttons for 'Back', 'Next', 'Process', and 'Cancel'.

5. On the **Properties not found in the Target System** tab, type the name of the target system property to map to the raw attribute property in the box in the **Name of Property in Target System to map to** column.
6. When all the target system properties have been entered, click **Process**.

#### **NOTES**

- A warning message displays if any entered property names do not exist in the target system. You can modify the existing entry, or continue to create the property mapping. If you continue to create a property mapping with a target system property that does not exist, the raw attribute property mapping is still created. However, the property must be created in the target system to use the export functionality.
  - If a target system property name contains any illegal characters, a warning message is displayed when you click **Process**. You cannot proceed without removing the illegal characters.
7. In the **Results** pane, each of the raw attribute property mappings for export generated is displayed with **Success** in the **Result Details** column and a green checkmark in the **Has Created** column. Click **Close**.

 **NOTE** You can check that the raw attribute property mappings for export have been created in **Export Mapping**. From there you can also modify or update them as required. For more information, see *Stage property to target property mappings* (on page 51).

### InvertedFilter

This function checks for the property name in the inverted CSV file and if it matches the property name configured in the computed column, the property values are displayed in the output based on the functions defined in the computed column.

#### **Problem**

You can use the Computed API for columns in Inverted CSV formatted file, but duplicate entries are created in the staging system.

#### **Solution**

Create a computed column for a property by applying the **InvertedFilter** function to check if the property name in the inverted CSV file matches the property name configured in the computed column. If they match, the property values are displayed in the output based on the functions defined in the computed column.

#### **Example**

You want to convert the input value dd-MM-yyyy in an inverted CSV file with a DateTime column to an invariant date and time format "yyyy/MM/dd-HH:mm:ss:fff". Type the function as follows:

```
func.InvertedFilter ([PROPERTY_NAME], {"DateTime",
func.DateTimeColumn([PROPERTY_VALUE], "dd-MM-yyyy", "UTC")})
```

The above computed API function checks for the property name "DateTime" in the inverted CSV file. If it matches the column name defined in the computed API, it is displayed as the DateTime column value yyyy/MM/dd-HH:mm:ss:fff.

## SECTION 8

# Define Export Mapping

You can export data to a target system by mapping the imported staging area data objects to the existing objects, properties, relationships, and relationship properties found in the target system. You can manually configure the export mapping for each object in the imported data, or you can automatically generate the export mapping from the target system by selecting the **Auto Generate Export Mapping** option when you create a new import definition. The export mapping can be further refined for each stage object to target object mapping by selecting specific property-to-property, relationship-to-property, relationship-to-relationship, and relationship property-to-relationship property mappings.

**TIP** If you have two target systems that use the same model or schema, they can share the same export mapping.

You can view, edit, or create export mappings from the main toolbar list box by selecting **Export Mapping**. The details of the export mappings are displayed in two panes.

- **Export Mapping** - This pane details all of the existing export mappings as a list with three columns, **Name**, **Description**, and **Target System**. To open the details of a stage to target object map, select an export mapping in the **Export Mapping** pane.

**TIP** The **Description** column is automatically set to **Auto generated** if the **Auto Generate Export Mapping** option is selected for the import definition.

- **Stage to Target Object Map** - This pane displays the details for a selected stage to target object mapping in three columns; the stage object, the target object to which it is mapped, and the object weight.

You can access and refine the properties of a stage object in the list by selecting one of the property-to-property, property-to-relationship, relationship-to-relationship, or relationship property-to-relationship property mapping commands located on the **Stage Object To Target Object Map** toolbar. For more information, see *Define stage object to target object mappings* (on page 49).

### NOTES

- When mapping, you can only use **Check File Exists** or **Check Target Exists** as an action on an object, you cannot use both.
- When any export mapping is deleted that contains class definitions, property definitions, relationship definitions, or relationship property definitions that are no longer used by Smart Data Validator, those schema items are automatically removed from the database.
- The object, property, relationship, and relationship property mapping names can include underscore and alphanumeric characters.


---

### What do you want to do?


- *Create an export mapping* (on page 48)
- *Edit an export mapping* (on page 48)
- *Copy an export mapping* (on page 48)

- *Define stage properties and relationships to target object mappings* (on page 50)
- 


## Create an export mapping

1. Click **Create Export Mapping**  on the **Export Mapping** toolbar.
2. In the **Create New Export Map** dialog box, provide a name and description for the new export mapping.  
**NOTE** If the export mapping was automatically generated by the import definition, **Auto Generated** appears in the **Description** box.
3. Select the target system for the export mapping from the **Target Systems** list.
4. Click **Save**.


## Edit an export mapping

1. In **Export Mapping** pane, select an export mapping from the list.
2. In **Edit Export Mapping** toolbar, click **Edit Export Mapping** .
3. In **Edit Export Map** dialog box, modify the details for the selected export mapping as required.
4. Click **Save**.

### NOTES

- The **Edit Export Map** dialog box **Name** field is not editable due to a technical restriction.
- You can delete an export mapping, the stage object to target object map, and all related property and relationship mappings associated with it, by selecting an export mapping, and click **Delete Export Mapping**  on the **Export Mapping** toolbar.

## Copy an export mapping

1. In the **Export Mapping** pane, select an **Export Mapping** from the list.
2. Click **Copy Export Mapping**  on the **Export Mapping** toolbar.
3. In the **Name** and **Description** boxes, provide the new name and description for the export mapping being created by the copy operation.
4. In the **Target System** list, select a target system for the new export mapping from the list.  
**NOTE** A new export mapping cannot use the same target system as an existing export mapping.
5. Click **Save** to create the new export mapping and display it in the **Export Mapping** list.



## Define stage object to target object mappings

The **Stage to Target Object Map** pane displays the names of all the objects stored in the staging area of Smart Data Validator and the name of the target objects that the stage objects are mapped to in the target system.

The **Object Weight** value controls the order of export of the object or error report. The lower the number, the earlier in the process the object is exported or reported. For example, a class can exist without a tag, but a tag cannot exist without a class, so class must have a higher priority and therefore, a lower object weight.

### NOTES




- You can filter for the data in each column to limit the objects to a manageable list, such as for only those objects that have the highest priority.
- You must specify the object weighting for each **Stage to Target Object Map**. For example, the object weightings must be set to export the revision before exporting the file to which it is attached.


---

### What do you want to do?

- *Create a stage object to target object mapping* (on page 49)
  - *Edit a stage object to target object mapping* (on page 50)
  - *Define stage properties and relationships to target object mappings* (on page 50)
- 

## Create a stage object to target object mapping

1. Click **Create Stage Object To Target Object Mapping**  on the **Stage To Target Object Map** toolbar.
2. In the **Stage Object** section of the **Create Stage Object To Target Object Map** dialog box, click **Search**  to find the local object that you want to map to a target object.
3. Select an object from the search results to display it in the **Selected stage object** box.
4. In the **Target Object** section, select the target system to which you want to connect from the list, and click **Connect**.
5. In the **Target System** search box, click **Search**  to get a list of all the target system objects to map to the stage object.


 **NOTE** If the target object does not exist in the target system, you can enter a name for it in the **Free Text** box.


6. Select the target object from the results to display it in the **Selected target object** box.
7. In the **Details** pane, provide a description for the object mapping in the **Description** box.

8. In the **Object Weighting** box, provide a value for the object weighting to set the order for the object mapping. Objects with lower weights are processed first.
9. In the **Target System Unique Key** box, type the unique key of the target object.
10. In the **Target System Query Definition** box, type the query language to find the target object.
11. Click **Save**.

**NOTE** The software uses the UID to find and identify objects in the target system when running validation rules and exporting data. If the object cannot be found using the UID, you can search for the object by the unique key entered. If the software cannot find the target system object using the UID or the unique key, you can search for the object using the query language entered. For more information on query definitions, see *Unique keys and query definitions* (on page 110).

## Edit a stage object to target object mapping

1. To edit an existing stage object to target object map, select it and click **Edit Stage to Target Object Map**  on the **Stage to Target Object Map** toolbar.
2. In the **Edit Stage to Target Object Map** dialog box, update the details as necessary. If required, change the stage or target objects that are mapped.
3. Click **Save**.

**NOTE** Click **Delete**  to delete a stage object to target object mapping, as well as any related stage property to target property mappings, stage relationship to target property mappings, stage relationship to target relationship mappings, or stage relationship property to target relationship property mappings.

## Define stage properties and relationships to target object mappings


The stage to target object properties and relationships can be further defined as mappings for each object found in the stage object to target object map. There are four mapping types that can be created.

- *Stage property to target property mappings* (on page 51)
- *Stage relationship to target property mapping* (on page 52)
- *Stage relationship to target relationship mappings* (on page 53)
- *Stage relationship property to target relationship property mappings* (on page 54)




## Stage property to target property mappings

The **Stage Property To Target Property Mappings** pane displays the names of the stage properties mapped to the target properties in the target system database.

Use the following steps to view the stage property to target property mappings:

1. In the **Stage To Target Object Map** pane, select a stage object in the **Stage Object Name** column.
2. Click **Show Stage Property To Target Property Mappings**  to display the stage property to target property list.

### Create a stage property to target property mapping

1. Click **Create Stage Property To Target Property Mapping**  on the **Stage Property To Target Property Mappings** toolbar.
2. In the **Stage Property** section of the **Create Stage Property To Target Property Mapping** dialog box, click **Search**  to find the local property that you want to map to a target property.
3. Select a property from the results in the **Selected stage property** box.
4. In the **Target Property** section, select the target system from the list, and click **Connect**.
5. In the **Target System** search box, click **Search**  to locate the target system property to map to the stage property.

**NOTE** If the target property does not exist in the target system, you can enter a name for it in the **Free Text** box.


6. Selected a target property from the results as it displays in the **Selected target property** box.
7. In the **Details** pane, provide the description for the mapping in the **Description** box.
8. In the **Block Property on Export Minimum Severity** list, select the severity threshold at which the property will be blocked from exporting if it fails a validation rule. Any severity setting greater than this will also propagate; if the setting is warning, then both warnings and errors propagate up to the object.
9. In the **Property Error Propagation Minimum Severity** list, select the severity threshold at which the failure of the property will be propagated up to the object and prevent it from being exported. Any severity setting greater than this will cause the property to be blocked; if the setting was warning, then both warnings and errors propagate up to the object.

**TIP** For more information about the severity level assigned to validation rules, see *Define Validation Rules and Rulesets* (on page 56).




10. Click **Save**.

## Stage relationship to target property mapping

The **Stage Relationship To Target Property Mappings** pane displays the stage relationships that are mapped to the target properties in the target system database.

1. In the **Stage To Target Object Map** pane, select a stage object in the **Stage Object Name** column.
2. Click **Show Stage Relationship To Target Property Mappings**  to display the stage relationship to target property list.

### *Create a stage relationship to target property mapping*


1. Click **Create Stage Relationship To Target Property Mapping**  on the **Stage Relationship To Target Property Mappings** toolbar.
2. In the **Stage Relationship** section of the **Create Stage Relationship To Target Property Mapping** dialog box, Click **Search**  to find the relationship that you want to map to a target property.
3. Select a property from the results to display it in the **Selected stage relationship** box.
4. In the **Target Property** section, select the target system to which you want to connect from the list, and click **Connect**.
5. In the **Target Property** search box, click **Search**  to get a list of the target system properties to map to a stage relationship.

**NOTE** If the target relationship does not exist in the target system, you can enter a name for it in the **Free Text** box.




6. Select a target property from the results to display it in the **Selected target property** box.
7. In the **Details** pane, provide a description for the mapping in the **Description** box.
8. In the **End1 to End2 Error Propagation Minimum Severity** list, select the severity threshold at which an issue with End 1 will be propagated to the object on End 2. Any severity setting greater than this will also propagate, if the setting was warning, then both warnings and errors propagate up to the object.
9. In the **End2 to End1 Error Propagation Minimum Severity** list, select the threshold at which an issue with End 2 will be propagated to the object on End 1. Any severity setting greater than this will also propagate, so if the setting was warning, then both warnings and errors propagate up to the object.
10. Click **Save**.

## Stage relationship to target relationship mappings

The **Stage Relationship To Target Relationship Mappings** pane displays the stage relationships mapped to the target relationships in the target system database.

1. In the **Stage To Target Object Map** pane, select a stage object in the **Stage Object Name** column.
2. Click **Show Stage Relationship To Target Relationship Mappings**  to display the stage relationship to target relationship list.

### *Create a stage relationship to target relationship mapping*

1. In **Stage Relationship To Target Relationship Mappings** toolbar, click **Create Stage Relationship To Target Relationship Mapping** .
2. In the **Stage Relationship** section of the **Create Stage Relationship To Target Relationship Mapping** dialog box, use the search box to find the relationship that you want to map to a target relationship. Click **Search**  to get a list of all the relationships that match your criteria.
3. Select the property and it appears in the **Selected stage relationship** box.
4. In the **Target Relationship** section, select the target system, and click **Connect**.
5. In the **Target Relationship** search box, click **Search**  to search for the target system relationship to map to a stage relationship.

**NOTE** If the target relationship does not exist in the target system, you can enter a name for it in the **Free Text** box.

6. Select a target relationship from the search results to display it in the **Selected target relationship** box.

**NOTE** If the target relationship does not exist in the target system, you can enter a name for it in the **Free Text** box.

7. In the **Details** pane, provide a description for the mapping in the **Description** box.
8. In the **End1 stage Object** list, select the stage object required for End 1.
9. In the **End2 stage Object** list, select the stage object required for End 2.
10. In the **End1 to End2 Error Propagation Minimum Severity** list, select the severity threshold at which an issue with End 1 will be propagated to the object on End 2. Any severity setting greater than this will also propagate, so if the setting was warning, then both warnings and errors propagates up to the object.
11. In the **End2 to End1 Error Propagation Minimum Severity** list, select the severity threshold at which an issue with End 2 will be propagated to the object on End 1. Any severity setting greater than this will also propagate, so if the setting was warning, then both warnings and errors propagates up to the object.

### **NOTES**

- Click the **Reverse Stage Rel Direction** option to reverse the stage relationship direction, if required.


- If you are using a different target system for validation and export, the propagate error severity settings are then taken from the validation target system.

12. Click **Save**.





## Stage relationship property to target relationship property mappings

The **Stage Relationship Property To Target Relationship Property Mappings** pane displays the names of the stage relationship properties mapped to the target relationship properties in the target system database.


To view the stage relationship property to target relationship property mappings:

1. In the **Stage To Target Object Map** pane, select a stage object in the **Stage Object Name** column.
2. Click **Show Stage Relationship To Target Relationship Mappings**  to display the stage relationship to target relationship and the stage relationship property to target relationship property lists.

### *Create a stage relationship property to target relationship property mapping*

1. Click **Create Stage Relationship Property To Target Relationship Property Mapping**  on the toolbar.
  2. In the **Stage Relationship Property** section, click **Search**  to find the local relationship property that you want to map to a target relationship property.
  3. In the **Selected stage relationship property** box, select a relationship property from the list.
  4. In the **Target Relationship Property** section, select the target system from the list, and click **Connect**.
  5. In the **Target System** search box, click **Search**  to locate the target system relationship property to map to the stage relationship property.
-  **TIP** If the target relationship property does not exist in the target system, you can enter a name for it in the **Free Text** box.
6. Select a target relationship property from the search results to display it in the **Selected target relationship property** box.
  7. In the **Details** pane, provide the description for the mapping in the **Description** box.
  8. In the **Block Property on Export Minimum Severity** list, select the severity threshold at which the relationship property will be blocked from exporting if it fails a validation rule. Any severity setting greater than this will also propagate; if the setting is warning, then both warnings and errors propagate up to the object.
  9. In the **Property Error Propagation Minimum Severity** list, select the severity threshold at which the failure of the relationship property will be propagated up to the object and prevent it from being exported. Any severity setting greater than this will cause the property to be

blocked; if the setting was warning, then both warnings and errors propagate up to the object.

 **TIP** For more information about the severity level assigned to validation rules, see *Define Validation Rules and Rulesets* (on page 56).

10. Click **Save**.

## SECTION 9

# Define Validation Rules and Rulesets

### When do I need to create a validation rule?

Create a validation rule when you need to validate the data before you export it to a target system. The validation rule ensures that the data conforms to the quality requirements, schema standards, and so on.

### How do I ensure that imported data conforms to the object property schema and the relationship property schema?


The Check Properties Against Schema validation rule ensures that imported data conforms to the object and relationship property schemas. This rule can be auto-generated when configuring the system options.


### Validation rules and rulesets

Validation rules verify that the data in an imported record meets the data standards specified for the target system. Each validation rule contains a formula that evaluates the imported data in one or more fields and verifies that it matches the existing criteria and hierarchy found in the target system database. Each validation rule includes an error message that is recorded in a validation report when the rule returns an invalid value.

All the validation rules that will be used in a job are grouped into a set. You can create a validation rule set, and then create the individual rules that will make up that set.

For a list of the existing validation rules and their actions supplied with Smart Data Validator, see *Validation Rules and Actions* (on page 113).

 **TIP** You can configure Smart Data Validator to automatically generate validation rules when you create your import mapping. For more information, see *Configure the automatic generation of validation rules* (on page 59) and *Create a validation rule relationship* (on page 63).

 **NOTE** When any validation rule is deleted that contains class definitions, relationship definitions, or property definitions that are no longer used by Smart Data Validator, those schema items are automatically removed from the database.

---

### What do you want to do?


- *Create a new validation rule set* (on page 57)
- *Edit a validation rule set* (on page 57)
- *Copy a validation rule set* (on page 58)
- *Configure the automatic generation of validation rules* (on page 59)
- *Create a validation rule* (on page 58)
- *Edit a validation rule* (on page 59)
- *Manage validation rules* (on page 59)
- *Create a validation rule relationship* (on page 63)





## Create a new validation rule set

### Can a validation rule be associated with different rule sets?


Yes. You can use the **Rule Management** dialog box to move a rule to any rule set, or you can search for a rule in the **Edit a Rule Set** dialog box and associate it with any rule set.

1. Click **Create a new Rule Set**  on the **RuleSets** toolbar.
2. In the **Create New Rule Set** dialog box, provide a name and description for the new rule set.


Use the **Rule Name**, **Class Definition**, and **Property Definition** search boxes to find the name of the rule, class definition, or property definition required for the new rule set. Click **Search**  and the rules that match your search criteria appear in the list.


3. To add a validation rule to the rule set, select it in the grid, and click **Add** .


#### **TIPS**

- To remove a validation rule from the rule set, click the validation rule in the **Rules** pane, and click **Remove** .
  - To change the order that the validation rules run in the rule set, click the validation rule and use the up or down arrows to change the order. The rules can be run in a specified order. For example, the **Check Exist** rule can be run first, because if the object does not exist, any rules preceding the **Check Exist** rule will result in an error.
4. Click **Save**.

## Edit a validation rule set

1. Click **Edit the Rule Set**  on the **RuleSets** toolbar.
2. Edit the details of the validation ruleset as necessary.


 **NOTE** For more information on editing a validation ruleset, see *Create a new validation rule set* (on page 57).

 **TIP** To edit the order that the validation rules run in the rule set, click the validation rule and use the up or down arrows to change the order. The rules can be run in a specified order. For example, the check exist rule can be run first, because if the object does not exist, any rules preceding the check exist rule will result in an error.

3. Click **Save** when finished.

 **NOTE** You can delete a rule set by selecting it and clicking **Delete Rule Set** .

## Copy a validation rule set

1. In **RuleSets** list, select the rule set you want to copy.
2. Click **Copy the RuleSet**  on the **RuleSets** toolbar.
3. Provide the name for the new ruleset in the **New RuleSet Name** box.
4. Click **Save**.

## Create a validation rule

**How do I decide whether to select a class definition or a relationship definition schema type when creating a validation rule?**


When you create a validation rule, select the class definition schema type to validate objects and their properties, and select the relationship definition schema type to validate relationships and their properties exposed by their linked interfaces.


**What does the End for Error Propagation option in the Rule Details page do?**

The **End for Error Propagation** option specifies either one of the ends of a relationship to be considered as invalid when the imported data fails to be validated against a relationship property rule.


1. From the **RuleSets** pane, select a rule set in which you want to create the new rule.


 **NOTE** Rules must belong to at least one rule set.

2. Click **Create a new Rule**  on the **Rules** toolbar to open the **Create Rule** dialog box.
3. In the **Basic Details** page, type a name and description for the new validation rule.
4. Select either a **Class Definition** or a **Relationship Definition** from the **Schema Type** options.
5. In the **Rule Type** box, select the validation rule type required from the list. For more information, see *Validation rule description* (on page 114).


 **NOTE** The validation rule types listed in the **Rule Type** box are dependent on the schema type selected.


6. Search for the class definition or relationship definition on which you want to base the rule in the **Class Definition** or **Relationship Definition** boxes.
7. Select the class definition or relationship definition required from the search results.

 **NOTE** You can check the **Turn Validation Rule Off** option if you do not require the rule to run on specific data.
8. In the **Validation Rule Error Message** box, type the full error message to be displayed when the validation rule fails.
9. From the **Validation Rule Severity Level** list box, select the validation rule severity level to be assigned when this rule fails.
10. Click **Next**.




11. In the **Rule Details** page, use the **Property Definition** box to search for the property definition on which you want to base this rule, and click **Search** .
12. Choose a property from the results list box.
13. Provide additional details for the rule, as necessary, based on the rule type selected. Click **Next**.
14. View the **Summary** and review the details defined for the rule.  
**NOTE** You can click any page headings to edit the properties on that page, as required.
15. Click **Finish** to save the validation rule. The new validation rule appears in the **Rules** list and displays the properties associated with the rule in the **Properties** pane.

## Edit a validation rule

1. Click **Edit the Rule**  on the **Rules** toolbar to open the **Edit Rule** dialog box.
2. Edit the details of the validation rule properties as necessary.  
**NOTE** For more information on properties that define rules, see *Create a validation rule* (on page 58).
3. Click **Finish** to save the changes to the rule.

**NOTE** To delete a rule, select a rule in the **Rules** list and click **Delete**  on the **Rules** toolbar.

## Manage validation rules

1. Click **Manage the Rule**  on the **Rules** toolbar to open the **Rule Management** dialog box.
2. Use the  and  buttons to move the rule in and out of the rulesets, as required.
3. Click **Save**.

**NOTE** Rules must belong to at least one rule set.

## Configure the automatic generation of validation rules

When you create your import mappings from the target system, you can automatically generate validation rules that are set on the import definition. By selecting this option, you allow the system to determine which rules it will run for each action, saving configuration time. You can configure which rules are automatically generated for each action in the Smart Data Validator system options. The actions and rules that appear are configurable. For more information, see *Rule auto-generation model* (on page 60).

For more information about the system default options, see *Set Smart Data Validator Default Options* (on page 23).

## Rule auto-generation model

Each action you see in the import mapping user interface comes from a specific enumerated list. The following three enumerated lists for actions are delivered with Smart Data Validator:

**VTLObjectActions**, **VTLPropActions**, and **VTLRelActions**.

All validation rules have the same class definition and realize a specific interface definition with properties for each rule type, such as `IVTLCheckObjectExistsValidationRule`.

For an action to appear in the Validation page of the **Smart Data Validator Default Options** dialog box, the enum must instantiate the `IEnumEnumVTLExt` interface. For a rule to appear, it must have the `IVTLRuleSupportsAutoGen` interface. The rules must also have the `IPropertyValidationRule` interface to be available for property actions, the `IVTLRelationshipValidationRule` interface to be available for relationship actions, and the `IVTLRelDefValidationRule` interface to be available for relationship property actions.

To ensure the creation of logical relationships, the relationship definition for `VTLActionVTLValidationRule` includes specific interfaces to relate the action to the rule.

The following validation rules are provided in the base configuration:

Item	Action	Rule
Object	Check Exists	Check Object Exists, Compare All Property Values For Objects
Object	Create Update	Check Properties Against Schema, Compare All Property Values For Objects
Object	Delete	Check Object Exists, Compare All Property Values For Objects
Object	Terminate	Check Object Exists, Compare All Property Values For Objects
Object	Rename	Check Object Exists, Compare All Property Values For Objects
Object	Update	Check Object Exists, Check Properties Against Schema, Compare All Property Values For Objects
Object	No Action	Compare All Property Values For Objects
Property	Check Staging File Exists	Check File Exists
Property	Check Target System File Exists	Check File Exists

Item	Action	Rule
Relationship	Create Update	Target System Cardinality
Relationship	Check Exists	Check Relationship Exists
Relationship	Delete	Check Relationship Exists
Relationship	Terminate	Check Relationship Exists
Relationship Property	Create Update	Check Properties Against Schema

★ **IMPORTANT** The **Compare Value** rule ignores null values in the target system if the property action is set to **Compare Create**, but respects the null values if the action is set to **Compare Value**. This is because exporting with a **Compare Create** action requires blank target system values to be updated without altering any existing values.

The following rules are not auto-generated in the delivered system, but they can be changed to auto-generate using the system options.

Item	Action	Rule
Object	Check Exists	Check Claimed To Sub Config, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config
Object	Create Update	Check Claimed To Sub Config, Check Object Exists, Check Parallel Claim, Check Revised to Parallel Config, Check Revised to Sub Config, Check File Exists for Document Revision, Check is Claimable, Check is Revisable
Object	Delete	Check Claimed To Sub Config, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config
Object	No Action	Check Claimed To Sub Config, Check Object Exists, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config
Object	Rename	Check Claimed To Sub Config, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config
Object	Terminate	Check Claimed To Sub Config, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config

Item	Action	Rule
Object	Update	Check Claimed To Sub Config, Check Parallel Claim, Check Revised to Parallel Config, Check Revised to Sub Config, Check File Exists for Document Revision, Check is Claimable, Check is Revisable
Object	Validate Only	Check Claimed To Sub Config, Check Object Exists, Check Parallel Claim, Check Properties Against Schema, Check Revised to Parallel Config, Check Revised to Sub Config, Compare All Property Values For Objects, Check File Exists for Document Revision
Property	Check Staging File Exists	Compare Value
Property	Check Target System File Exists	Compare Value
Property	Compare Create	Check File Exists, Compare Value
Property	Compare Value	Check File Exists, Compare Value
Property	Validate Only	Check File Exists, Compare Value
Relationship	Check Exists	Check for Cyclic Relationship, Compare All Property Values For Objects, Target System Cardinality
Relationship	Create Update	Check for Cyclic Relationship, Compare All Property Values For Objects, Check Relationship Exists
Relationship	Delete	Check for Cyclic Relationship, Compare All Property Values For Objects, Target System Cardinality
Relationship	Terminate	Check for Cyclic Relationship, Compare All Property Values For Objects, Target System Cardinality
Relationship Property	Compare Create	Check Properties Against Schema
Relationship Property	Compare Value	Check Properties Against Schema

## NOTES

- The VTLActionVTLValidationRule relationship definition links the action to the specific rule interface definition.
- For a detailed description of each validation rule, see *Validation rule description* (on page 114).

- For more information on setting the Smart Data Validator default options, see *Set Smart Data Validator Default Options* (on page 23).

## Naming convention for rule auto-generation

There is a naming convention for the validation rules that are automatically generated. Depending on the rule type, the rule name is a combination of the rule interface display name, class or property definition name, and relationship definition name.

Rule Type	Rule Name
Object Rule	[Rule interface display name]-[Class definition name]
Property Rule	[Rule interface display name]-[Property definition name]-[Class definition name]
Relationship Rule	[Rule interface display name]-[Relationship definition name]-[Class definition name]
Relationship Property Rule	[Rule interface display name]-[Relationship definition name]

### NOTE

- The auto-generated rule name length cannot exceed 128 characters. In such case, the rule name is a combination of the rule interface display name and the global unique identifier (GUID) separated by a hyphen '-'.


## Smart Data Validator rule severity levels

The `IEnumVTLExt` interface definition provides the `VTLReportingColour` property for the rule severity levels to the two supported validation rules.


## Create a validation rule relationship

Use the following example to create a validation rule relationship.

- In the SmartPlant Foundation Desktop Client, find the interface definition to create the validation rule, such as **IVTLCheckObjectExistsValidationRule**.
- Select the action required, for example, **CheckExists**, from one of the three action enum lists that are found in SmartPlant Foundation: **VTLObjectActions**, **VTLPropActions**, and **VTLRelActions**.

 **TIP** You can select the action depending on what type of validation rule is required, such as an object, property, or relationship rule.

- Relate the **IVTLCheckObjectExistsValidationRule** interface to the appropriate action using a drag-and-drop operation.

 **NOTE** To automatically generate the validation rule relationship, relate the **VTLActionVTLValidationRule** to the enum **CheckExists**.

For more information on creating relationships in SmartPlant Foundation, see *SmartPlant Foundation Desktop Client User's Guide*.

## SECTION 10

## Define Implicit Delete Rules

### When is implicit delete process required?

When exporting data from the staging to target system, you may have some redundant data in the target system that needs to be deleted or terminated. In that case, you can configure an implicit delete rule to identify this data and use the Import Validate Delete Export workflow to delete it in the target system.

### Implicit delete rules

Implicit delete is one of the component processes that Smart Data Validator uses and is based on the configured mappings and groupings. Users can implicitly decide to delete or terminate a group of objects from the target system, because they are no longer in the supplied input data submission. Smart Data Validator compares the new data submission with the existing data in the target system and flags any items that are missing from the new data submission in the form of a deletes report. Once approved, the existing data in the target system can then be deleted implicitly on export of the new data submission. This behavior is configurable within Smart Data Validator and SmartPlant Foundation.

The deletion of data from the target system during export is controlled by a set of configurable **Implicit Delete Rules**. These implicit delete rules determine what is to be deleted or terminated from the target system based on what is present in the staging area. A deletes report is run after validation so that cardinality checks on relationship objects can be included in the validation report, and before the export of any data to the target system and any deletion of data.

The delete rules compile a set of items that exist in the target system, matching the context and object type. The rules are run against the data submission and generate the delete instructions, which are added to the data submission in Smart Data Validator. A deletes report is run so that you can review what is going to be deleted from the target system. Once the implicit delete is approved, the export process runs and deletes or terminates the items in the target system, as determined by the delete instructions.

**⚠ WARNING** As the implicit delete functionality can delete data from the target system database, you must run the deletes report each time before any export proceeds, so that all object deletions are reviewed and approved.

For more information on the implicit delete functionality, see *Understand implicit delete functionality* (on page 123).


---




### What do you want to do?




- *Create an implicit delete rule* (on page 65)
  - *Edit an implicit delete rule* (on page 66)
-




## Create an implicit delete rule

1. Click **Create Implicit Delete Rule**  on the **Implicit Delete Rules** toolbar to open the **Create Implicit Delete Rule** dialog box.
2. In the **Basic Details** page, type the name and description for the new implicit delete rule.
3. From the **Delete Action** list, select whether you would like items to be deleted or terminated as a result of the implicit delete process.
4. Select the delete rule you want from the **Rule Type** list. For example, **Object** or **Relationship**.
 

**TIP** The rule type selected determines which properties and pages are available on the **Create Implicit Delete Rule** dialog box.
5. Select the **Disable Auto Claim** check box to stop implicit delete claiming objects and relationships from a higher configuration that are not already in the current configuration.
6. Select the **Exclude Claimed Items** check box to stop the deletion of objects and relationships that are claimed to lower configuration.
- NOTE** This option restricts the deletion of claimed objects and relationships, but not documents.
7. Click **Next**.
8. If you selected **Relationship** as the rule type, use the **Target Relationships** box on the **Target Relationship Details** page to search for and select the relationship for which you want to create an implicit delete rule.
9. If you selected **Relationship** as the rule type, use the **Owner End** box to choose which end of the relationship owns the relationship. Click **Next**.
10. In the **Target Details** page, use the **Target Object** box to search for the object for which you want to create the delete rule, and click **Search** . Choose an object from the results list box.
11. In the **Properties Criteria** box, search for the name of a property that you want to add as a grouping property, and click **Search** .
12. Select a property from the results list box, and click **Add**  to add the property to the table.
 


**TIP** If necessary, click **Delete**  to remove any properties from the list.
13. In the **Relationship Criteria** box, search for the name of the relationship that you want to add as a grouping relationship, and click **Search** .
14. Select a relationship from the results list, and click **Add**  to add the property to the **Grouping Relationships** list table.
 

**TIP** If necessary, click **Delete**  to remove any relationships from the list.
15. Click **Next**.
16. View the **Summary** and review the details.



17. Click **Finish** to save the implicit delete rule. The new implicit delete rule appears in the **Implicit Delete Rules** list and displays the properties associated with the delete rule in the **Properties** pane.

## Edit an implicit delete rule

1. Click **Edit an Implicit Delete Rule**  on the **Implicit Delete Rules** toolbar.
2. Edit the details of the implicit delete rule properties as necessary.

 **NOTE** For more information on editing the properties of implicit delete rules, see *Create an implicit delete rule* (on page 65).

3. Click **Finish** to save the changes to the rule.

 **NOTE** To delete an implicit delete rule, select it and click **Delete the Implicit Delete Rule** .

## SECTION 11

# Define Job Definitions

A job definition is a combined set of components configured into a template for the import, validation, and export of data to one or more target systems. The job definition manages the quality of the data to be loaded into the target system configuration. You can define the import mappings, export mappings, validation rule sets, and delete rule sets.

When a job definition has been configured, it can be used by the Smart Data Validator Job Management application to load jobs containing data into a target system. For more information on the process of job management, see *Smart Data Validator Job Management User's Guide*.

To view, edit, create, or delete job definitions, select **Job Definitions** from the main toolbar list box.

### NOTES

- You create job definitions using a dynamic job wizard, which uses the workflow selected as the basis for the behavior of the job running through Smart Data Validator Job Management.
- Each job definition has a set of process steps that manages the import, validation, and export processes. Some process steps appear in the wizard only when a specific workflow has been selected, such as Implicit Delete Rules.
- Each job definition can be deleted using SmartPlant Foundation or Smart Data Validator Administration, but only if there are no active jobs attached to the job definition in Smart Data Validator Job Management. For more information, see *Delete a job definition* (on page 70) and *Delete Job Definitions Using SmartPlant Foundation* (on page 156).

Each job definition is configured using the following components:

- **Basic Details** - The job description name and description, as well as the workflow used to export data to the primary target system.
- **Workflow** - The defined set of steps that drive the behavior of the job running through Smart Data Validator. The following workflow templates are delivered with Smart Data Validator: **Import Validate Export**, **Import Validate Delete Export**, and **Import Validate Export Delete Job**.

### NOTES

- For more information on customizing Smart Data Validator workflows, see *Workflow Customization* in the *Smart Data Validator Customization Guide*.
- All workflows are created and configured using SmartPlant Foundation Desktop Client. For more information, see *Workflows* in the *SmartPlant Foundation Desktop Client User's Guide* and in the *How to Configure the Workflow Model Guide*.
- **Target Systems** - The target system or systems used to validate and export the data.
- **Import Definitions** - The import definitions used to import the data.
- **Rule Sets** - The validation rule sets used to validate and verify the imported data.
- **Implicit Delete Rules** - The delete rules used to implicitly delete data.


- **Validation Report Definitions** - The validation report definition used to create the validation report.
  - **Access Groups** - The access groups that have permission to use the job definition to create a new job in Smart Data Validator Job Management.
  - **Record Job Details** - These are used to store the job information, such as Import details, Validation details, Export details, and other job information.
- 


### What do you want to do?

- *Create a job definition* (on page 68)
  - *Edit a job definition* (on page 70)
  - *Delete a job definition* (on page 70)
- 


## Create a job definition

★ **IMPORTANT** The creation of a job definition and the dialog boxes that appear are based on the workflow selected. The following example shows creating a job definition using the workflow **Import Validate Delete Export**, which includes adding delete rules for the implicit delete functionality. For more information on implicit delete, see *Define Implicit Delete Rules* (on page 64).


1. Click **Create Job Definition**  on the **Job Definition** toolbar.
2. On the **Basic Details** page, type the name and optional description for the job definition.
3. Select the workflow to be used as the basis for the job definition from the **Workflow** list box. For example, **Import Validate Delete Export**.

 **NOTE** The workflow selected changes the pages that appear in the **Create Job Definition** dialog box for you to use defining the properties.

4. Select the **Run Inconsistency Property Values** option to ensure that the inconsistency check is always run when the workflow selected contains the validation process step. The inconsistency check processes all objects in the staging area, and when any two instances of the same property on the same object are found with differing values, an error message is flagged in the validation report. You can then decide whether to propagate the error to the object.

 **NOTE** If you clear this option, you stop the inconsistency check from running during validation, and you may get results that are not expected.

5. Select the **Report Invalid Action Combinations** option to report the number of invalid action combinations on an object by looking at the data from combined mappings. For example, if an object is mapped for terminate, then there should not be any properties mapped on the object for create update. If any properties for create update are found to be mapped to the object, an error is propagated.

 **NOTE** If you clear this option, you stop the ability to report the number of invalid action combinations running during validation, and you may get results that are not expected.

6. Select the **Report Multiple Action Conflicts** option that checks for multiple actions on a single object. If multiple actions are found, the user is warned about which action is being taken forward. For example, if an object is found to have both an update and a create update action, then the user is informed that the object is being taken forward with the create update action.

#### **NOTES**

- If you clear this option, you stop the system from reporting on multiple actions running during validation, and you may get results that are not expected.
- The user cannot choose the action that is taken forward.

**★ IMPORTANT** The previous three options do not appear if the workflow does not contain a validation process step.

7. Select the **Is Enabled** option to activate the job definition for use in creating jobs in Smart Data Validator Job Management.

**💡 TIP** If you clear this option, this job definition cannot be used to create jobs in Smart Data Validator Job Management.

**📝 NOTE** A job definition can only be deleted in SmartPlant Foundation when there are no jobs using the job definition in Smart Data Validator Job Management. For more information, see *Delete Job Definitions Using SmartPlant Foundation* (on page 156).

8. Select the **Record Job Details** option to store the job information, such as Import details, validation, export details and more.

9. Click **Next**.


10. On the **Target Systems** page, select the target system or systems to validate against and export to.


**💡 TIP** You must set at least one target system as the validation destination, and at least one target system as the export destination. They can be different target systems.

11. Click **Next**.

12. On the **Export Configuration** page, set the default options for new document version creation:


- **Allow Version Creation** - Select this option to create a new document version if the same file is available in the input CSV file during subsequent imports.
- **Copy files from the Previous Version** - Select this option to copy files from the previous version of a document while creating a new version.


13. Using the remaining pages of the **Create Job Definition** dialog box, select an **Import Definition** to add to the job definition, and then add one or more of the **Delete Rules, Rule Sets, Validation Report Definitions**, and **Access Groups** to use by clicking the left add arrow  to add them to the appropriate list to apply them to the job definition.

**💡 TIP** You can remove any of the selected objects from the list by selecting it and clicking the right remove arrow .

14. Click **Next**.


15. On the **Summary** page, check the details for all the components selected in the job definition. All completed component pages are displayed with a check mark.


 **TIP** To update a setting in a previous component page, click on the page and update the setting, as necessary.


 **WARNING** If you change a setting that affects other pages, for example by changing the workflow, then all subsequent pages must be completed again.

16. Click **Finish** to save the job definition.

## Edit a job definition

1. In the **Job Definition** pane, select a job definition from the list.
2. In the main toolbar, click **Edit Job Definition** .
3. In **Edit Job Definition** dialog box, modify the job definition components on each page as required. For more information on the component properties for each page, see *Create a job definition* (on page 68).


 **IMPORTANT** If the job definition is disabled, existing jobs using this job definition and still running in Smart Data Validator Job Management can finish. However, you cannot create any new jobs with this job definition.

 **WARNING** If you change a setting that affects other pages, for example by changing the workflow, then all subsequent pages must be completed again.

4. Click **Finish** to update the job definition.

## Delete a job definition

You can delete a job definition from the system using Smart Data Validator Administration or the SmartPlant Foundation Desktop Client. However, you cannot delete a job definition if it has any active jobs still attached.

1. Click a job definition from the **Job Definitions** list
2. Click **Delete** , and a warning message appears. If you want to delete the job definition, click **OK**.

## SECTION 12

# Use Function Column Type Mappings

When creating a column header mapping, you can select either **Computed** or **Prompted API** column types that use functions to match the imported data to the target system configuration.

Administrator users must have programming or function creation experience to use this functionality. However, this has been made easier as the functionality is provided using an Intellisense style control, which streamlines the process.

**NOTE** When a CSV file contains data that is not in invariant format, for example, the CSV file uses a comma as decimal separator, you must create a computed column type to convert the comma to a decimal.

---

### What do you want to do?

- Use the *Computed column* (on page 71)
  - Use the *Prompted API column* (on page 75)
- 

## Use the Computed column

When you select the **Computed** column type, you must enter a function as the **Computed API** box cannot be blank. For example:

1. Type **func.** in the **Computed API** box, and a drop-down list displays all the functions that are available.
2. Double-click the parameter **Replace** to select it, and then type the rest of the parameter string syntax, for example **func.Replace([TAG\_NAME], "PSS", "EDW")**. For more examples of computed API functions, see *Functions* (on page 89).

### TIPS

- Other input columns can be viewed inside each function by typing **[** to display a drop-down list.
- A green check mark indicates that the function is valid.

The following is a list of all the available functions for the **Computed** column type; each function has a set of parameters that can also be viewed.

Function Name	Description
<b>AddDefaultUOMIfMissing</b>	Applied to a column with a UOM value. If there is no UOM supplied and the specified property maps to a property that is scoped by a UOM type, then the output has the default UOM added to the value. Otherwise, the value is left untouched. Default value is based on SI in the schema.
<b>Concat</b>	Concatenates a set of input strings.
<b>ConvertToHash</b>	Returns the SHA-1 (Secure Hash Algorithm 1) capitalized hash value for the input string. The SHA-1 algorithm computes a uniquely identifiable fixed hash value for the input string.
<b>DateTimeColumn</b>	Input value is converted to date time format and can include time zone, yyyy/MM/dd-HH:mm:ss:fff.
<b>Decode</b>	Looks for value matches and replaces them with a new value.
<b>Divide</b>	Divides parameters using a numerator.
<b>GetDefaultUOMIfMissing</b>	Applied to a column with a UOM value. If there is no UOM supplied and the specified property maps to a property that is scoped by a specific UOM type, then the output is set to that default UOM. Otherwise, the value is left untouched. Default value is based on SI in the schema.
<b>GetDocumentRevisionsCountFromTargetSystem</b>	Returns the number of revisions available for a document in the target system.
<b>GetFileName</b>	Returns the file name from a file path.
<b>GetJobDetails</b>	Returns the value of the property specified in the input string.
<b>GetMajorRevisionCodeFromTargetSystem</b>	Uses major revision code from the target system.
<b>GetMinorRevisionCodeFromTargetSystem</b>	Uses minor revision code from the target system.
<b>GetParentObjectClassDef</b>	Returns the class definition of the parent object. A blank value is returned if the parent object is not found in either the input CSV file or in the target system for all the possible class definitions specified in the PossibleParentObjectClassDef parameter.



Function Name	Description
<b>GetTargetSystemValueIfEmptyReturnDefault</b>	Replaces input values with the value from the target system, and if the object specified in the QueryDef parameter is not available in the target system, the default value specified in the default value parameter is returned.
<b>GetValueFromTargetSystem</b>	Replaces values with the value from the target system.
<b>IndexOf</b>	Creates an index of parameters; part 1 is input string and part 2 is search string.
<b>InvertedFilter</b>	Checks for the property name in the inverted CSV file that matches the property name configured in the computed column. The corresponding computed column is then evaluated and the property value is imported into staging system.
<b>Join</b>	Joins parameters with same separator.
<b>Left</b>	Input is from the left by the specified length in the computed column.
<b>Length</b>	Value in length as an input string.
<b>Minus</b>	Reduces initial value by another value being subtracted.
<b>Multiply</b>	Multiplies input value by a value.
<b>PadLeft</b>	Takes the input string and adds padding to the left using a padding character to match the string length.
<b>PadRight</b>	Takes the input string and adds padding to the right using a padding character to match the string length.
<b>RegexMatch</b>	If matches a regular expression (such as uppercase characters) returns True, if not then returns False.
<b>RegexReplace</b>	Replaces existing regular expression characters (such as uppercase characters) in the name with new characters.
<b>Replace</b>	Replaces existing characters in the name with new characters.
<b>Right</b>	Starts the count from the right by the specified length in the computed column.
<b>Split</b>	Splits a character from the input string.
<b>SplitAlphaNumericSequence</b>	Splits up an alpha numeric sequence into separate values.

Function Name	Description
<b>SubString</b>	Starts input string from sub string.
<b>Sum</b>	Adds values together so changes input values to a new value.
<b>ToLower</b>	Changes the input string value to lower case.
<b>ToUpper</b>	Changes the input string value to upper case.
<b>Translate</b>	Replaces a sequence of characters found in a string with another set of characters, a single character at a time.
<b>Trim</b>	Trim a character value.
<b>TrimEnd</b>	Trim a character value from the end.
<b>TrimStart</b>	Trim a character value from the start.
<b>YMDColumn</b>	Input value is converted to the yyyy/MM/dd format.


## NOTES

- Microsoft .NET methods calculate some of the computed columns. For more information on how some computer columns function, see *Computed column .NET methods* (on page 107).
- Comma decimal separated values in your CSV file must be surrounded by double quotes, for example, "123".
- When a CSV file uses comma decimal separators, you have to create a computed column to convert the comma separated value to a decimal value. Create a computed column, map it to the property that was mapped on your physical column, and run this computed column function `Func.Replace([PHYSICAL COLUMN], ",", ".")`.
- You can also make use of SmartPlant Foundation environment- and session-related variables, for example, Job Name, UserName, and Current query configuration. These are accessible using the @ character. For further information, see *Environment variables* (on page 127) for a list of all the variables supported.
- When you use the **Replace** function to replace the @ symbol, Smart Data Validator considers the @ symbol as an environment variable and does not replace it. To replace the @ symbol, you must create a column of Constant type, and give it the value of @. For example, `Func.Replace([PHYSICAL COLUMN], "[CONSTANT COLUMN]", "1")`. Here, this example replaces the @ symbol with 1.
- When you use the **GetParentObjectClassDef** function, if the second parameter, `ObjectClassDef`, is not available in the CSV file, you can use another computed column to get the class definition of the object.  
  
For example, if the CSV file for SPO/E system tags provides an object with the classification Cooling Tower, we can get its class definition of 'SPXTagEquipment by using the computed column function: `func.GetValueFromTargetSystem("true",`


```
"#SPXTagClass, .Name=[TagClass]", "-  
SPFClassMember+TagClassClassDef.Name" ) ).
```

## Use the Prompted API column

When you select the **Prompted API** column type, you must add a function to use. The **Prompted API** box cannot be blank. Type **func.** in the **Prompted API** box, and a drop-down list displays all the parameters that are available to help you write a function. Double-click the parameter to select it, and then type the rest of the parameter syntax. For examples of prompted API functions, see *Functions* (on page 89).

 **NOTES** The following is a list of all the available parameters for the **Prompted API** column type.

Parameter Name	Description
<b>GetEnumEntries</b>	Gets values with enum entries.
<b>GetObjectPropertiesByType</b>	Gets object properties grouped by the type.
<b>GetValueFromTargetSystemObjects</b>	Gets values from the target system objects.
<b>GetValuesFromTargetSystemRelationshipExpansion</b>	Gets values from the target system relationship expansions.

 **NOTE** A green check mark indicates that the function is valid.

## SECTION 13

# Triggering a Job External to the Job Management Module

You can trigger a job external to the Smart Data Validator Job Management when you want to run a job process without any user interaction, warnings, or errors. To trigger a job externally, the job has two defined stages:

- Create the job definition configured for the import, validation, and export of data to a specific target system or multiple target systems in Smart Data Validator Administration.
- Create a control file that contains all the information required for the job creation to be externally processed. The control file is an XML format file.

### Job definition

The foundation for each job is the job definition that is created in the Smart Data Validator Administration module. This combines all the settings into one job, which has the following steps:

- Create the import mapping, validation rules, export mapping, and implicit delete rules.
- Create a job definition configured for the import, validation, and export of data to a specific target system or multiple target systems.

### Control file

You must create the XML control file to define the job information and process the definitions in Smart Data Validator Job Management. An example control file is delivered in the product installation folder as an XML file, **Silent Job Creation Control File - Template.xml**. By default, it is located at `[installation location]\Smart\SDV\[version]\SilentJobSetup\SampleControlFiles`.

---

### What do you want to do?

- *Using silent workflows and job scheduler* (on page [77](#))
  - *Understanding the control file format* (on page [77](#))
  - *Trigger the job* (on page [80](#))
-

## Using silent workflows and job scheduler

There are three silent workflow templates delivered with Smart Data Validator that you can use to process the job data without any user interaction, warnings, or errors. They are:

- **Silent Import Validate Export**
- **Silent Import Validate Delete Export**
- **Silent Import Validate Export Delete Job**

The VTLSilentJobCreationScheduler scheduler is delivered with silent workflows and has a minimum polling interval of 30 minutes. The scheduler includes the **Control files location** property, which contains the file path to the **Control Files** folder. It looks into the control files folder every 30 minutes and processes new control files.

**NOTE** Administrators can change the polling value and properties defined on the scheduler using the Update form. However, administrators must override the default property using custom code to change the polling value. For more information, see *Polling Intervals for the Scheduler* in the *SmartPlant Foundation Server Customization Guide*.

You can load the silent workflows and the scheduler VTLSilentJobCreationScheduler into SmartPlant Foundation using the **Loader** in SmartPlant Foundation Desktop Client. The workflows and the scheduler are located in the *[Install folder]\Program Files (x86)\Smart\SDV\version\SilentJobSetup\Model* folder.

## Understanding the control file format

The XML control file is used to define the job information and process the definitions in Smart Data Validator Job Management.

As seen in the following example, the XML file has some pre-defined XML tags, which are described in the table below.



- 1 **<User\_Name>** - Allows you to provide the role name that will access the application server.
- 2 **<StartDateTime>** - Allows you to provide the date and time value, date format, and the time zone for the date for the job to be picked up by the scheduler.
- 3 **<Jobs>** - Allows you to list all the jobs that you want to run using the same control file.
- 4 **<Target\_System>** - Holds the information of the target system elements that are used to process the data at each stage.
  - **<Target\_System\_for\_Validation>** - Defines the target system where the validation of data takes place.
  - **<Target\_System\_for\_Export>** - Lists all the target systems.
  - **<Target\_System>** - Defines the target system name for exporting data. You can define more than one target system for exporting the data.
  - **<Suppress\_ENS>** - Defines whether to suppress the creation of objects with ENS in the target system for the data exported in this job. For example, if the value is set to True then the objects are created with ENS in the target system after export.
  - **<Target\_System\_Configuration>** - Defines the target system configuration for exporting the data. For example, ConfigurationTop~PlantA~Project1.
- 5 **<Import\_Definition\_And\_Files>** - Allows you to define the import definition name, file path for the input files, and prompted column information.

The following table defines the XML tags that are available in the control file:

XML Tag	Description
Job_Creation_Information	This consists of all the information required to trigger one or more jobs.
System_Name	Contains the name of the application server where the control file exists and the job is initiated
User_Name	Contains the username for logging into SmartPlant Foundation.
StartDateTime	Contains the date and time value, date format, and time zone.
Jobs	Contains job information for one or more jobs in it.
Job	Contains all the information required for a job.
Job_Description	Defines the job description used to identify the job in SDV Job Management.
Job_Definition_Name	Defines the name of the job definition used for the job.
Target_System_for_Validation	Defines the name of the target system where the data validation takes place.
Target_System_for_Export	Contains one or more target system name for the data export.
Target_System	Defines the name of the target system required for data export.
Suppress_ENS	Defines whether to suppress the creation of objects with ENS in the target system for the data exported in this job.
Target_System_Configuration	Defines the target system configuration for the exported data.
Import_Definitions_And_Files	Contains the import definition name, input file path, and prompted column information. This can also contain the information for one or more import definitions.
Import_Definition	Contains the import definition information.
Name	Defines the name of the import definition that is used for the job.

XML Tag	Description
Files_to_Import	Contains file information for one or more input files.
File	Contains the input file and the prompted column information.
File_Path	Defines the file path for the input file.
Has_Headers	Defines if the input files have headers and accepts the values: True or False.
Prompted_Values	Contains information for one or more prompted columns.
Prompted_Value	Contains the prompted column information.
Column_Name	Defines the name of the prompted column.
Value	Defines the value that is used for the prompted column at runtime.

## Trigger the job

You can trigger a job external to the Smart Data Validator Job Management module using the Control File. You must create a control file and place them in the **Control Files** folder. You can schedule the job for a later date by adding a start date and time in the control file.

The movement of the control file to the specified folder location depends on the status of the job creation. Each control file is appended with a timestamp and is moved to one of the following folders:

- Processing folder - Contains the control file when the job creation is in progress.
- Processed folder - Contains the control file when the job creation is completed without any warnings or errors.
- Failed folder - Contains the control file when the job creation either failed or has some warnings and errors.

**NOTE** The Log folder contains the log file that consists of detailed information about the job creation. The naming convention of the log file is: *[Control File Name]-[Timestamp].log*.

**IMPORTANT** Before following the steps below, ensure that the user group **SPFUsers** has access to the folder that contains the control files and the folder that contains the input files.

1. Log on to SmartPlant Desktop Client as superuser.
2. Select **File > Loader**.
3. In the **Loader** dialog box, click **Browse** and navigate to the path: *[Install folder]:\Program Files (x86)\Smart\SDV\[version]\SilentJobSetup\Model*.



4. Select **VTLSilentJob.xmlldr**, and click **Process**.
5. Click **Find > Administration > Schedulers**, and search for scheduler, **VTLSilentJobCreationScheduler**.
6. In the search results page, right-click **VTLSilentJobCreationScheduler**, and click **Update**.
7. Type the file path to where the control files are located.
8. Click **Finish**.
9. Start the scheduler **VTLSilentJobCreationScheduler** to trigger the job.

## SECTION 14

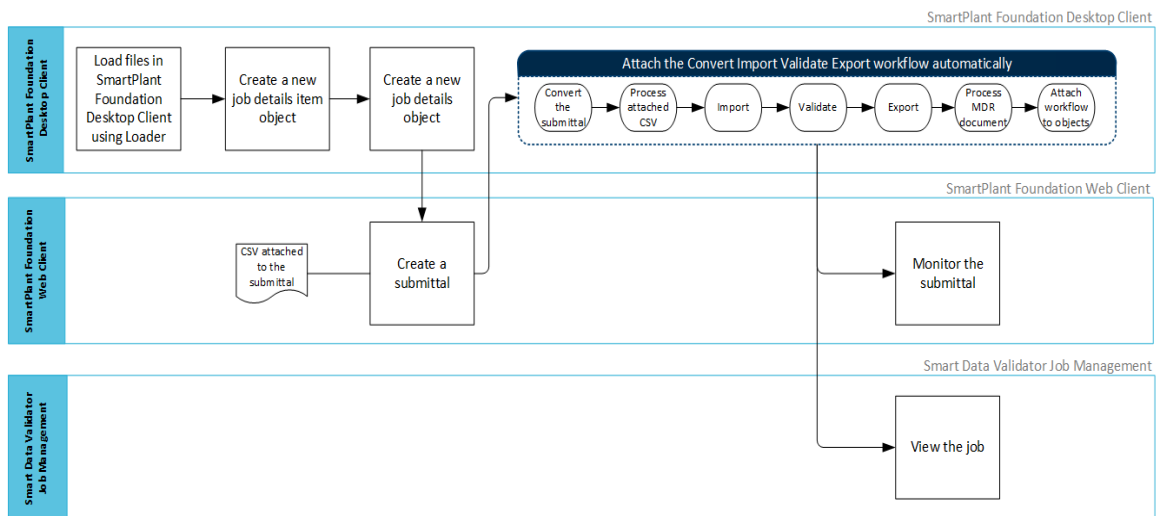
# Converting a SmartPlant Foundation Object to a Smart Data Validator Job

Smart Data Validator allows the conversion of SmartPlant Foundation objects, such as a submittal, into a Smart Data Validator job. Contractors typically supply submittals to owner-operators. A submittal could be a Master Document Registry (MDR), a Master Tag Registry (MTR), a Tag Document Registry (TDR), or other files.

You can convert a SmartPlant Foundation object into an SDV job using the Convert Import Validate Export workflow. A separate schema, import mappings, export mappings, and rule sets are delivered with Smart Data Validator. You must load these files using the Loader in the SmartPlant Foundation Desktop Client. After loading the schema, you must create a job details object. The job details object includes the classification type of the submittal, and a relationship is created between the job details object and the selected classification type.

The submittal itself is created using the SmartPlant Foundation Web Client. The Convert Import Validate Export workflow is specified as one of the arguments on the method used for creating the submittal. Based on the classification type of the submittal, the related job details object, along with the associated job details item objects, are selected to process it in the job conversion step of the workflow. After the submittal is converted into an SDV job, the import, validate and export steps are executed automatically in Smart Data Validator Job Management. You can monitor the submittal in the SmartPlant Foundation Web Client. The exported objects can be attached to a workflow specified in the final step of the Convert Import Validate Export workflow.

The following work process diagram illustrates the steps involved in converting a SmartPlant Foundation object to an SDV job using the Convert Import Validate Export workflow.



### What do you want to do?

- *Load files in SmartPlant Foundation Desktop Client using Loader* (on page 83)
  - *Create a new job details item object* (on page 84)
  - *Create a new job details object* (on page 85)
  - *Create a submittal* (on page 86)
  - *Attach the Convert Import Validate Export workflow automatically* (on page 86)
  - *Monitor the submittal* (on page 88)
  - *View the job* (on page 88)
- 

## Load files in SmartPlant Foundation Desktop Client using Loader

For an object, such as the SCLBSubmittal, to be used to process data in CSV files, the IVTLJob, IVTLValidationJob, IVTLJobExportDetails, IVTLJobImportDetails, and ISPFConfigurationItem optional interfaces must be added. This schema is required to convert any type of SmartPlant Foundation object, such as a submittal, to an SDV job.

1. In SmartPlant Foundation Desktop Client, select **File > Loader**.
2. **Browse** to *[installation folder]\SDV\2018\ProjectData\Model*, and then select **Loadfiles.xmlldr**.
3. Click **Process**. The Convert Import Validate Export workflow schema, the job details object schema, and the schema for converting the submittal object into an SDV job are loaded into the SmartPlant Foundation Desktop Client.
4. **Browse** to *[installation folder]\SDV\2018\ProjectData\Mappings*, and then select **LoadMappings.xmlldr**.
5. Click **Process**. The import mappings, export mappings, rule sets, job definitions, and job details are loaded into the SmartPlant Foundation Desktop Client.
6. Browse to *[installation folder]\SDV\2018\ProjectData\ProcessSteps*, and select **SCLBSubmittalProcessSteps.xml**.
7. Click **Process**.

The workflow process steps, **SetSubmittalCompletedDate** and **SetSubmittalStateToCOMPLETED**, are loaded into the SmartPlant Foundation Desktop Client.

### NOTES

- After loading the **SCLBSubmittalProcessSteps.xml** file, you can add two optional workflow steps, **SetSubmittalCompletedDate** and **SetSubmittalStateToCOMPLETED** to the Convert Import Validate Export workflow. These two steps indicate the completion date and status of the submittal. For more information on how to insert steps into a workflow, see *Workflows* in the *SmartPlant Foundation Desktop Client User's Guide*.

- If you are using a site created using SmartPlant Foundation 2016 R2 or earlier, you have to delete the job details objects created earlier and create new job details objects as well as new job details item objects. For more information, see *Create a new job details object* (on page 85) and *Create a new job details item object* (on page 84).
- You must update the job details objects after the upgrade, if you are upgrading to the latest version of SmartPlant Foundation.

## Create a new job details item object


The job details item object contains information to process the CSV files attached to the submittal.

You can split the basic CSV files attached to the submittal. You must specify a column name and values for splitting the CSV files as inputs in the **New SDV Job Details Item** dialog box. The split CSV files are then related to the specified import definitions for create and delete.

You can also unzip the zipped files uploaded with the submittal object. To skip referenced files when unzipping the files, you can specify the file column names in the **New SDV Job Details Item** dialog box. If the **Import definition for create action** or the **Column name used for file names (zip)** boxes are blank, the unzip process is skipped.

If you attach a raw property (inverted) CSV file, you must specify the characters which are included in the file name to distinguish it from the basic CSV file to avoid errors during validation.

1. In the SmartPlant Foundation Desktop Client, click **File > New > New SDV Job Details Item**.
2. In the **Main details** section, type a name and description for the job details object.
3. In the **Basic CSV format details** section, do the following:
  - Select a job definition from the **Job Definition for basic CSV** list.
  - Type an import definition name in the **Import definition for create action** box.
  - Type an import definition name in the **Import definition for delete action** box.
  - Type a column name in the **Column name used to split CSV file** box to split the CSV file.
  - Type the values in the **Value used to split create or update records** box to split the create or update records in the CSV file. Use a comma to separate multiple values.
  - Type the values in the **Values used to split delete records** box to split the delete records in the CSV file. Use a comma to separate multiple values.
  - Type a column name in the **Column name used for file names (zip)** box to skip any referenced files when unzipping the zip files.

 **TIP** This is required only if the submittal is an MDR.
4. In the **Inverted CSV format details** section, do the following:
  - Specify an import definition in the **Import definition for inverted format** box.
  - Specify the characters that are included in the CSV file name in the **Inverted CSV file name includes** box.

## Create a new job details object

The job details object contains information about the target system, classification type of the SmartPlant Foundation object, job details item object, job definition for the input CSV, and other details. A relationship is created between the job details object and the classification type of the SmartPlant Foundation object. Based on the classification type of the submittal, the related job details object is selected to process it in the job conversion step of the Convert Import Validate Export workflow.

★ **IMPORTANT** One job details record can be applied to multiple classifications, but only one job details record can be related to a classification.

To process the attached CSV, you must also specify a job definition and the target system name in the **New SDV Job Details** dialog box.

You can split the basic CSV file attached to the submittal. You must specify a column name and values for splitting the CSV file as inputs in the **New SDV Job Details** dialog box. The split CSV files are then related to the specified import definitions for create and delete.

You can also unzip the zipped files uploaded with the submittal object. To skip referenced files when unzipping the files, you can specify the file column names in the **New SDV Job Details** dialog box.

 **NOTE** If the **Import definition for create action** or the **Column name used for file names (zip)** boxes are blank, the unzip process is skipped.

If you attach a raw property (inverted) CSV file, you must specify the characters which are included in the file name to distinguish it from the basic CSV file to avoid errors during validation.

1. In the SmartPlant Foundation Desktop Client, click **File > New > New SDV Job Details**.
2. In the **Main details** section, type a name and description for the job details object.
3. In the **Common details** section, do the following:
  - Select a classification type from the **Classification type** list.
  - Select a job details item from the **Job Details Item** list.
  - Select a job definition from the **Job definition** list.
  - Specify a target system name in the **Target system name** box.
  - Select the **Suppress ENS** check box if you want the ENS to be suppressed.


## Create a submittal

The submittal is created in the SmartPlant Foundation Web Client. A submittal is used in the collaboration process for contractors to send documents and files to the owning company's system. The submittal is a control file that contains all the details of the submitted objects, including tags and any document file attachments. Each submittal has a defined classification that determines which workflow path the submittal process will follow.

When a submittal file is uploaded, the attached files are transferred to a specified location where the files can be scanned for viruses before being loaded into Smart Data Validator. Smart Data Validator validates the quality of the incoming data in the supplied files before they are loaded into the system.

There are four types of submittal:

- **Master Document Registry (MDR)** - Initiated by the contractor to send data in a master document register to the owner company.
- **Master Tag Registry (MTR)** - Initiated by the contractor to send data in a master tag register to the owner company.
- **Tag Document Relationships (TDR)** - Initiated by the contractor to send data on tag document relationships to the owner company.
- **MDR/MTR** - Initiated by the contractor to send data in a master document registry and master tag registry simultaneously to the owner company.

 **TIP** You can group a MDR, MTR, and TDR in one package to submit documents and tags together.

For more information, see *Create a submittal* in the *Smart Digital Asset Collaboration Help*.

## Attach the Convert Import Validate Export workflow automatically

When you create an object to process CSV files, such as a submittal, in the SmartPlant Foundation Web Client, the Convert Import Validate Export workflow automatically attaches to the object and runs through the workflow steps. The workflow uses existing and new process steps to accomplish the conversion of the object into a job.

- The SDVJobConversionStep converts the SmartPlant Foundation object into an SDV job.
- The SDVUnzipStep unzips the zipped files uploaded with the submittal. It reads all the file references in an MDR submittal and unzips any zipped files that are not referenced.
- The SDVSplitCSVFileStep splits the input CSV file.
- The Import, Validate, and Export steps validate the data imported into the staging area and export objects to the target system.
- The SDVMDRProcessStep creates a relationship between the MDR Document Master and other document masters in the submittal using the FDWMDRDocMasters relationship definition. This process step also instantiates the corresponding optional interfaces, based on the document classification.

---

## Converting a SmartPlant Foundation Object to a Smart Data Validator Job

---

- The final step, SDVAttachWorkflowStep, attaches a workflow, such as the document review workflow, to the exported objects. For more information, see *Attach a workflow to an object* in the *Intergraph Smart Digital Asset | Collaboration Configuration Guide*.

The following table provides the process steps used by the **Convert Import Validate Export** workflow.

Process step name	Description	Arguments
SDVJobConversionStep	Converts a SmartPlant Foundation object to a Smart Data Validator job object.	None
SDVUnzipStep	Reads all the file references in an MDR submittal and unzips any zipped files that are not referenced.	None
SDVSplitCSVFileStep	Splits input CSV file using properties defined in the SDV Job Details dialog box.	None
Import step	Runs the import of the supplied data to the staging area database using a defined job definition.	None
Validate step	Runs the validation rules on the imported data and generates a validation report. If there are validation errors, the validation report must be approved before the workflow can continue.	Arg1: VTLValidationReportFile
Export step	Transfers the data to the target system and loads the data to the database. This step starts automatically if there are no validation errors.	None
SDVMDRProcessStep	Creates relationships between the MDR Document Master and other document masters in the submittal using the FDWMDRDocMasters relationship definition.	Arg1: SDA document type Arg2: FDWMDRDocMasters relationship definition between the MDR document master and other document masters.

Process step name	Description	Arguments
SDVAttachWorkflowStep	Attaches a workflow to the SmartPlant Foundation object.	Arg1: EDG_SubmittalToRevision - edge definition between the objects in the workflow and the target objects.  Arg2: WD_SCLBQARreview - the name of the workflow.

## Monitor the submittal

You can use the **My Submittals** page in the SmartPlant Foundation Web Client to monitor the progress of submittals you created in the current scope and view their summary. For more information, see *My Submittals actions* in the *Smart Digital Asset Collaboration Help*.

## View the job

You can view the progress and statistics of each job in Smart Data Validator Job Management as it runs through each stage displayed under three job status tabs in the right-hand pane, **Progress**, **Summary**, and **Details**. For more information, see the *Smart Data Validator Job Management User's Guide*.

## Use planned revisions of documents with an MDR submittal

You can load the planned revision of a document before any data comes into the system, and relate it to plant equipment that is expected to be loaded into a document revision at a later date. You can use the planned revision document to compare the data that comes in with the actual revision.

When you initially attach the input CSV file to the submittal object and export it, the planned revision of the document is created at the target site with a status of RESERVED. Later, when the actual revision is exported, the status of the planned revision is changed to RESERVEDEPERSEDED automatically. However, if the actual revisions are deleted for any reason, the planned revision status is changed to RESERVED again.

### NOTES

- You cannot export a planned revision if an earlier revision is already available at the target site.
- You also cannot export planned and actual revisions in the same CSV file for creating or updating them at the target site.
- In object mapping, the IFDWPlannedDocRev optional interface must be added on the planned revision document to differentiate it from other revisions. For more information, see *Object mapping* (on page 37).



## APPENDIX A

# Functions

Functions are programmed by you to change the original input data to match a specific requirement expected when data is exported to a target system configuration. There are two types of columns that use functions:

- Computed columns
- Prompted API columns

## Computed column functions

The following computed column function examples show the original input and the expected output when data is imported to match a specific requirement in a target system configuration.

**NOTE** For information on using functions in the Computed column, see *Use the Computed column* (on page 71).

Function Name	Computed Column Sample String	Input Values and Output Results
<b>AddDefaultUOMIfMissing</b>	<code>func.AddDefaultUOMIfMissing("DEVTag", "DEVWeight", [TAG_WEIGHT])</code>	<b>INPUT</b> If the values for [TAG_WEIGHT] are "123 " and "456"  <b>OUTPUT</b> The outputs are 123 kg and 456 kg, as the default UOM is kg based on the SI in the schema
<b>Concat</b>	<code>func.Concat( { "A", "B", "C" } )</code>	<b>INPUT</b> If the values are "A", "B", and "C"  <b>OUTPUT</b> ABC
<b>ConvertToHash</b>	<code>func.ConvertToHash( [Name] )</code>	<b>INPUT</b> If the value for [Name] is "ABC Industries"  <b>OUTPUT</b> The output is "170C0B71E9F68CC294A6614BAE3E7347C208EA48"

Function Name	Computed Column Sample String	Input Values and Output Results
<b>DateTimeColumn</b>	<code>func.DateTimeColumn([Input Column], "Input value format", " ")</code>	<p><b>INPUT</b></p> <p>Converts date or date time to invariant format to work with SmartPlant Foundation and Smart Data Validator. For example, the Input Value Format</p> <p>"dd/MMM/yyyy-HH:mm:ss:fff"</p> <p><b>OUTPUT</b></p> <p>Output as invariant date format and time</p> <p>"yyyy/MM/dd-HH:mm:ss:fff"</p>
<b>DateTimeColumn</b>	<code>func.DateTimeColumn([Input Column], "Input value format", "TimeZone")</code>	<p><b>INPUT</b></p> <p>Converts date time to invariant format date time and time zone. For example, the Input Value Format</p> <p>"dd/MMM/yyyy-HH:mm:ss:fff"</p> <p><b>OUTPUT</b></p> <p>Output as invariant date time format and includes time zone. For example</p> <p>"yyyy/MM/dd-HH:mm:ss:fff", "GMT"</p>
<b>Decode</b>	<code>func.Decode([Input column], { "IfValueMatches1", "SetValueTo1", "IfValueMatches2", "SetValueTo2" }, "DefaultValue")</code>	<p><b>INPUT</b></p> <p>Looks at the input string and allows you to replace values; Func.decode([inputcol], { "N", "North", "S", "South", "E", "East", "W", "West", "Default" })</p> <p><b>OUTPUT</b></p> <p>If input is N it is replaced by North, and so on. If input is not N, S, W, or E it returns to the default value</p>
<b>Divide</b>	<code>func.Divide("100", "10")</code>	<p><b>INPUT</b></p> <p>Divides the numerator by the denominator; func.Divide(100,10)</p> <p><b>OUTPUT</b></p> <p>If 100 is the numerator and 10 is the denominator, 10 is the output</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>GetDefaultUOMIfMissing</b>	<code>func.GetDefaultUOMIfMissing("DEVTag", "DEVWeight", [TAG_WEIGHT_UOM])</code>	<p><b>INPUT</b></p> <p>If the values for [TAG_WEIGHT_UOM] is "kg" and ""</p> <p><b>OUTPUT</b></p> <p>The outputs are "kg" and "kg" assuming that the default UOM is kg</p>
<b>GetDocumentRevisionsCountFromTargetSystem</b>	<code>func.GetDocumentRevisionsCountFromTargetSystem("#SPF DesignDocMaster, .Name=[DocumentName]", "true")</code>	<p><b>INPUT</b></p> <p>Takes the QueryDef for the unique identification of the document master, and a true/false flag to determine the search configuration.</p> <p>For example, the input can be Name = Document1 and Ignore configuration = true.</p> <p><b>OUTPUT</b></p> <p>Returns the number of revisions available for Document1 in all configurations.</p> <p>If the Ignore configuration value is set to "false", then the count of revisions available for Document1 that are specific to the configuration selected during job creation is considered.</p>
<b>GetFileName</b>	<code>func.GetFileName([FilePath])</code>	<p><b>INPUT</b></p> <p>Returns the file name from a file path. For example, if the file path value is C:\Folder1\SubFolder1\File1.txt.</p> <p><b>OUTPUT</b></p> <p>File1.txt</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>GetJobDetails</b>	<code>func.GetJobDetails([PROPERTY_NAME])</code>	<p><b>INPUT</b></p> <p>Takes the property name from the job as input. For example, VTLJobWorkflowName.</p> <p><b>OUTPUT</b></p> <p>Returns the value associated with the specified property. For example, Import Validate Export.</p>
<b>GetMajorRevisionCodeFromTargetSystem</b>	<code>func.GetMajorRevisionCodeFromTargetSystem([REV_SCHEME], [REV_CODE])</code>	<p><b>INPUT</b></p> <p>Takes the document revision scheme and revision code. For example, REV_CODE=1A, REV_SCHEME=RS_Rev1A</p> <p><b>OUTPUT</b></p> <p>Returns the document major revision code specified in the input. For example, Major revision code = 1</p>
<b>GetMinorRevisionCodeFromTargetSystem</b>	<code>func.GetMinorRevisionCodeFromTargetSystem([REV_SCHEME], [REV_CODE])</code>	<p><b>INPUT</b></p> <p>Takes the document revision scheme and revision code. For example, REV_CODE=1A, REV_SCHEME=RS_Rev1A</p> <p><b>OUTPUT</b></p> <p>Returns the document minor revision code specified in the input. For example, Minor revision code = A</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>GetParentObjectClassDef</b>	<pre>func.GetParentObjectClassDef([Tag],[TagClass],[ParentTagName], {"SPXTagInstrument", "SPXTagPipingComponent"})</pre>	<p><b>INPUT</b></p> <p>Takes the object name, object class definition, parent object name, and possible parent object class definitions. For example, ObjectName = "Tag201"</p> <p>ObjectClassDef = "SPXTagEquipment"</p> <p>ParentObjectName = "Tag101"</p> <p>PossibleParentObjectClassDef = {"SPXTagInstrument", "SPXTagEquipment"}</p> <p><b>OUTPUT</b></p> <p>Returns the class definition of the parent object. For example, checks if Tag101 is provided in the CSV file and if yes, gets the ObjectClassDef value for Tag101. If not provided, queries for Tag101 and SPXTagInstrument in the target system; if found, returns SPXTagInstrument. If not found, queries for Tag101 and SPXTagEquipment; if found, returns SPXTagEquipment. If the object is still not found, a blank value is returned.</p>
<b>GetTargetSystemValueIfEmptyReturnDefault</b>	<pre>func.GetTargetSystemValueIfEmptyReturnDefault( "true","This is default tag description", "#DEVTag,.Name=[TAG_NAME]" , ".Description")</pre>	<p><b>INPUT</b></p> <p>Replaces input values with the value from the target system, and if the object identified by the QueryDef parameter is not available, returns the default value specified in the input string. For example, TAG_NAME = "Tag01"</p> <p><b>OUTPUT</b></p> <p>This goes to the target system and finds the DEVTag with the name "Tag01" and returns its description. If the DEVTag, "Tag01" is not available in the target system, then the description, "This is default tag description" specified in the input string is returned.</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>GetValueFromTargetSystem</b>	<code>func.GetValueFromTargetSystem("true", "#DEVTag, .Name=[TAG_NAME]", ".Description")</code>	<p><b>INPUT</b></p> <p>Replaces input values with the value from the target system. For example, TAG_NAME = "Tag01"</p> <p><b>OUTPUT</b></p> <p>This goes to the target system and finds the DEVTag with the name "Tag01" and returns its description. If "Tag01" is not found in the target system, then "NULL" value is returned.</p>
<b>IndexOf</b>	<code>func.IndexOf("ABCDEF", "C")</code>	<p><b>INPUT</b></p> <p>Finds the index of the search string within the input string. For example, <code>func.IndexOf("ABCDEF", "C")</code></p> <p><b>OUTPUT</b></p> <p>The output is 2 because counting begins at zero</p>
<b>InvertedFilter</b>	<code>func.InvertedFilter([PROPERTY_NAME], {"DateTime", func.DateTimeColumn([PROPERTY_VALUE], "dd-MM-yyyy", "UTC")})</code>	<p><b>INPUT</b></p> <p>Checks for the Property name "DateTime", in an Inverted CSV formatted file and if it matches the column name ("DateTime") then the DateTime column value is evaluated based on the defined function DateTimeColumn.</p> <p><b>OUTPUT</b></p> <p>Output as invariant date and time format</p> <p>"yyyy/MM/dd-HH:mm:ss:fff"</p>
<b>Join</b>	<code>func.Join({"T", "A", "G"}, ",")</code>	<p><b>INPUT</b></p> <p>Joins an array of strings separated by a user-defined separator. For example, <code>func.Join(["T", "A", "G"], ",")</code></p> <p><b>OUTPUT</b></p> <p>T,A,G</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>Left</b>	<code>func.Left([TAG_NAME], "3")</code>	<b>INPUT</b> Adds a number of input string characters to the output string starting from the left, matching the number specified. For example, [TAG_NAME] = PSS-105 <b>OUTPUT</b> PSS
<b>Left</b>	<code>func.Left([TAG_NAME], "5")</code>	<b>INPUT</b> Adds a number of input string characters to the output string starting from the left, matching the number specified. For example, [TAG_NAME] = PSS-105 <b>OUTPUT</b> PSS-1
<b>Length</b>	<code>func.Length([TAG_NAME])</code>	<b>INPUT</b> Length of the input string. For example, [TAG_NAME] = PSS-105 <b>OUTPUT</b> 7
<b>Minus</b>	<code>func.Minus("10", "2")</code>	<b>INPUT</b> Subtracts the second value from the initial value. For example, 10 - 2 <b>OUTPUT</b> 8
<b>Multiply</b>	<code>func.Multiply({"4", "2"})</code>	<b>INPUT</b> Multiplies two values. For example, 4 x 2 <b>OUTPUT</b> 8

Function Name	Computed Column Sample String	Input Values and Output Results
<b>PadLeft</b>	<code>func.PadLeft([TAG_NAME], "A", "10")</code>	<p><b>INPUT</b></p> <p>Takes the given input string [TAG_NAME] and adds padding to the left using a padding character (A) to match the string length given in the function (10). For example,</p> <p>TAG_NAME = PSS-105</p> <p><b>OUTPUT</b></p> <p>AAAPSS-105</p>
<b>PadRight</b>	<code>func.PadRight([TAG_NAME], "A", "10")</code>	<p><b>INPUT</b></p> <p>Takes the given input string [TAG_NAME] and adds padding to the right using a padding character (A) to match the string length given in the function (10). For example,</p> <p>TAG_NAME = PSS-105</p> <p><b>OUTPUT</b></p> <p>PSS-105AAA</p>
<b>RegexMatch</b>	<code>func.RegexMatch([TAG_NAME], "[A-Z]")</code>	<p><b>INPUT</b></p> <p>If uppercase characters in the tag name, returns = True, if no uppercase characters, returns = False. For example,</p> <p>TAG_NAME = TAG01</p> <p><b>OUTPUT</b></p> <p>TAG_NAME = TAG01 returns = True</p>
<b>RegexMatch</b>	<code>func.RegexMatch([TAG_NAME], "[^A-Z]")</code>	<p><b>INPUT</b></p> <p>If no uppercase characters in tag name, returns = True. If uppercase characters exist returns = False. (^ = string may not contain alphabetic characters A - Z). For example,</p> <p>[TAG_NAME] = tag01</p> <p><b>OUTPUT</b></p> <p>TAG_NAME = tag01 returns = True</p>



Function Name	Computed Column Sample String	Input Values and Output Results
<b>RegexReplace</b>	<code>func.RegexReplace([TAG_NAME], "\s+", " ")</code>	<p><b>INPUT</b></p> <p>Uses the second parameter to find the characters to replace and replaces them with the third parameter specified.</p> <p>[TAG_NAME] = "A string with too many spaces in"</p> <p><b>OUTPUT</b></p> <p>Where "\s+" means any number of spaces and we are replacing them with one space, so the output is "A string with too many spaces in"</p>
<b>Replace</b>	<code>func.Replace([TAG_NAME], "P", "E")</code>	<p><b>INPUT</b></p> <p>[TAG_NAME] = P-105</p> <p><b>OUTPUT</b></p> <p>E-105</p>
<b>Right</b>	<code>func.Right([TAG_NAME], "3")</code>	<p><b>INPUT</b></p> <p>Adds a number of input string characters to the output string starting from the right, matching the number specified</p> <p>[TAG_NAME] = PSS-105</p> <p><b>OUTPUT</b></p> <p>105</p>
<b>Split</b>	<code>func.Split("TAG_NAME", "_", "1")</code>	<p><b>INPUT</b></p> <p>Takes a string and splits it on the given character. The maximum number of strings that are returned is given by the integer value provided</p> <p><b>OUTPUT</b></p> <p>NAME</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>SplitAlphaNumericSequence</b>	<code>func.SplitAlphaNumericSequence([REV_CODE], "True", "1")</code>	<p><b>INPUT</b></p> <p>Takes a string value, a true/false flag to determine if the primary sequence should be returned and a default value to be used in case the value is empty. Splits the string based on the change in alphanumeric characters and returns the first or second part of the string depending on the true or false flag</p> <p>[REV_CODE]="A1"</p> <p><b>OUTPUT</b></p> <p>"A"</p>
<b>SubString</b>	<code>func.SubString(" [TAG_NAME]", "0", "3")</code>	<p><b>INPUT</b></p> <p>Takes part of a string, starting from a given starting index character number, and continues for a given number of characters</p> <p><b>OUTPUT</b></p> <p>TAG</p>
<b>Sum</b>	<code>func.Sum( {"1", "2", "3"} )</code>	<p><b>INPUT</b></p> <p>Adds the elements of an integer array, so 1 + 2 + 3</p> <p><b>OUTPUT</b></p> <p>6</p>
<b>ToLower (Lowercase)</b>	<code>func.ToLower( [TAG_NAME] )</code>	<p><b>INPUT</b></p> <p>[TAG_NAME] = PSS-105</p> <p><b>OUTPUT</b></p> <p>pss-105</p>
<b>ToUpper (Uppercase)</b>	<code>func.ToUpper( [TAG_NAME] )</code>	<p><b>INPUT</b></p> <p>[TAG_NAME] = pss-105</p> <p><b>OUTPUT</b></p> <p>PSS-105</p>

Function Name	Computed Column Sample String	Input Values and Output Results
<b>Translate</b>	<code>func.Translate([TAG_NAME], "TAG-", "tag")</code>	<b>INPUT</b> This takes the first character of the StringToReplace parameter and replaces it with the first character of the ReplacementString. It then takes the second character of the StringToReplace parameter and replaces it with the second character of the ReplacementString and so on. If the StringToReplace is longer than the ReplacementString then the extra characters are replaced with blanks [TAG_NAME] = TAG-PSS-123 <b>OUTPUT</b> tagPSS123
<b>Trim</b>	<code>func.Trim([TAG_NAME], {"EIS -"})</code>	<b>INPUT</b> [TAG_NAME] = EIS-PSS-105 <b>OUTPUT</b> PSS-105
<b>TrimEnd</b>	<code>func.TrimEnd([TAG_NAME], {"0"})</code>	<b>INPUT</b> [TAG_NAME] = 0001-PSS-10500 <b>OUTPUT</b> 0001-PSS-105
<b>TrimStart</b>	<code>func.TrimStart([TAG_NAME], {"0"})</code>	<b>INPUT</b> [TAG_NAME] = 0001-PSS-10500 <b>OUTPUT</b> 1-PSS-10500
<b>YMDColumn</b>	<code>func.YMDColumn([InputColumn], "Input value format")</code>	Takes date value of dd/MMM/yyyy and converts to yyyy/MM/dd <b>INPUT</b> 30/Jan/2014 <b>OUTPUT</b> 2014/01/30

## Prompted API column functions

The following examples show the entry in the Prompted API column and the results expected when data is imported to match a specific requirement in a target system configuration.

**NOTE** For information on using functions in the Prompted API column, see *Use the Prompted API column* (on page 75).

Function Name	Prompted API Sample Strings	Result
<b>GetEnumEntries</b>	<code>func.GetEnumEntries("SPFCountries")</code>	The prompted box contains the entries in the target system's SPFCountries enumeration, and displays the name, and stores the UID.
<b>GetObjectPropertiesByType</b>	<code>func.GetObjectPropertiesByType("PPMTag", "TAG_*", "Name", "UID")</code>	The prompted box contains the tags in the target system that have a name beginning with "TAG_", displays the name, and stores the UID.
<b>GetValueFromTargetSystemObjects</b>	<code>func.GetValueFromTargetSystemObjects("#SPFEquipmentClasses", "Description", "Name")</code>	The prompted box contains the equipment classifications in the target system, displays the description, and stores the name.
<b>GetValuesFromTargetSystemRelExpansion</b>	<code>func.GetValuesFromTargetSystemRelExpansion("+SPFConfigurationTree", "#SPFPlant, .Name='PlantA'", "Name", "UID")</code>	The prompted box contains the subconfigurations for PlantA in the target system, displays the name and stores the UID.

## Document functions

There are two types of computed columns predominantly for use with documents: **Target system computed columns** and **Non-target system computed columns**.

## Target system computed columns

The following computed columns use information from the target system so that the computed columns conforms to the target system.

### GetDocumentRevisionsCountFromTargetSystem

This computed column retrieves the number of revisions available for a document from the target system. It uses these parameters:

**QueryDef** – Used to uniquely identify the document master in the target system. You can specify the class definition of the document master and the document name.

**IgnoreConfiguration** - Used to specify whether to include all configurations in the query or not.

#### NOTES

- This QueryDef searches for class definitions that realize the interface ISPFDocumentMaster.
- If more than one document is identified by the QueryDef, then a cumulative count of all the revisions available for the identified documents is returned.

#### Example

In this example, the first parameter finds the object with the class definition SPFDesignDocMaster and name Document1. The second parameter searches in all configurations.

```
func.GetDocumentRevisionCountFromTargetSystem("#SPFDesignDocMaster,.Name='Document1'", "true")
```

#### Scenario

To configure a rule that would result in a validation failure message for a document having more than one version, do the following:

- Create an integer rule with minimum value set as zero and maximum value set as one. The **Allow Blank Values** option should be set to True.
- Create a computed column with the following function:

```
func.TrimStart(func.GetDocumentRevisionsCountFromTargetSystem("#SPFDesignDocMaster,.Name=[DOCUMENT_NUMBER]", "True"), "0")
```

### GetMajorRevisionCodeFromTargetSystem

Given a revision scheme and a revision code, the computed column retrieves the details for the revision scheme from the target system, then splits the major revision code and validates that it conforms to the criteria for the revision code.

For example,

```
func.GetMinorRevisionCodeFromTargetSystem([REVISION_SCHEME], [REVISION_CODE])
func.GetMajorRevisionCodeFromTargetSystem([REVISION_SCHEME], [REVISION_CODE])
```


### GetMinorRevisionCodeFromTargetSystem

Given a revision scheme and a revision code, the computed column retrieves the details for the revision scheme from the target system, then splits the minor revision code and validates that it conforms to the criteria for the revision code.

#### Errors

If an error occurs in either computed columns, the system displays the following error message:

```
ERROR resolving MINOR/MAJOR revision code REV_CODE using revision  
scheme REV_SCHEME
```

 **NOTE** This error is not used for any .Net exceptions. This error occurs when either the computed column produces an empty value, both the minor and sequences are numeric, or both the sequences are alphabetic.

For example:

```
func.GetMajorRevisionCodeFromTargetSystem("RS_REV01", "A1") = A  
func.GetMinorRevisionCodeFromTargetSystem("RS_REV01", "A1") = 1
```

### GetTargetSystemValueIfEmptyReturnDefault

This computed column retrieves the value of a property from the target system, and if the object specified by the QueryDef is not found in the target system, returns the default value specified in the input string. It uses the following parameters:

**IncludeHigherConfiguration** – Used to specify whether to include a higher configuration in the query or not.

**Default value** – Used to specify the default value if the object specified by the QueryDef is not found in the target system.

**QueryDef** – Used to find an object in the target system. You can specify the class definition of the object and any property values found on that object or a related object. Any number of criteria can be added by separating them with a comma. This parameter must be a constant string, but it can reference other columns within the string if you enclose each of those columns in brackets [ ]. For example, func.Concat([TestColumn1], [TestColumn2], [TestColumn3]).

**ReturnDef** - Used to describe the property to be returned from the target system. You can specify the relationships to which to navigate, followed by the property you want to retrieve. Any number of relationships can be navigated. This parameter must be a constant string.

#### NOTES

- You must use the target system schema in the QueryDef and ReturnDef.
- If multiple objects are found, either by the QueryDef or by navigating to the relationships, then multiple values are returned, each separated by a comma.

#### Example 1

In this example, the first parameter specifies whether to include items from a higher configuration or not. The second parameter specifies the default value to be returned if the object specified by the QueryDef is not found in the target system. The third parameter is specified to find the object with the class definition SPFDesignDocMaster and the UID Document1. The fourth parameter specifies to return the description of that object.

```
func.GetTargetSystemValueIfEmptyReturnDefault("false","This is default document master description","#SPFDesignDocMaster,.UID='Document1'", ".Description")
```

### Example 2

In this example, the same object is navigated using the SPFPrimaryClassification relationship from end 2 to end 1 to find the required UID property. If the object specified by the QueryDef is not found in the target system, then the default string, "Object is not found", is returned.

```
func.GetTargetSystemValueIfEmptyReturnDefault("false","Object is not found","#SPFDesignDocMaster,.UID='Document1'", "-SPFPrimaryClassification.UID")
```

### Example 3

In this example, a match on the name is used to find the object, instead of the UID. The name DOCMASTER\_NAME is used to match on from the column, instead of entering it manually. If the object is not found, then the default string, "Object is not found", is returned.

```
func.GetTargetSystemValueIfEmptyReturnDefault("false","Object is not found","#SPFDesignDocMaster,.Name=[DOCMASTER_NAME]", "-SPFPrimaryClassification.UID")
```

### Example 4

This example shows how multiple relationships can be navigated. -RelDef1+RelDef2.Name navigates RelDef1 from end2 to end1, and RelDef2 from end 1 to end 2. This returns the object name from the end of those relationships. If the object is not found, then the specified default string value is returned.

```
func.GetTargetSystemValueIfEmptyReturnDefault("false","Folder1 is not found","#SPFFolder,.Name='Folder1'", "-SPFFolderDocumentVersion+SPFFolderSupportItem.Name")
```

## GetValueFromTargetSystem

This computed column retrieves the value of a property from the target system. It uses the following parameters:

**IncludeHigherConfiguration** – Used to specify whether to include a higher configuration in the query or not.

**QueryDef** – Used to find an object in the target system. You can specify the class definition of the object and any property values found on that object or a related object. Any number of criteria can be added by separating them with a comma. This parameter must be a constant string, but it can reference other columns within the string if you enclose each of those columns in brackets [ ]. For example, func.Concat([TestColumn1], [TestColumn2], [TestColumn3]). If the object is not found in the target system, then "NULL" value is returned.

**ReturnDef** - Used to describe the property to be returned from the target system. You can specify the relationships to navigate to, followed by the property you want to retrieve. Any number of relationships can be navigated. This parameter must be a constant string.

## NOTES

- You must use the target system schema in the QueryDef and ReturnDef.

- If multiple objects are found either by the QueryDef or by navigating to the relationships, then multiple values are returned, each separated by a comma.

### Example 1

In this example, the first parameter specifies items at a higher configuration or not. The second parameter is specified to find the object with the class definition SPFDesignDocMaster and the UID Document1. The third parameter specifies to return the Name of that object.

```
func.GetValueFromTargetSystem("false",  
"#SPFDesignDocMaster,.UID='Document1'", ".Name")
```

### Example 2

In this example, the same object is navigated using the SPFPrimaryClassification relationship from end 2 to end 1 to find the required UID property.

```
func.GetValueFromTargetSystem("false", "#SPFDesignDocMaster,.UID='Document1', '-SPFPrimaryClassification.UID")
```

### Example 3

In this example, a match on the name is used to find the object, instead of the UID. The name DOCMaster\_NAME is used to match on from the column, instead of entering it manually.

```
func.GetValueFromTargetSystem("false",  
"#SPFDesignDocMaster,.Name=[DOCMaster_NAME]", "-  
SPFPrimaryClassification.UID")
```

### Example 4

This example shows how multiple relationships can be navigated. -RelDef1+RelDef2.Name navigates RelDef1 from end2 to end1, and RelDef2 from end 1 to end 2. This returns the object name from the end of those relationships. If the object is not found, then the value, "NULL" is returned.

```
func.GetValueFromTargetSystem("false", "#SPFFolder,.Name='Folder1'", "-  
SPFFolderDocumentVersion+SPFFolderSupportItem.Name")
```

## Non-target system computed columns

This non-target system computed column performs a similar function as the previous computed columns, but does not use SPFRevisionScheme or the target system for validation.

### SplitAlphaNumericSequence

This computed column splits a value based upon a set of simple defined rules to determine the minor and major versions of documents. It can be used to split any value and is not limited to just document revisions.

```
SplitAlphaNumericSequence(Input String, ReturnPrimaryValue,  
DefaultValue)
```

For example,

```
func.SplitAlphaNumericSequence([REVISION_CODE], "True", "ERROR NO VALUE")
```



```
func.SplitAlphaNumericSequence([REVISION_CODE], "False", "ERROR NO  
VALUE")
```

**NOTE** The first example returns the first alphanumeric sequence (True) and the second returns the last alphanumeric sequence (False).

## Document function examples

### Example 1

RevisionCode = 1A, RevisionScheme = RS\_Rev1A

Returns the major revision code:

```
GetMajorRevisionCodeFromTargetSystem("RS_REV1A", "1A" ) = 1  
SplitAlphaNumericSequence("1A", "True", "") = 1
```

Returns the minor revision code:

```
GetMinorRevisionCodeFromTargetSystem("RS_REV1A", "1A" ) = A  
SplitAlphaNumericSequence("1A", "False", "") = A
```

### Example 2

RevisionCode = A01, RevisionScheme = RS\_RevA01

Returns the major revision code:

```
GetMajorRevisionCodeFromTargetSystem("RS_ RevA01", "A01" ) = A  
SplitAlphaNumericSequence("A01", "True", "") = A
```

Returns the minor revision code:

```
GetMinorRevisionCodeFromTargetSystem("RS_ RevA01", " A01" ) = 01  
SplitAlphaNumericSequence("A01", "False", "") = 01
```

### Example 3

RevisionCode = A, RevisionScheme = RS\_RevAlpha

Returns the major revision code:

```
GetMajorRevisionCodeFromTargetSystem("RS_ RevAlpha", "A" ) = A  
SplitAlphaNumericSequence("A", "True", "") = A
```

Returns the minor revision code:


```
GetMinorRevisionCodeFromTargetSystem("RS_ RevAlpha", " A" ) = A  
SplitAlphaNumericSequence("A", "False", "") = A  
SplitAlphaNumericSequence("A", "False", "Error no value") = Error no  
value
```

## Date and time functions

A date and time function string defines the date time value resulting from a formatting operation and defines the date time value required to convert one date and time string to the required date and time string format.

There are two computed column types for date time values:

- **DateTimeColumn** - This column is for the date and time for a specific time zone. The parameters are column, format of the input value, and time zone, for example, `func.DateTimeColumn([InputColumn], "Input value format", "TimeZone")`. Using this format the input value is converted to SPFDateTime format, `yyyy/MM/dd-HH:mm:ss:fff`.  
  
For example, the input value is `12/06/2015-09:00:00` and the input value format is `dd/MM/yyyy-HH:mm:ss`. Therefore, the input value is converted to SPFDateTime format, `2015/06/12-09:00:00:000`.
- **YMDColumn** - This column is for the YMD format. The parameters are the column name and the format of the input value, for example `func.YMD([InputColumn], "Input value format")`. Using this format, the input value is converted to the SPFDate format, `yyyy/MM/dd`.  
  
For example, the input value is `dd/MM/yyyy` format (`12/06/2015`), which is converted to SPFDate format that is `yyyy/MM/dd` (`2015/06/12`).

 **NOTE** The DateTime rule should only be configured on YMD or DateTime computed columns.

## ConvertToHash function

The ConvertToHash function can be used to generate a unique hash code for objects when you have case-sensitive data in your CSV file, but you are exporting to a case-insensitive database. The ConvertToHash function takes the input string and returns a capitalized hash value using the Secure Hash Algorithm 1 (SHA-1) algorithm.

For a given object, the SHA-1 algorithm consistently returns a unique alpha-numeric value known as the hash code for that object.

For example, you may have two company names in your CSV file, ABC Industries and Abc Industries, each with different capitalizations. If you are exporting the CSV file to a case-insensitive database, the export will fail. In this case, you can use the ConvertToHash function to map the ConvertToHash computed column to the SPFCaseSensitiveHashCode property for the object.

```
func.ConvertToHash("ABC Industries") returns
170C0B71E9F68CC294A6614BAE3E7347C208EA48
```

```
func.ConvertToHash("Abc Industries") returns
98761FC6EB12227F5367A34379998F8055373E1A
```

### Mappings for new and existing objects

Two separate schema files are delivered with Smart Data Validator, which you can load into the system using the SmartPlant Foundation Loader. They can be found at: `[drive]:\Program Files (x86)\Smart\SDV\2018\Model\CaseSensitivitySchema`.

- `ISPFCaseSensitiveObject.xml` – Comprises the interface definition, `ISPFCaseSensitiveObject`, and the property definition, `SPFCaseSensitiveHashCode`.

- HelperSchema-To-Imply-ISPFCaseSensitiveObject.xml – This is a sample file to optionally imply and realize the class definitions that require the hash value. You can specify the object class definitions and interface definitions in this file and load it.

### NOTES

- During import mapping, you must select the ISPFCaseSensitiveObject optional interface for the object and the ConvertToHash computed column must be mapped to the SPFCaseSensitiveHashCode property.
- For new objects, the target system query definition in the export mapping must be defined as.SPFCaseSensitiveHashCode.
- For existing objects, the target system query definition in the export mapping can be defined as .Name. The query definition can be changed to .SPFCaseSensitiveHashCode after all the objects in the database are assigned the hash code.


## Computed column .NET methods

Microsoft .NET methods calculate some of the computed columns. The following table provides further details on how some computer columns function:

Function	Description
Concat	Concatenates a set of input strings.
Divide	Divides the first operand by the second operand and throws an exception if the divisor is 0.
IndexOf	Returns the character position of the value parameter if the specified string is found, or -1 if it is not found. If the value is empty, the returned value is 0 (zero).
Join	Returns a string that consists of the elements in the value delimited by the separator string. If the value is an empty array, the method returns an empty string "".
Left	Sets the startIndex parameter to 0. See Substring for further details.
Length	Returns the number of char objects in the current string object. This function does not return the number of Unicode characters, because a particular Unicode character might be represented by more than one char object (standard character).
Minus	Returns the difference between two double values.
Multiply	Calculates the product of its operands.

Function	Description
PadLeft	Returns a new string that is equivalent to this, but left-aligned and padded on the right with as many padding characters as needed to create a length of totalWidth. However, if totalWidth is less than the length of this, the method returns a reference to the existing instance. If totalWidth is equal to the length of this, the method returns a new string that is identical.
PadRight	Returns a new string that is equivalent to this instance, but left-aligned and padded on the right with as many padding characters as needed to create a length of totalWidth. However, if totalWidth is less than the length of this, the method returns a reference to the existing instance. If totalWidth is equal to the length of this value, the method returns a new string that is identical.
RegexMatch	Searches the specified input string for the first occurrence of the regular expression supplied in the pattern.
RegexReplace	Returns a new string that is identical to the input string, except that a replacement string takes the place of each matched string.
Replace	Returns a string that is equivalent to the current string, except that all instances of oldValue are replaced with newValue. If oldValue is not found in the current instance, the method returns the current instance unchanged. This function throws an exception if oldValue null is empty or null.
SubString	Returns a string object equivalent to the substring of the length specified in the length parameter, beginning at the starting index number in the current String object.  -or-  Returns an empty field if the startIndex parameter is equal to the length of the current object and length is 0 (zero).
Sum	Returns the sum of the two operands.
ToLower (Lowercase)	Returns the current string in lowercase.
ToUpper (Uppercase)	Returns the current string in uppercase.
Trim	Passes a single character. Removes all occurrences of the specified character from the beginning and end of the string.
TrimEnd	Passes a single character. Removes all occurrences of the specified character from the end of the current string object.

Function	Description
TrimStart	Passes a single character. Removes all occurrences of the specified character from the beginning of the string object.

 **NOTE** For more information on Microsoft .NET methods and their operators, see <https://msdn.microsoft.com>.

## APPENDIX B

# Unique Keys and Query Definitions

By default, Smart Data Validator identifies objects in the target system using the UID constructed in the staging area and the class definition specified in the export map. Sometimes, however, the UIDs for a specific type of object in a target system cannot be constructed from the data provided in a CSV file. In that case, you can use a target system unique key or a query definition to specify how to identify the object in the target system.

**CREATE STAGE OBJECT TO TARGET OBJECT MAP**  
Provide Stage Object To Target Object mapping details

**Stage Object**

Local

Search local system

Selected stage object: \*

**Target Object**

Target System Free Text

Select target system

Connect

Selected target object: \*

**Details**

Description

Object Weighting\* 0

Target System Unique Key

Target System Query Definition

Save Cancel

## Target system unique keys

When a unique key definition has been specified in SmartPlant Foundation, Smart Data Validator can then identify objects in the target system. You can specify the unique key for the object type and enter it in the **Target System Unique Key** box when you create or edit the stage object to target object mapping.

The unique key definition is specified in the same way as in SmartPlant Foundation, but with a few changes.

- Any properties used must be prefixed with a full stop.
- Any environment variables must be surrounded by @ symbols
- Any constants must be surrounded by double quotes"". Properties and relationships must be Smart Data Validator properties and they must have an export map setup.

For example,

- The Unique Key Definition TG,Name becomes "TG",.Name.

- The Unique Key Definition TG,CURRENTCONFIG,Name becomes "TG",@CURRENTCONFIG@,.Name.
- The Unique Key Definition TG,CURRENTCONFIG,+SPFTagArea.Name becomes "TG",@CURRENTCONFIG@,+SPFTagArea.Name.

#### NOTES

- A formatting check is performed as you enter the unique key details. A green tick indicates that the data format is correct.
- A schema check is performed as you save the stage object to target object mapping.
- The supported environment variables are: @CURRENTCONFIG@, @CONFIGLEVEL1@, @CONFIGLEVEL2@, @CONFIGLEVEL3@, @CONFIGLEVEL4@, and @CONFIGLEVEL5@. These variables are not case sensitive.

## Target system query definitions

If the target systems do not have reliable unique keys setup, then a target system query definition can be used. The query definition is entered in the **Target System Query Definition** box as a series of properties, which can identify the object in the target system along with the class definition. These properties can be retrieved by navigating the relationships.

- The query definition is a comma separated list of these properties, with no limitation to number or order.
- Properties are specified by a period and then the VTL property definition UID.
- Properties that require navigating relationships should have the relationship information before the property. This is the relationship definition UID prefixed with a plus sign for navigation from end 1 to 2 and a minus sign for end 2 to 1
- If an object can be identified in the target system by just its name and class definition, the query definition would be .Name.
- If a tag object can be identified in the target system by its name, the name of the related area, and the class definition, then the query definition would be .Name,+VTLTagArea.Name.

#### NOTES

- Query definitions should only be used after unique key definitions or UID matching as they have better performance.
- You can specify the query definition parameters to be in any order, such as .Name,+VTLTagArea.Name can be +VTLTagArea.Name,.Name.
- A formatting check is performed as you enter the query definition details. A green check mark indicates that the data format is correct.
- A schema check is performed as you save the stage object to target object mapping.
- Environment variables are not allowed in the query definition.
- If both a unique key and query definition are specified, then Smart Data Validator will use the unique key to identify the objects in the target system. If neither are specified then Smart Data Validator will use UIDs to match the target system.

- If you export from Smart Data Validator leaving the Unique Key and Query definition fields blank, the UIDs in the staging area are used to match the UIDs in the target system. Sometimes, however, the UIDs for a specific type of object in a target system cannot be constructed from the data provided in a CSV file. In that situation, you can use the unique key and query definition as an alternative to identify those objects.



## APPENDIX C

# Validation Rules and Actions

The following table lists all the validation rules supplied with Smart Data Validator and the actions that invoke them.

RULE TYPE	ACTIONS										
	Check Exist	Create Update	Update	Delete	No Action	Terminate	Check Staging File Exists	Check Target System File Exists	Compare Create	Compare Value	Rename
Check Claimed To Sub Config		O	O	O		O					O
Check Duplicate Issue Dates		O	O								
Check File Exists							P	P			P
Check For Cyclic Relationship		R									
Check Object Exists	O <sup>1</sup>		O <sup>1</sup>	O <sup>1</sup>		O <sup>1</sup>					O
Check File Exists for Document Revision		O	O								O
Check Is Claimable		O	O								
Check Is Revisable		O	O								
Check Parallel Claim		O	O	O		O					O
Check Revised to Sub Config		O	O	O		O					O
Check Revised to Parallel Config		O	O	O		O					O
Check Properties Against Schema		O,RP	O						RP	RP	
Check Relationship Exists	R			R		R					
Check Unique Key		O									O <sup>3</sup>
Check Value and UOM		P							P		P
Compare Value									P	P	P
Compare All Property Values for Objects									P	P	P
Date Time		P, RP							P, RP	P, RP	O <sup>3</sup> P
Double		P, RP							P, RP	P, RP	O <sup>3</sup> P
Integer		P, RP							P, RP	P, RP	O <sup>3</sup> P
Regular Expression		P, RP							P, RP	P, RP	O <sup>3</sup> P
Staging System Cardinality	O <sup>2</sup>	O <sup>2</sup>	O <sup>2</sup>								
String Does Not Start With		P, RP							P, RP	P, RP	O <sup>3</sup> P
String Length		P, RP							P, RP	P, RP	O <sup>3</sup> P
String Not Equal To		P, RP							P, RP	P, RP	O <sup>3</sup> P
Target System Cardinality	O <sup>2</sup>	O <sup>2</sup>	O <sup>2</sup>								

**O** = Object **P** = Property **R** = Relationship **RP** = Relationship Property

**O<sup>1</sup>** = If the same object is found for Create Update locally, this is reported as True.

**O<sup>2</sup>** = The relationship must also have the Create Update action on it.

**O<sup>3</sup>** = The object Rename action forces the rule to evaluate against the new name if it was configured against the name property.

**★ IMPORTANT** Only one **Check Staging File Exists** property action or one **Check Target System File Exists** property action is allowed per object. If there is more than one, an error is raised and the export is blocked. For more information, see *Export checks list* in the *Smart Data Validator Job Management User's Guide*.

### NOTES

- The **Compare Value** rule ignores null values in the target system if the property action is set to **Compare Create**, but respects the null values if the action is set to **Compare Value**. This is because exporting with a **Compare Create** action requires blank target system values to be updated without altering any existing values.
- The **Check Exist** action is invoked for **Staging System Cardinality** and **Target System Cardinality** rules only when the user selects this action during a rule creation.

For more information on how to define validation rules and create rulesets, see *Define Validation Rules and Rulesets* (on page 56).

## Validation rule description

The following table lists all the validation rules and their descriptions supplied with Smart Data Validator.

Rule Name	Description
<b>Check Claimed to Sub Configuration</b>	Checks to see if a plant-level item has been claimed to the project.
<b>Check Duplicate Issue Dates</b>	Checks to see if there are revisions with duplicate issue dates for the same master document.
<b>Check File Exists</b>	Checks to see that a file exists in the specified path and that security has been set to configure access permissions.
<b>Check For Cyclic Relationships</b>	Checks to determine if the same object is set on both End 1 and End 2 of a relationship definition.
<b>Check Object Exists</b>	Checks to see if the object exists in the target system already.
<b>Check File Exists for Document Revision</b>	Checks to see if any file is attached to the document revision if the document revision is not a planned revision.
<b>Check Parallel Claim</b>	Checks to see if the item is claimed to a parallel project under the same plant.
<b>Check Is Claimable</b>	Checks to see if the data object is claimable into the SmartPlant Foundation collaboration project and prevents the object from being loaded.
<b>Check is Revisable</b>	Checks to see if the document is revisable to the SmartPlant Foundation collaboration project and prevents the document revision from being loaded.
<b>Check Revised to Sub Config</b>	Checks to see if a plant-level document has been revised to the project.

Rule Name	Description
<b>Check Revised to Parallel Config</b>	Checks to see if the document item has been revised to a parallel project under the same plant.
<b>Check Properties Against Schema</b>	Checks to see if the target property definitions are valid for the target object type or relationship type, and if the values are valid for these properties, including whether required values are present.
<b>Check Relationship Exists</b>	Checks to see if the relationship exists in the target system.
<b>Check Unique Key</b>	Checks to see if a specific unique key already exists elsewhere in the system and cannot be implicitly claimed. This check runs only on the target system and detects duplicates in the source files.
<b>Check Value and UOM</b>	Checks to see if the UOM of the object property is the same as the UOM configured in the target system. It also checks to see if the object property value is defined within a specified range of minimum and maximum values.
<b>Compare All Property Values For Objects</b>	Checks to see if all the staging system object property values are the same as the target system object property values.
<b>Compare Value</b>	Checks to see if the staging system property value is the same as the target system property value.
<b>Date Time</b>	Checks to see if an object property or relationship property must validate against some date and time, YMD, or DMY. Must be in invariant format.
<b>Double</b>	Checks to see if an object property value or relationship property value has a minimum and a maximum value.
<b>Integer</b>	Checks to see if an object property value or relationship property value is defined within a specified range by a minimum and maximum value.
<b>Regular Expression</b>	Checks to see if an object property value or relationship property value adheres to a specified pattern.
<b>Staging System Cardinality</b>	Checks for the cardinalities between the related objects.
<b>String Does Not Start With</b>	Checks to see that the value does not start with the given value.
<b>String Length</b>	Checks to see if the length of an object property or relationship property is within a specified minimum and maximum limit or adheres to a specified pattern.

Rule Name	Description
<b>String Not Equal To</b>	Checks to see that the value is not equal to the given value.
<b>Target System Cardinality</b>	Checks for the minimum and maximum cardinality values between related objects from the target system.

## NOTES

- The **Check Duplicate Issue Dates** rule cannot be used if the revision object is configured to be identified using a query definition.
- The **Check File Exists** rule validates the existence of the file specified in your CSV file by checking in your Smart Data Validator application server file system or your target system application file system, depending on the property action.
- The **Check File Exists for Document Revision** rule validates the document revision and checks if any file is attached to it, unless the document revision is a planned revision. In the case of a planned revision, the rule does not check for attached files.
- The **Check is Claimable** rule is applicable only when data objects are loaded into collaboration projects.
- The **Check is Revisable** rule is applicable only when document revisions are loaded into collaboration projects.
- The **Check Unique Key** rule can be used in collaboration projects to avoid loading of duplicate data for non-claimable items.
- The **Compare Value** rule ignores null values in the target system if the property action is set to **Compare Create**, but respects the null values if the action is set to **Compare Value**. This is because exporting with a **Compare Create** action requires blank target system values to be updated without altering any existing values.
- The **Check Revised to Sub Config** rule and **Check Revised to Parallel** rule are not auto generated in the supplied system, but they can be changed to auto generate using the system options.
- The **Regular Expression** rule allows you to create a regular expression by including object properties in the pattern. The object property text should exactly match the case of the object property in the database. Additionally, you must add an ampersand (&) around each property for it to be parsed. For example, the following syntax will search for two digits, followed by a hyphen, and then an object property: `^(\d\d-&OBJ.<Property Definition UID>&)$`.
- The **Check Properties Against Schema** rule checks for the disabled enums and unmapped properties and reports the results in the validation report. You can use the **Allow Disabled Enum Entries** check box in the **Edit Rule** dialog box to determine whether the disabled enums are to be reported in the validation report. The **Check for Unmapped Properties** check box in the **Edit Rule** dialog box allows you to check if any of the properties of an object are not mapped to the target system properties.
- The **Check Value and UOM** rule checks the UOM of the object property when the object property value is blank. You can use the **Allow Blank Values** check box to specify that the object property value is blank.

## Actions

When you define a column header in Smart Data Validator, you assign it an action. This action indicates what Smart Data Validator will do with the data imported from the related column in the CSV file. The kinds of operations supported varies, depending on whether the column header is mapped to an object, a property, relationship, or relationship property.

**NOTE** The list of actions available for a property or a relationship is determined by the action selected for the object. For more information, see *Export mapping actions* (on page 119).

### Object actions

Smart Data Validator supports the following object actions:


Action	Description
<b>Check Exists</b>	Checks that the object exists in the target system.
<b>Create Update</b>	Creates a new object and updates the existing object in the target system.
<b>Delete</b>	Deletes an object in the target system.
<b>No Action</b>	No action is taken on the object in the target system.
<b>Rename</b>	The object is renamed in the target system.
<b>Terminate</b>	The object is terminated in the target system.
<b>Update</b>	The object is updated in the target system.
<b>Validate Only</b>	No action is taken on the object in the target system, but if errors are found then they are propagated to the related objects.

### Property actions

Smart Data Validator supports the following property actions:

Action	Description
<b>Check Staging File Exists</b>	Checks that the staging property file exists in the staging area.
<b>Check Target System File Exists</b>	Checks that the target system property exists in the target system.
<b>Compare Create</b>	Invokes the compare value rule, which creates an error if the target system value is different. The target system property is only overwritten if the value is not set.

Action	Description
<b>Compare Value</b>	Invokes the compare value rule, which never changes the value in the target system.
<b>Create Update</b>	Creates a new property and updates the existing property in the target system.
<b>Delete</b>	Deletes a property in the target system.
<b>No Action</b>	No action is taken on the property in the target system.
<b>Terminate</b>	The property is terminated in the target system.
<b>Validate Only</b>	No action is taken on the property in the target system, but if errors are found then they are propagated to the related objects.

 **NOTE** Using Smart Data Validator, you cannot delete or terminate the required properties of an object that exists in the target system.

### Relationship actions

Smart Data Validator supports the following relationship actions:

Action	Description
<b>Check Exists</b>	Checks that the relationship exists in the target system.
<b>Create Update</b>	Creates a new relationship and updates the existing relationship in the target system.
<b>Delete</b>	Deletes a relationship in the target system.
<b>No Action</b>	No action is taken on the relationship in the target system.
<b>Terminate</b>	The relationship is terminated in the target system.

### Relationship property actions

Smart Data Validator supports the following relationship property actions:

Action	Description
<b>Create Update</b>	Creates a new relationship property and updates the existing relationship property in the target system.
<b>No Action</b>	No action is taken on the relationship property in the target system.

Action	Description
<b>Compare Create</b>	Invokes the compare value rule, which creates an error if the target system value is different. The target system relationship property is only overwritten if the value is not set.
<b>Compare Value</b>	Invokes the compare value rule, which never changes the value in the target system.

## Export mapping actions

Export mappings are restricted by what actions you select when creating the object, as there are permitted combinations of actions. For example, the actions available for the Delete class definition mapping are **Compare Value** and **No Action**.

The following table shows the supported combination of default actions available in Smart Data Validator.

OBJECT ACTION	PROPERTY ACTION							
	Create Update	Check Staging File Exists	Check Target System File Exists	Compare Create	Compare Value	Delete	No Action	Terminate
Check Exist		x	x		x		x	
Create Update	x	x	x	x	x		x	
Update	x	x	x	x	x	x	x	x
Delete					x		x	
No Action		x	x		x		x	
Terminate					x		x	
Rename	x				x		x	
Validate Only								x

OBJECT ACTION	RELATIONSHIP ACTION				
	Check Exist	Create Update	Delete	No Action	Terminate
Check Exist	x	x	x	x	x
Create Update	x	x		x	
Update	x	x	x	x	x
Delete	x		x	x	
No Action	x		x	x	x
Terminate	x			x	x
Rename	x			x	
Validate Only	x	x	x	x	x

RELATIONSHIP ACTION	RELATIONSHIP PROPERTY ACTION			
	Create Update	Compare Create	Compare Value	No Action
Check Exist			x	x
Create Update	x	x	x	x
Delete			x	x
No Action			x	x
Terminate			x	x

★ **IMPORTANT** Only one **Check Staging File Exists** property action or one **Check Target System File Exists** property action is allowed per object. If there is more than one, an error is raised and the export is blocked. For more information, see *Export checks list* in the *Smart Data Validator Job Management User's Guide*.



## Relationship cardinality rules

### Staging system cardinality

For this rule, select a class definition and a relationship definition for the rule, then select the direction of the relationship. Two Boolean properties are used to enforce the minimum and maximum cardinality of 1. If neither the minimum nor the maximum cardinalities are set, no validation takes place.

For example, if you want to enforce only one primary classification, set both properties to True.

- **Enforce min cardinality of 1**
- **Enforce max cardinality of 1**

Therefore, if no relationship definition exists and the enforce minimum is set to 1, an error message is generated. If multiple relationships for the relationship definition are found and the enforce maximum cardinality is set to 1, an error message is generated.

**NOTE** Unlike the other rule types, you have the flexibility to select one or more actions from the **Applicable Actions For Rule** list box for the **Staging System Cardinality** and **Target System Cardinality** rules.

### Target system cardinality

For this rule, select a class definition and a relationship definition to run the rule, and select the direction of the relationship. The cardinality information is defined by the relationship definition in the target system and is used to perform the validation against the data in the staging area.

**NOTE** Unlike the other rule types, you have the flexibility to select one or more actions from the **Applicable Actions For Rule** list box for the **Staging System Cardinality** and **Target System Cardinality** rules.

## Rename action in Smart Data Validator

SmartPlant Foundation Desktop Client supports the rename of objects that changes the name and updates the unique identifier (UID) and unique key when they contain the name. Smart Data Validator supports the ability to rename objects using object mapping, and the specific object action **Rename** has been created for this purpose.

The Smart Data Validator rename functionality relies on the UID definitions in the staging area matching the UID definitions in the target system. UID definitions must be set on all objects that Smart Data Validator exports from the staging area. When you create your mappings, you must set the correct UID in the import mapping by setting the UID definitions in any object mapping. For more information, see *Define Column Headers* (on page 33).

The primary intention of the rename mapping is to change the name, it is not recommended to use rename for updating properties. For documents, the rename functionality in SmartPlant Foundation renames the Master, all revisions, all versions, and updates the UIDs on all relationships.

For more information, see *Configuring unique identifiers* in the *How to Setup and Configure SmartPlant Foundation*.

## DEVTag rename mapping example

This is an example mapping for a Tag using the rename action, where the column names are OLDNAME, NEWNAME, AREA, and PLANTCODE.

Column Header - OLDNAME ^	
Column Type	Physical
Name	OLDNAME
Column Header Text	OLDNAME
Maps to Object DEVTag with action Rename ^	
Maps to Object	DEVTag
Object Action	Rename
Unique Identifier	PLANTCODE, AREA, OLDNAME
Prefix	TG
New Name Column	NEWNAME
Maps to Property Name with action CreateUpdate ^	
Maps to Property	Name
Property Action	CreateUpdate
Parent Objects	<input type="text" value="OLDNAME"/>

## Design document master rename mapping

This is an example mapping for a design document using the rename action, where the column names are OLD\_DOCUMENT, NEW\_DOCUMENT, and CONFIG.

Column Header - OLD_DOCUMENT_NUMBER ^	
Column Type	Physical
Name	OLD_DOCUMENT_NUMBER
Column Header Text	OLD_DOCUMENT_NUMBER
Maps to Object SPFDesignDocMasterSA with action Rename	
Maps to Object	SPFDesignDocMasterSA
Object Action	Rename
Unique Identifier	CONFIG, OLD_DOCUMENT_NUMBER
Prefix	DM
New Name Column	NEW_DOCUMENT_NUMBER
***	
Maps to Property Name with action CreateUpdate ^	
Maps to Property	Name
Property Action	CreateUpdate
Parent Objects	<input type="text" value="OLD_DOCUMENT_NUMBER"/>

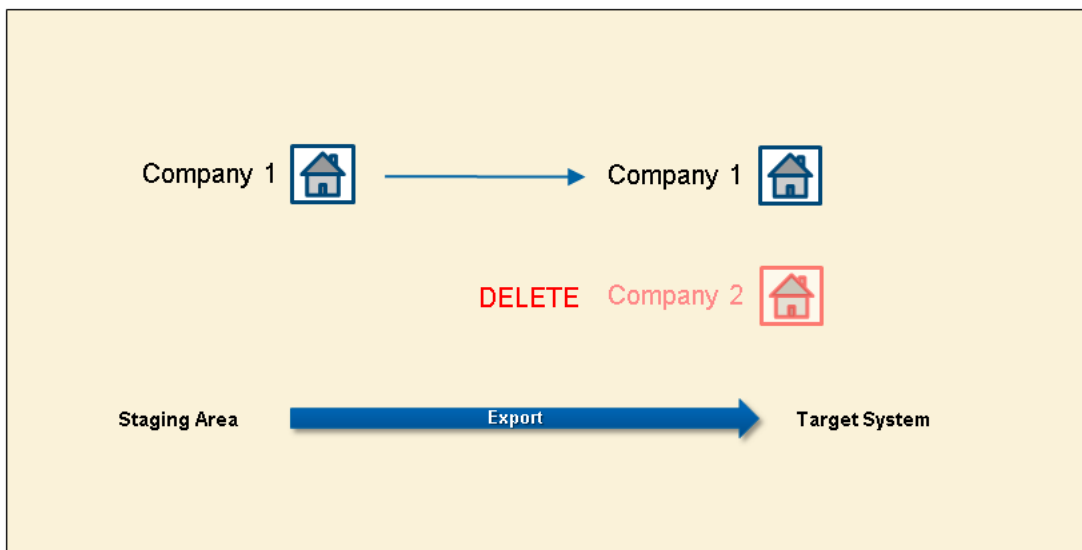
## APPENDIX D

# Understand Implicit Delete Functionality

The implicit delete functionality allows the system to delete objects and relationships that are no longer being supplied. The system determines which objects to delete from the database by running user configured rules, rather than deleting specified object types. The following scenarios better illustrate the implicit delete functionality.

## Deleting objects

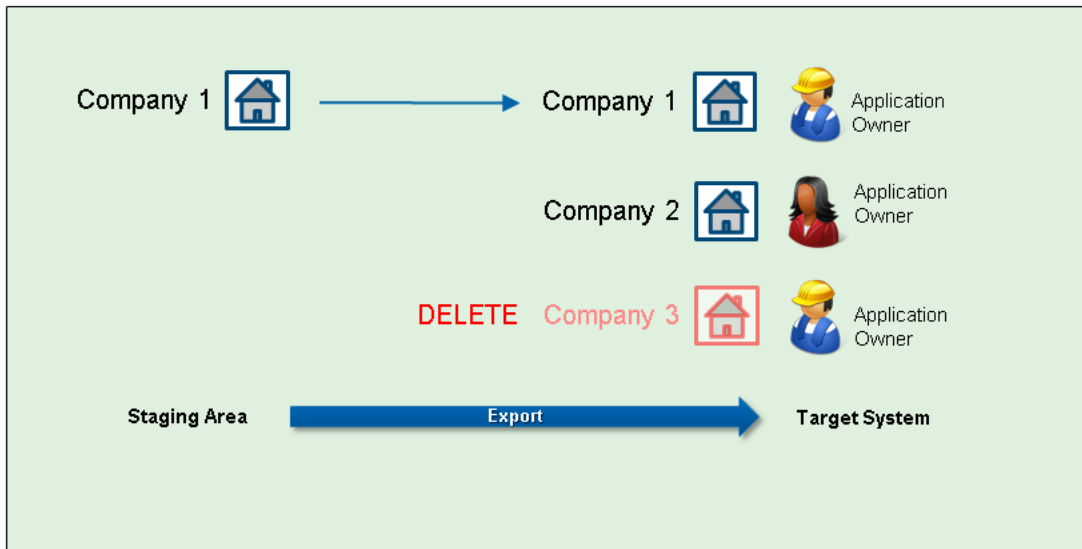
In this example for deleting an object, the CSV local data file in the staging area contains company 1, which is exported from the local staging area to the target system. The result of this export is that company 2 in the target system is set for deletion, as the CSV local data file does not contain company 2. The deletes report shows that company 2 is not present in the CSV local data file, so it is deleted implicitly. The job definition contains only the class definition as the grouping property.



## Deleting objects by property grouping

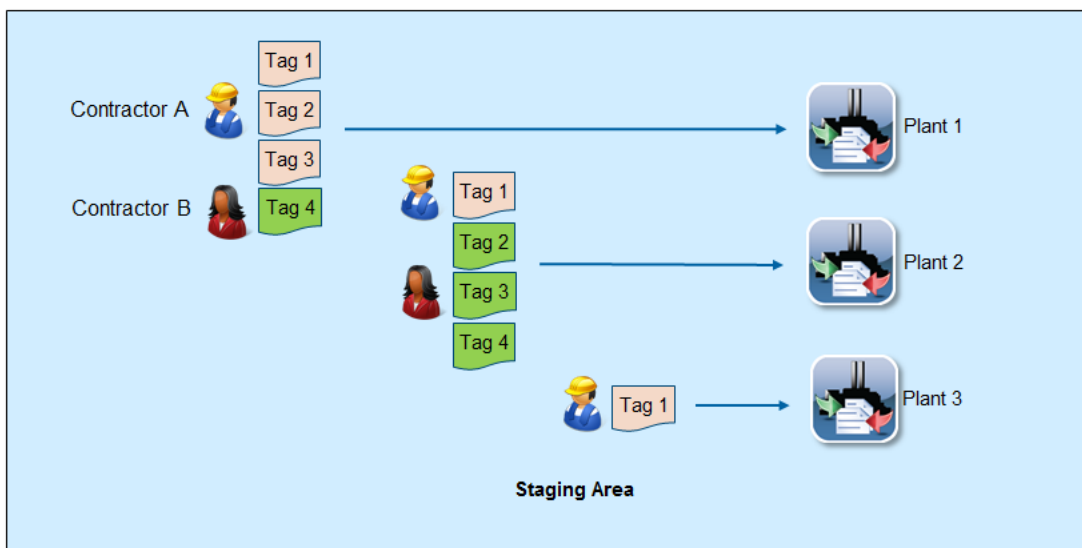
In this example, the CSV local data file in the local staging area is exported to the target system using the **Application Owner** grouping property. The result of the export is that company 3 is implicitly deleted in the target system, as the CSV local data does not contain company 3 and has the same application owner. However, company 2 is not deleted, even though it is not in the CSV local data file, because it is under a different application owner who controls the delete or

termination. The deletes report shows that company 3 is deleted implicitly. The job definition also contains the application owner as the grouping property.



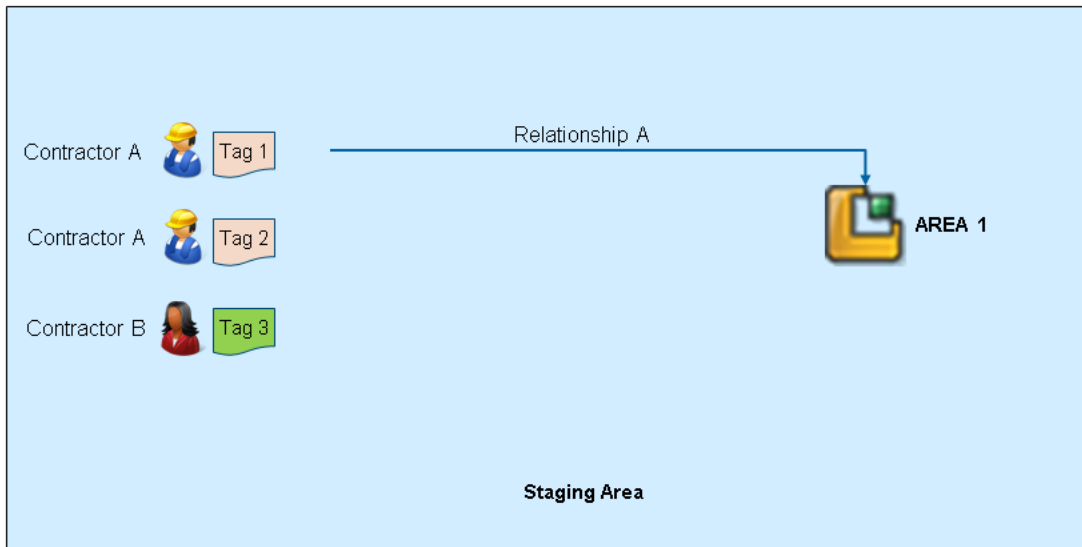
## Deleting objects by object and relationship grouping

In this example, the data is loaded from a CSV file into the staging area and that data is used to create the groups that query the target system for data to be removed. Therefore, the import of tags to the staging area is shared by both contractor A (pink tags) and contractor B (green tags). The **Application Owner** property identifies each contractor that supplied the tags that are related to plants. In this case, when a delete rule uses the plant relationship and the contractor **Application Owner** property as the delete groupings, any tags owned by contractor A and B, related to Plant 1 and Plant 2, and not in the CSV file are deleted. Finally, only the tags owned by contractor A are removed from Plant 3 and any tags that are not shown, but are related to other plants are not removed.



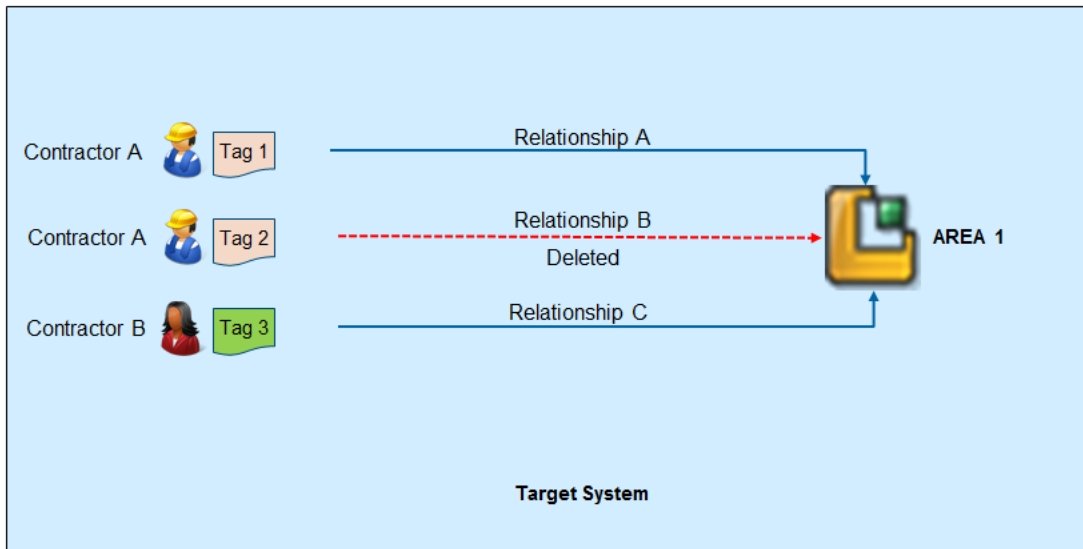
## Deleting relationships

The following example illustrates an implicit delete scenario where a rule has been configured to target the relationship between a tag and an area. The additional criteria for this rule is on the **Application Owner** property of the tag, which is used to establish that the contractor supplying the CSV file is the same contractor that owns the tag. Therefore in the Smart Data Validator staging area, contractor A is the application owner of Tag 1, which has a relationship to Area 1 in the target system. No other relationship exists at this time.



The next example shows the data as it exists in the target system. The implicit delete rule targets the relationship between the tag and area. The relationship A from Tag 1 was supplied in the CSV file and put into the staging area, where this relationship remains. The other two relationships B and C were not supplied in the CSV file. Contractor A is the same owner of Tag 2 in the staging area and the target system, so relationship B is deleted. However, the owner of Tag 3 in the staging area is contractor B, which is not the same owner of Tag 3 in the target

system. This means that the relationship C is not deleted as contractor B is not the owner of Tag 3 in the target system.



## APPENDIX E

# Environment Variables

The following is a list of descriptions for the environment variables found in Smart Data Validator.

**NOTE** Smart Data Validator does not provide an equivalent environment variable when the ConfigLevel1OrBlank environment variable is used as the unique ID definition or unique key definition for a class definition.

Object Properties	Description
ACCESSGROUPSFORUSERINCREATECONFIG	All valid access groups associated with the current create configuration/user (user's name).
ACCESSGROUPSFORUSERINQUERYCONFIG	All valid access groups associated with the current query configuration/user (user's name).
ALLDOMAINSFORUSER	All domains valid for the current user.
CALLEDFROM	Returns the string from the called client. For example, VBClient/WebClient based on a session parameter.
CALLEDFROMDIALOG	Returns the string from the called client based on a session parameter. For example, VBClient/WebClient.
CONFIGCREATE	Current create configuration string includes parent configurations. So if in project scope will include the plant.
CONFIGLEVEL	Current create configuration level. For example, 0, 1, 2, and so on.
CREATECONFIGSTRING	All current create configuration names in a comma separated list.
CREATECONFIGUID	Current create configuration UID.
CURRCONFIGLEVELNAME	Current configuration level name.
CURRENTDATE	Current date.
CURRENTTIME	Current time.

Object Properties	Description
HOSTNAME	Current Smart Data Validator server machine name.
JOBDESCRIPTION	Current Smart Data Validator job description.
JOBNAME	Current Smart Data Validator job name.
OWNINGGROUPSFORCREATECONFIG	All valid owning groups at the current create configuration level.
OWNINGGROUPSFORQUERYCONFIG	All valid owning groups at the current query configuration level.
OWNINGGROUPSFORUSERINCREATE CONFIG	All valid owning groups for the user at the current create configuration level.
OWNINGGROUPSFORUSERINQUERYCONFIG	All valid owning groups for the user at the current query configuration level.
QUERYCONFIG1, QUERYCONFIG2,3,4,5	Name of the query configuration at the specified level.
QUERYCONFIGLEVEL	Current query configuration level. For example 0,1,2 and so on.
QUERYCONFIGSTRINGS	All current query configuration names in a comma separated list.
QUERYCONFIGUIDS	All current query configuration UID.
SERVERURL	Current Smart Data Validator server address.
SESSIONID	Current Smart Data Validator session identification.
USERNAME	Current user's login user name.
USERORGANIZATIONNAME	Current user's company name.
USERDEFAULTCREATECONFIG	Current user's default create configuration.
USERUSERGROUPS	Current user query access groups.
USERUSERTYPE	Current user's types.



## Environment variable examples

The following is a list of examples of the environment variables found in Smart Data Validator.

Object Properties	Example
ACCESSGROUPSFORUSERINCREATECONFIG	UserAdmin,InstallationAdmin,SuperUserGroup
ACCESSGROUPSFORUSERINQUERYCONFIG	VTLDDataController,VTLDDataMapper
ALLDOMAINSFORUSER	InstD,SPFLLADesD,LLAD,SPFREFERENCE,METASHEMA,SPF,VTL
CALLEDFROM	VBClient
CALLEDFROMDIALOG	VBClient
CONFIGCREATE	PR_Project_1
CONFIGLEVEL	2
CREATECONFIGSTRING	PlantA,Project1
CREATECONFIGUID	PL_PlantA
CURRCONFIGLEVELNAME	Project1
CURRENTDATE	2015/01/30
CURRENTTIME	2015/01/30-10:38:12:991
EFFECTIVITYDATE	2015/01/30
HOSTNAME	SERVER-PROJECT1
JOBDESCRIPTION	EnvironmentVariableExamples
JOBNAME	ENVIRON1
OWNINGGROUPSFORCREATECONFIG	ADMIN,OPEN TO ALL, SDVADMIN
OWNINGGROUPSFORQUERYCONFIG	ADMIN,OPEN TO ALL, SDVADMIN
OWNINGGROUPSFORUSERINCREATECONFIG	ENGINEER,ADMIN,OPEN TO ALL, SDVADMIN

Object Properties	Example
OWNINGGROUPSFORUSERINQUERYCONFIG	ENGINEER,ADMIN,OPEN TO ALL, SDVADMIN
QUERYCONFIG1, QUERYCONFIG2,3,4,5	PlantA
QUERYCONFIGLEVEL	1
QUERYCONFIGSTRINGS	PlantA
QUERYCONFIGUID	PR_project1
SERVERURL	http://localhost/SPFServer
SESSIONID	aa1f956c-d28e-4fb7-83c4-c6e51d109618
USERNAME	SDVADMIN
USERORGANIZATIONNAME	Hexagon PPM
USERDEFAULTCREATECONFIG	PlantA
USERUSERGROUPS	ENGINEER,ADMIN,OPEN TO ALL, SDVADMIN
USERUSERTYPE	Full

## APPENDIX F

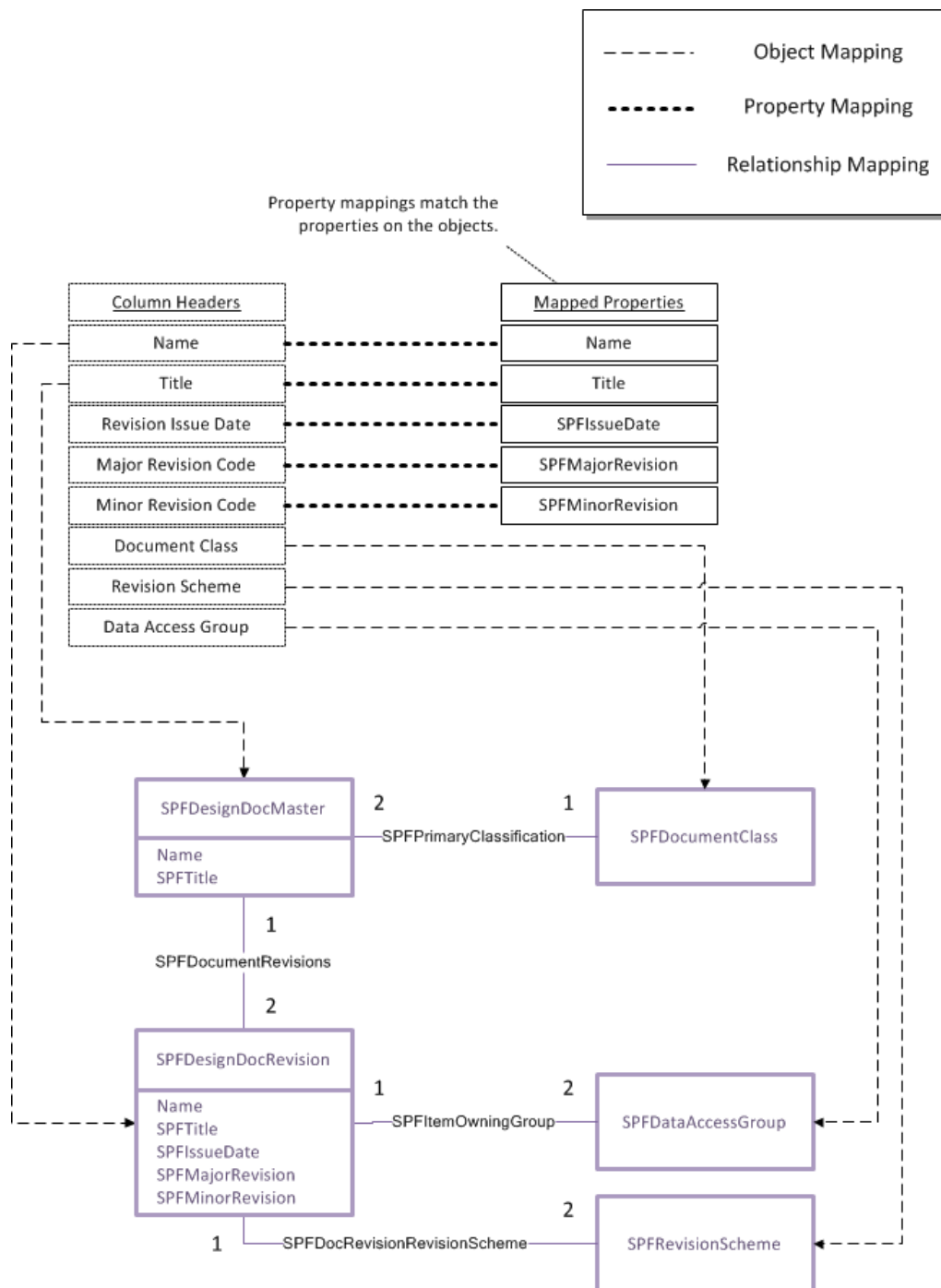
# Document Mappings

In SmartPlant Foundation, each document consists of three parts, a master, a revision, and a version. The supplied Smart Data Validator adapter for SmartPlant Foundation server allows the export of documents to the Smart Data Validator target system using the export mapping for document revisions and masters.

### NOTES

- The import mapping does not have versions, so a version is automatically created for each revision when exported.
- When documents are exchanged between EPCs and Owner Operators, documents may need to have matching version numbers in both the systems. You can use the **SPFDocVerion** property in your import mapping to specify the version number in your CSV file. A document version will be created in the target system with the version number specified in the CSV file.

- To map version object properties in the target system, you must map the properties to the revision class definition in the staging system. These properties are instantiated only on the version object in the target system.

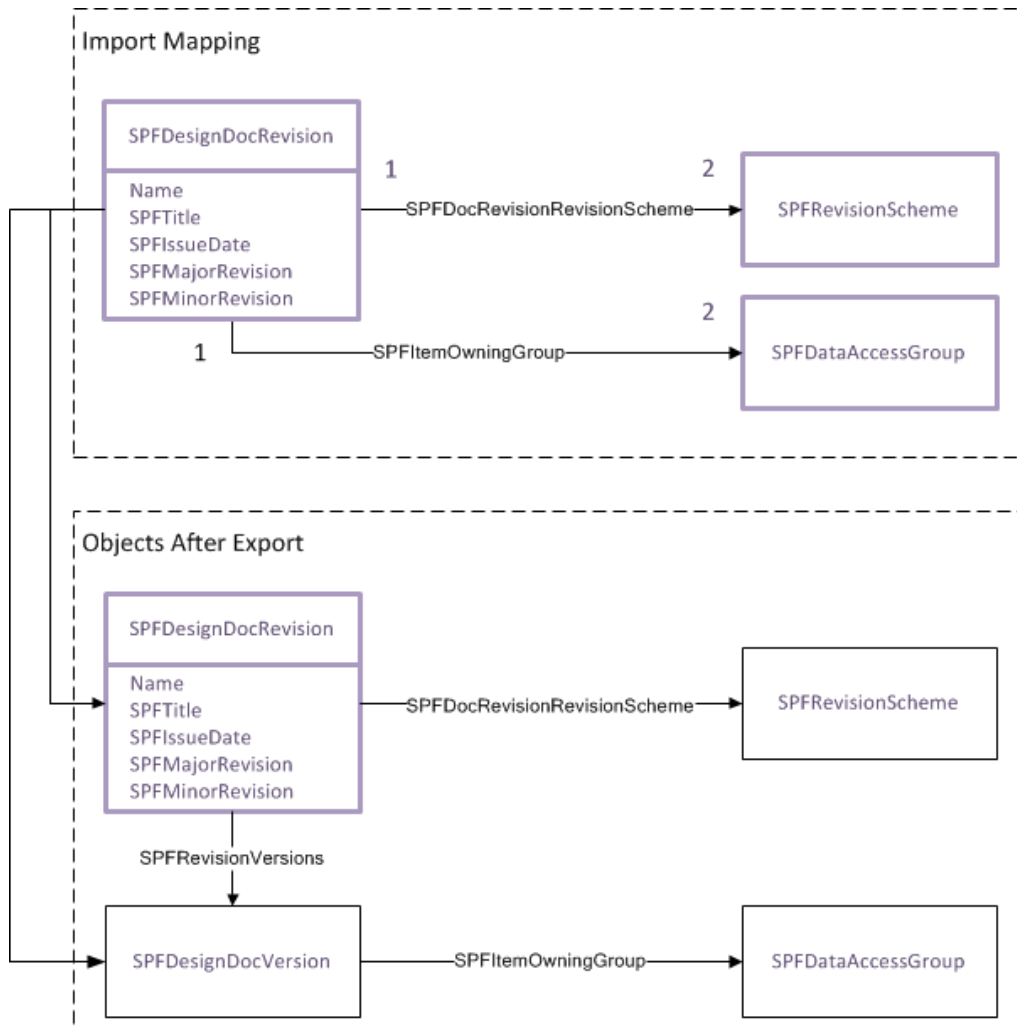


The relationships attached to revisions, as shown in the previous mapping image, are all mandatory and must be exported with the revisions. The **SPFPrimaryClassification** is also mandatory for a document master, and any master must have at least one

SPFDocumentRevisions relationship. When the target system has properties or relationships that need to be exported, the document version can be exported by mapping it to the revision. In this case, the target system adapter for SmartPlant Foundation puts those properties and relationships onto the latest version of the document revision.

★ **IMPORTANT** The export mapping must export a revision and the SPFDocumentRevisions relationship to the document master, or the revisions cannot be under the same document master for processing.

The following diagram shows how the target system adapter creates a revision and a version from the revision sent to the target system:



For information on documents using concurrent engineering, see *Documents with concurrent engineering* in the *Smart Data Validator Job Management User's Guide*.

## Import mapping examples

The following example mappings illustrate how to create an import mapping from the object mappings to the relationships.

### CSV file example

Name	Title	IssueDate	MajorRev	MinorRev	Class	RevScheme	OwningGroup
CT-111	Contract 111	02/01/2015	1	A	Contract Document	Rev1A	ENGINEER

### Object mappings example

Column Header	Column Type	Object Mappings	Object Name
Name	Physical	SPFDesignDocRevision	Document Name
Title	Physical	SPFDesignDocMaster	Document Title
Class	Physical	SPFDocumentClass	Document Class
Revision Scheme	Physical	SPFRevisionScheme	Revision Scheme
Owning Group	Physical	SPFDataAccessGroup	SPFDataAccessGroup

### Property mappings example

Column Header	Column Type	Property Mappings	Parent Objects
Name	Physical	Name	[Document Name], [Document Title]
Title	Physical	SPFDescription	[Document Name], [Document Title]
Issue Date	Physical	SPFIssueDate	[Document Name]
Major Revision	Physical	SPFMajorRevision	[Document Name]
Minor Revision	Physical	SPFMinorRevision	[Document Name]
Class	Physical	SPFDocumentClass	[Document Class]
Revision Scheme	Physical	SPFRevisionScheme	[Revision Scheme]
Owning Group	Physical	SPFDataAccessGroup	[SPFDataAccessGroup]

**NOTE** SPFIssueDate must be set as GMT and invariant format (the date time computed column can format and change the value).

## UID Definitions mappings example

Column Header	Object Mappings	Unique Identifier	Prefix
Name	SPFDesignDocRevision	[ <i>Document Name</i> ], [ <i>Major Revision Code</i> ], [ <i>Minor revision Code</i> ]	DM
Title	SPFDesignDocMaster	[ <i>Document Name</i> ]	DM
Class	SPFDocumentClass	[ <i>Document Class</i> ]	DC
Revision Scheme	SPFRevisionScheme	[ <i>Revision Scheme</i> ]	RS
Owning Group	SPFDataAccessGroup	[ <i>SPFDataAccessGroup</i> ]	DAG

**★ IMPORTANT** Smart Data Validator requires all UID definitions must be in place in the target system.

## Relationships mappings example

Relationship Mappings	End Column 1	End Column 2
SPFDocumentRevisions	Title	Name
SPFItemOwningGroup	Name	Owning Group
SPFDocRevisionRevisionScheme	Name	Revision Scheme
SPFPrimaryClassification	Class	Title

## Documents with concurrent engineering

Documents can be exported to different configurations, and the target system adapter for SmartPlant Foundation can handle revising the documents to sub configurations.

**★ IMPORTANT** When you are loading documents to different configurations:

- Updates to the master document are ignored when revising to a sub configuration; a warning is raised as a target system check.
- Duplicate masters cannot be created in a sub configuration.
- Revisions in a higher configuration cannot be updated from a sub configuration; this generates an error during a target system check.

- A revision that exists in a sub configuration cannot be updated from a higher configuration. The target adapter cannot attempt to merge the revision up to the target configuration; this must be done manually.

## Exporting multiple revisions

When exporting multiple revisions of the same document, the SmartPlant Foundation target adapter revises any existing revisions to create the latest revision from the CSV file. It then inserts any other revisions as historic revisions.

**NOTE** This behavior is different to revising a document incrementally in separate loads or loading multiple revisions in one load.

If the relationship copy behavior is defined so that the system copies the relationships on a revise, a relationship instantiated on Rev 1A, which is already in the database, is copied to Rev 2A, even if the relationship is not supplied in the CSV file. For more information on using revise, see *Revisions and Versions* in the *SmartPlant Foundation Desktop Client Help*.

### Example 1

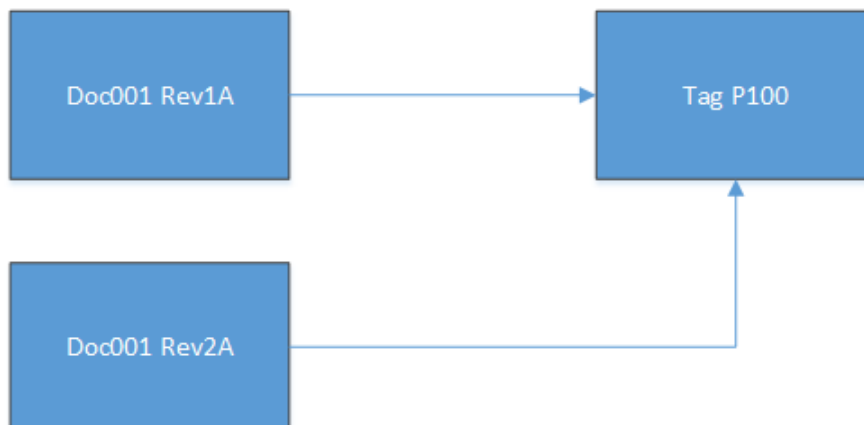
The existing data starts as follows:



The CSV file is laid out as follows:

DocNumber	MajorRev	Minor Rev	Tag
Doc 001	2	a	

As the existing revision has a relationship to the tag, the copy relationships ensure that new revisions have a relationship to the same tag.





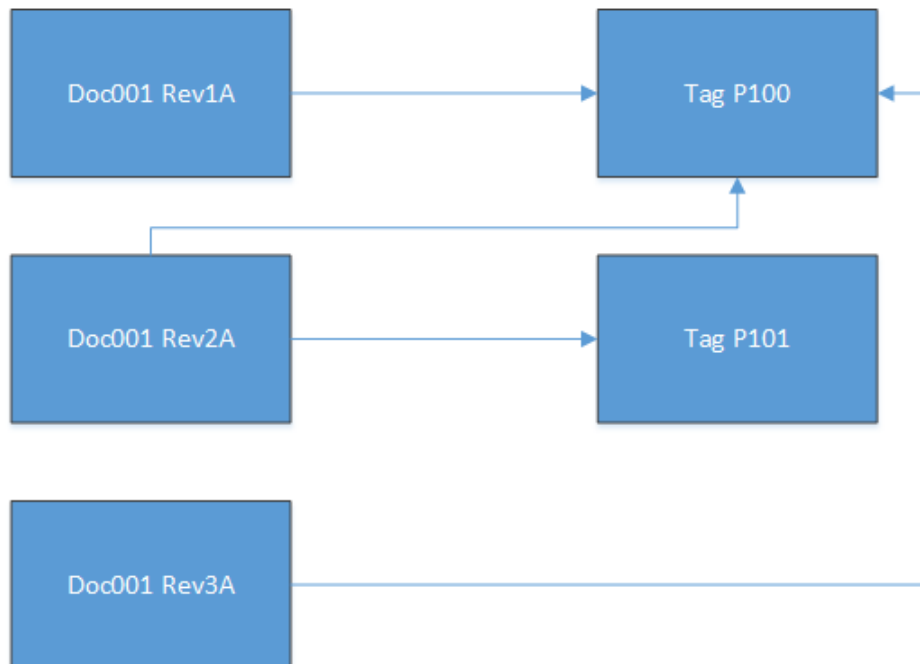
### Example 2

Alternatively, if two revisions are given in the CSV file working from the original starting data in the first example:

DocNumber	MajorRev	Minor Rev	Tag
Doc 001	2	a	P101
Doc 001	3	a	

Both new revisions are related to the original tag (P100), because the system copies relationships from the existing revision. However, only revision 2 is related to the new tag (P101), because that relationship was explicitly created in the load file for revision 2, but not for revision 3.

Smart Data Validator expects the relationship to be specified on every revision to which it is applicable. This behavior is consistent with interactively revising and then inserting historic revisions.



## Reservation and activation of document masters

To reserve document masters in the target system, the DocumentMaster class definition must realize the ISPFReservableDocumentItem and ISPFDocumentMaster interfaces as optional.

**NOTE** If the target system DocumentMaster class definition has a mandatory relationship with the ISPFDocumentMaster interface, and the import data does not contain document revision details, then the export fails.

Depending on the input data and the document mappings, the following actions happen in the target system:

- If the imported data contains information about document masters, then document masters are created. The document state is set to RESERVED\_INACTIVE.
- If the imported data contains information on document masters and revisions, then document objects are created with revisions related to the document masters. The document state is set to RESERVED.

In order to activate a reserved document, you must provide revisions for the existing document masters in the import data. The document revisions are created and get related to the document master.

## Mappings for published documents

Published documents may need to be exchanged frequently between EPCs and Owner Operators during the plant lifecycle. Smart Data Validator imports the published documents into a staging system, validates them, and exports them to a target system.

You can export published documents to a target system using the sample PUBLISHED\_DOC.csv file and the other attached files located at: *[drive]:\Program Files (x86)\Smart\SDV\2018\SampleData\PublishedDocuments*. Every published document has an associated tool signature, which must be exported to the target system before you export the document. You must create two import definitions using the PUBLISHED\_DOC.csv file as the input CSV file, one for the tool signature and another for the published document.

**NOTE** Instead of doing the import mappings manually, you can also load the PublishInstructionMappings.xmlldr load file located at *[drive]\Smart\SDV\2018\SampleData\PublishedDocuments*, using the Loader in the SmartPlant Foundation Desktop Client. The import mappings, export mappings, rule sets, and job definitions required for exporting the published documents are loaded into the SmartPlant Foundation Desktop Client and can be accessed using Smart Data Validator Administration.

## Tool signature mappings

The following example mappings illustrate how to create an import mapping for the tool signature using the SPFTEFToolSignature class definition. Use the PUBLISHED\_DOC.csv file to create the import mappings.

### CSV file example


ToolName	ToolDescription	OriginatorGUID	PIName	PIDescription	SPFTEFPI Schema Version	SPFTEFPI Revision Scheme Name	SPFTEFPI Document UID	SPFTEFPI Document Name	SPFTEFPI Document Description
SP3D-1	Desc: SP3D-1	21292849-643a-4000-b6f3-4b11ae383c81	PubIns_ PID- 02_A1_1	Desc for PID-02	4.02	RevA01	64ff1929-afe6-40b8-98ee-c4f4b838d900	PID-02	Desc for PID 02

### Object mappings example

Column Header	Column Type	Object Mappings	Action
Plant	Computed @Config1@	SPFPlant	Check Exists
SPFTEFPISignature	Computed func.GetTargetSystemValueIfEmptyReturnDefault("true", [OriginatorGUID], "#SPFTEFSignature,.SPFHandoverGUID=[OriginatorGUID]", ".Name")	SPFTEFSignature	Create Update

***Property mappings example***

Column Header	Column Type	Property Mappings	Parent Objects
ToolName	Physical	SPFTEFToolName	SPFTEFPISignature
ToolDescription	Physical	SPFTEFToolDescripti on	SPFTEFPISignature
OriginatorGUID	Physical	SPFHandoverGUID	SPFTEFPISignature

 **NOTE** The value in the OriginatorGUID column header is used to uniquely identify the tool signature on the target system.

***UID Definitions mappings example***

Column Header	Object Mappings	Unique Identifier	Prefix
Plant	SPFPlant	Plant	PL
SPFTEFPISignature	SPFTEFSignature	SPFTEFPISignature	TEFSIG

***Relationship mappings example***

Relationship Mappings	End Column 1	End Column 2
SPFTEFSignatureConfigurationItem	SPFTEFPISignature	Plant

**Published document mappings**

The following example mappings illustrate how to create an import mapping for published documents using the SPFTEFPublishInstruction class definition. Published documents have additional properties, such as tool signature, document UID, document version UID, revision and version information, which need to be mapped while creating the import definition and mappings.

The import mappings are mapped to the document metadata, tooldata, tombstone, and viewable files. These files are attached to the published documents and are related to the SPFTEFPublishedDocRevision revision object in the import mappings.

**CSV file example**

ToolName	ToolDescription	OriginatorGUID	PIName	PIDescription	SPFTEFPI Schema Version	SPFTEFPI Revision Scheme Name	SPFTEFPI Document UID	SPFTEFPI Document Name	SPFTEFPI Document Description
SP3D-1	Desc: SP3D-1	21292849-643a-4000-b6f3-4b11ae383c81	PubIns_ PID- 02_A1_1	Desc for PID-02	4.02	RevA01	64ff1929-afe6-40b8-98ee-c4f4b838d900	PID-02	Desc for PID 02

SPFTEFPI Document Version UID	SPFTEFPI Comp Schema	SPFTEFPI DocClassDef	SPFTEFPI DocTitle	SPFTEFPI Document Category	SPFTEFPI Document Type	SPFTEFPI Document Revision	SPFTEFPI Document Version	SPFTEFPI Document MajorRev	SPFTEFPI Document MinorRev
2312b275-c465-473f-b052-3f8bf79dcddb	PIDComponent	PIDDrawing	Title for PID-02	P&ID Documents	P&ID	A01	1	A	1

SPFTEFPI Owning Group Name	SPFTEFPI Publish Comment	ToolData	Tombstone	Metadata	Drawing	ToolData_Path	Tombstone_Path	Metadata_Path	Drawing_Path
OPEN TO ALL	Comments for PubIns_dc200886-21f2-403a-a278-8a950cc6b28c_A01_01	Tooldata_dc200886-21f2-403a-a278-8a950cc6b28c.xml	Tombstone_dc200886-21f2-403a-a278-8a950cc6b28c.xml	Metadata_dc200886-21f2-403a-a278-8a950cc6b28c.xml	Dwg_dc200886-21f2-403a-a278-8a950cc6b28c.pid	C:\Program Files (x86)\Smart\SDV\2018\SampleData\PublishedDocuments\SDV_Published_Files\PID-02\Tooldata_d4426701-6780-4f17-bde4-f960f01c6146.xml	C:\Program Files (x86)\Smart\SDV\2018\SampleData\PublishedDocuments\SDV_Published_Files\PID-02\Tombstone_d4426701-6780-4f17-bde4-f960f01c6146.xml	C:\Program Files (x86)\Smart\SDV\2018\SampleData\PublishedDocuments\SDV_Published_Files\PID-02\Metadata_d4426701-6780-4f17-bde4-f960f01c6146.xml	C:\Program Files (x86)\Smart\SDV\2018\SampleData\PublishedDocuments\SDV_Published_Files\PID-02\PID-02.pid

**Object mappings example**

Column Header	Column Type	Object Mappings	Optional Interfaces	Action
PIName	Physical	SPFTEFPublishe dFile	-	Create Update

Column Header	Column Type	Object Mappings	Optional Interfaces	Action
ToolData	Physical	SPFTEFPublishe dFile	ISPFTEFXMLFile ISPFTEFToolDataFile ISPFAlternateRenditionC omposition	Create Update
Tombstone	Physical	SPFTEFPublishe dFile	ISPFTEFXMLFile ISPFTEFInstructionsFile ISPFAlternateRenditionC omposition	Create Update
Metadata	Physical	SPFTEFPublishe dFile	ISPFTEFXMLFile ISPFTEFDocumentMetad ataFile ISPFAlternateRenditionC omposition	Create Update
Drawing	Physical	SPFTEFPublishe dFile	ISPFNavigationFileComp osition ISPFTEFViewableFile	Create Update
CompSchema	Computed <code>func.Concat([SPFTEFPICompSchema])</code>	CompSchema	-	Check Exists
Revision	Computed <code>func.Concat([SPFTEFPIDocumentName])</code>	SPFTEFPublishe dDocRevision	-	Validate Only
CompOwnGroup pName	Computed <code>func.Concat([SPFTEFPIOwningGroupName])</code>	SPFDataAccess Group	-	Check Exists

#### NOTES

- The VTL Target System Cardinality Validation Rule for the relationship SPFTEFXMLFileCompSchema on the class SPFTEFPublishedFile may fail because there is

no relationship created between the Drawing column header and the CompSchema column header. You can turn off the validation rule or optionally have the SPFTEFPublishedFile class definition associated with the Drawing column realize the ISPFTEFXMLFile interface. You can then create the SPFTEFXMLFileCompSchema relationship between the interfaces associated with the Drawing and CompSchema class definitions.

- Ensure the SPFTEFSignature is available in the target system.

### ***Property mappings example***

Column Header	Column Type	Property Mappings	Parent Objects
PIName	Physical	Name	PIName
SPFTEFPISignature	Computed func.GetValueFromTargetSystem("true", "#SPFTEFSignature,.SPFHandoverGUID=[OriginatorGUID]", ".Name")	SPFTEFPISignature	PIName
SPFTEFPRevisionSchemeName	Physical	SPFTEFPRevisionSchemeName	PIName
SPFTEFPIDocumentUID	Physical	SPFTEFPIDocumentUID	PIName
SPFTEFPIDocumentName	Physical	SPFTEFPIDocumentName	PIName
SPFTEFPIDocumentVersionUID	Physical	SPFTEFPIDocumentVersionUID	PIName
SPFTEFPICompSchema	Physical	SPFTEFPICompSchema	PIName
SPFTEFPISchemaVersion	Physical	SPFTEFPISchemaVersion	PIName
SPFTEFPIDocClassDef	Physical	SPFTEFPIDocClassDef	PIName
SPFTEFPIDocumentCategory	Physical	SPFTEFPIDocumentCategory	PIName

Column Header	Column Type	Property Mappings	Parent Objects
SPFTEFPIDocumentType	Physical	SPFTEFPIDocumentType	PIName
SPFTEFPIDocumentRevision	Physical	SPFTEFPIDocumentRevision	PIName
SPFTEFPIDocumentVersion	Physical	SPFTEFPIDocumentVersion	PIName
SPFTEFPIDocumentMajorRev	Physical	SPFTEFPIDocumentMajorRev	PIName
SPFTEFPIDocumentMinorRev	Physical	SPFTEFPIDocumentMinorRev	PIName
TargetWorkflowName	Constant	SPFTEFPITargetWorkflowName	PIName
SPFTEFPIOwningGroupName	Physical	SPFTEFPIOwningGroupName	PIName
Revision	Physical	Name	Revision
Tooldata	Physical	Name	ToolData
ToolData_Path	Physical	SPFLocalFileName	ToolData
Tombstone	Physical	Name	Tombstone
Tombstone_Path	Physical	SPFLocalFileName	Tombstone
Metadata	Physical	Name	Metadata
Metadata_Path	Physical	SPFLocalFileName	Metadata
Drawing	Physical	Name	Drawing
Drawing_Path	Physical	SPFLocalFileName	Drawing

**NOTE** For the TargetWorkflowName Constant column, enter the name of the target system workflow to which the document must be submitted.



***UID definitions mapping example***

Column Header	Object Mappings	Unique Identifier	Prefix
Name	-	-	-
ToolData	SPFTEFPublishedFile	SPFTEFPIDocumentVersionUID, ToolData	-
Tombstone	SPFTEFPublishedFile	SPFTEFPIDocumentVersionUID, Tombstone	-
Metadata	SPFTEFPublishedFile	SPFTEFPIDocumentVersionUID, Metadata	-
Drawing	SPFTEFPublishedFile	SPFTEFPIDocumentVersionUID, Drawing	-
CompSchema	CompSchema	CompSchema	-
Revision	SPFTEFPublishedDocRevision	Revision, SPFTEFPIDocumentMajorRev, SPFTEFPIDocumentMinorRev	-
CompOwnGroupName	SPFDataAccessGroup	CompOwnGroupName	DAG

***Relationship mappings example***

Relationship Mappings	End Column 1	End Column 2
SPFTEFXMLFileCompSchema	ToolData	CompSchema
SPFFileComposition	ToolData	Revision
SPFFileComposition	Tombstone	Revision
SPFTEFXMLFileCompSchema	Tombstone	CompSchema
SPFTEFXMLFileCompSchema	Metadata	CompSchema
SPFFileComposition	Metadata	Revision
SPFFileComposition	Drawing	Revision
SPFItemOwningGroup	Revision	CompOwnGroupName

## Export mappings example

An object weighting is required for each Stage to Target Object Map under the export mapping to load the data in the sample input file to the target system. Objects with lower weights are processed first. For example, set the object weightings to export the revision before the file it is attached to.

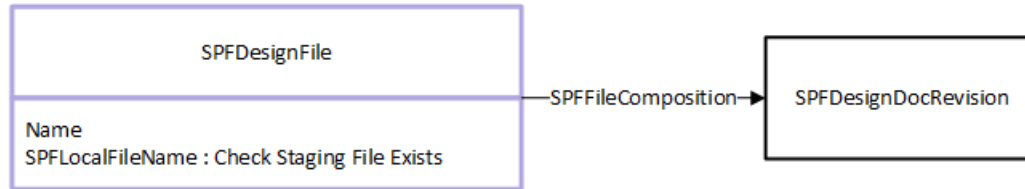
Use the following table to set the object weights:

Stage Object Name	Target Object Name	Object Weight
SDVSPFPlant	SPFPlant	10
SDVSPFDataAccessGroup	SPFDataAccessGroup	20
SDVSPFTEFSignature	SPFTEFSignature	30
SDVCompSchema	CompSchema	40
SDVSPFTEFPublishInstruction	SPFTEFPublishInstruction	50
SDVSPFPublishedDocRevision	SPFPublishedDocRevision	60
SDVSPFTEFPublishedFile	SPFTEFPublishedFile	70

## APPENDIX G

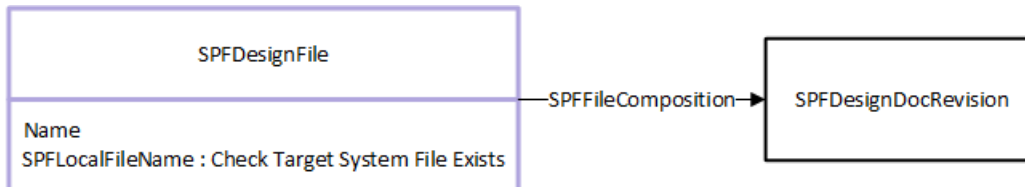
# Export File Mappings

When you export files to a target system, the import mapping maps a file object type, (for example SPFDesignFile) and creates a property with a **Check Staging File Exists** action, which is attached with the full file path of the file stored in SPFLocalFileName.

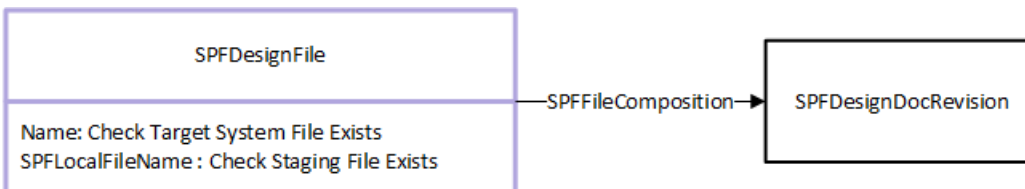


During the export process, all of the local staging files found are uploaded to the target system file services. The export process then creates all of the design files and saves the files into the vaults.

This mapping uses the **Check Target System File Exists** action, which ensures that the file is uploaded correctly using the path that is accessible by the target system server.



The target system loading will stop an export that contains an object with more than one property with a **Check Staging File Exists** or **Check Target System File Exists** action. This is because an object that represents a file can only have one file attached to it in SmartPlant Foundation. Additionally, the export process checks for any updates to an existing file object and warns the user that those updates are not supported. These update warnings are ignored so that the load can continue.



## CSV mapping example

The following CSV example mappings are an extension of the *Document Mappings* (on page 131). However, this section only shows the extensions to work with files and does not list the document mappings.

## CSV file example

Name	Title	IssueDate	MajorRev	MinorRev	Class	RevScheme	OwningGroup	File1	File2	FileShare
CT-111	Contract 111	02/01/2015	1	A	Contract Document	Rev1A	ENGINEER	a.txt	b.txt	\\Machine\share\

## Object mappings example

Column Header	Column Type	Object Mappings	Object Name
File 1	Physical	SPFDesignFile	File 1
File 2	Physical	SPFDesignFile	File 2

## Property mappings example

Column Header	Column Type	Property Mappings	Parent Objects
File1	Physical	None	n/a
File2	Physical	None	n/a
File1 Path	Computed (Concat, Fileshare, and File1)	SPFLocalFileName	File1
File2 Path	Computed (Concat, Fileshare, and File2)	SPFLocalFileName	File2

## UID Definitions mappings example

Column Header	Object Mappings	Unique Identifier	Prefix
File1	SPFDesignFile	[ <i>Document Name</i> ], [ <i>Major Revision Code</i> ], [ <i>Minor Revision Code</i> ], [ <i>Version Number</i> ], [ <i>File 1</i> ]	DM

Column Header	Object Mappings	Unique Identifier	Prefix
File2	SPFDesignFile	[ <i>Document Name</i> ], [ <i>Major Revision Code</i> ], [ <i>Minor Revision Code</i> ], [ <i>Version Number</i> ], [ <i>File2</i> ]	DM

**NOTE** The Version Number is a constant column, which contains the literal "1". This ensures that the uniqueness is the same as defined in the SmartPlant Foundation target system's document version UID Definition. This assumes that the document version UID Definition has the version number and the UID definition for the file object. For example:

SPFUIDDef="+SPFFileComposition.UID,Name"

## Relationships mappings example

Relationship Mappings	End Column 1	End Column 2
SPFFileComposition	[ <i>File1</i> ]	[ <i>Document Name</i> ]
SPFFileComposition	[ <i>File2</i> ]	[ <i>Document Name</i> ]

**NOTE** The revision must exist before the file can exist. The export weightings must be set to export the revision before the file.

## Files with custom target systems

If you have a target system that does not use SmartPlant Foundation, Smart Data Validator provides the ability to validate that the files exist in the target system.

Smart Data Validator attempts to send the physical files from the staging side to the target system when:

- The mapping contains properties with the **CheckTargetSystemFileExists** action.
- The mapping contains a **FileComposition** relationship between the file object and the object it is attached to.
- The target system returns **True** from the **SupportsPhysicalFileHandling** API.
- The target system has a **FileService** configured to receive the files.

## Creating new versions and copying files from the previous version

When the same file is available in the input CSV file for subsequent imports, the document version is created depending on the default settings in the **Job Definition** dialog box on the **Export Configuration** page.

### Example 1

**Scenario:** File1 already exists in the database. The **Allow Version Creation** option is set to True. Input CSV file with information about File1 is loaded.

**Result:** A new version V2 is created and the File1 from the input CSV file is attached to version V2.

### Example 2

**Scenario:** File1 already exists in the database. The **Allow Version Creation** and **Copy Files from the Previous Version** options are set to True. Input CSV file with information about File2 is loaded.

**Result:** A new version V2 is created. The File1 is copied from the previous version (V1) and the File2 are both attached to version V2.

### Example 3

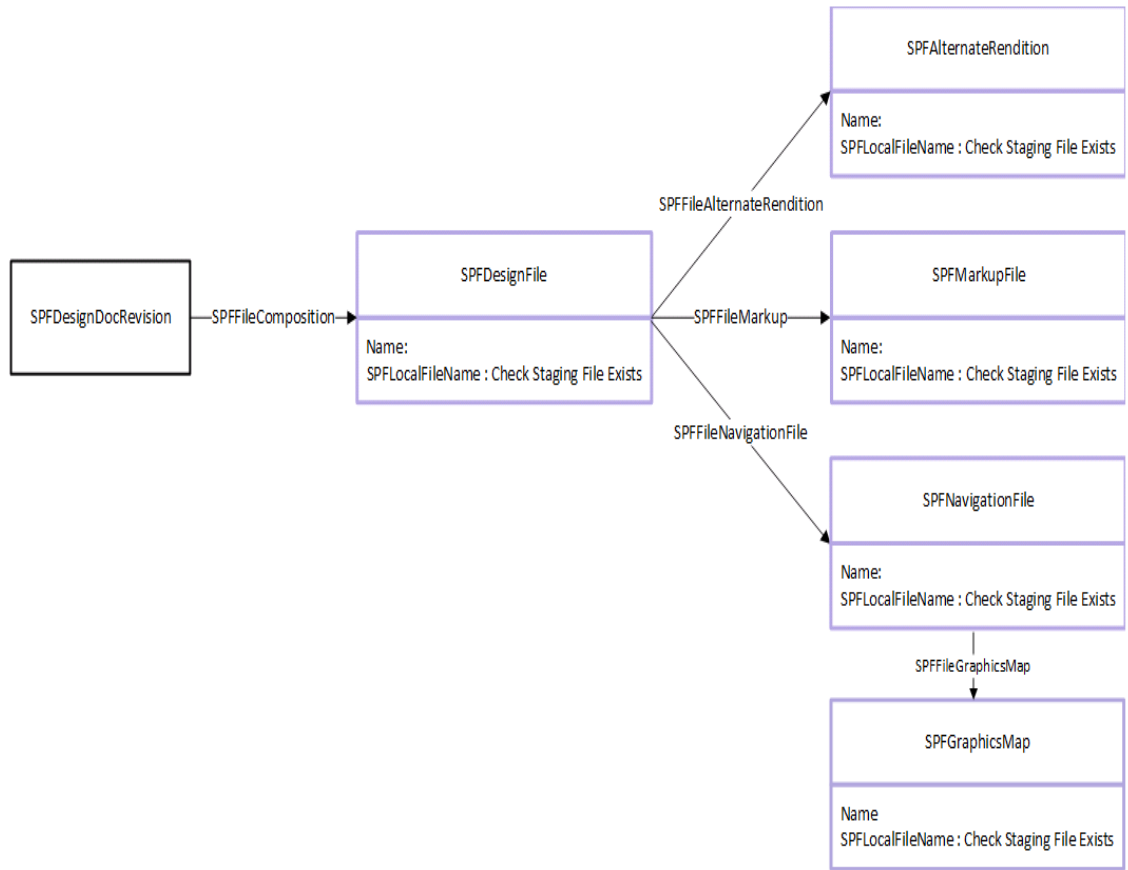
**Scenario:** File1 already exists in the database. The **Allow Version Creation** option is set to True and **Copy Files from the Previous Version** option is set to False. Input CSV file with information about File2 is loaded.

**Result:** A new version V2 is created and the File2 is attached to version V2.

## Mappings for supported files

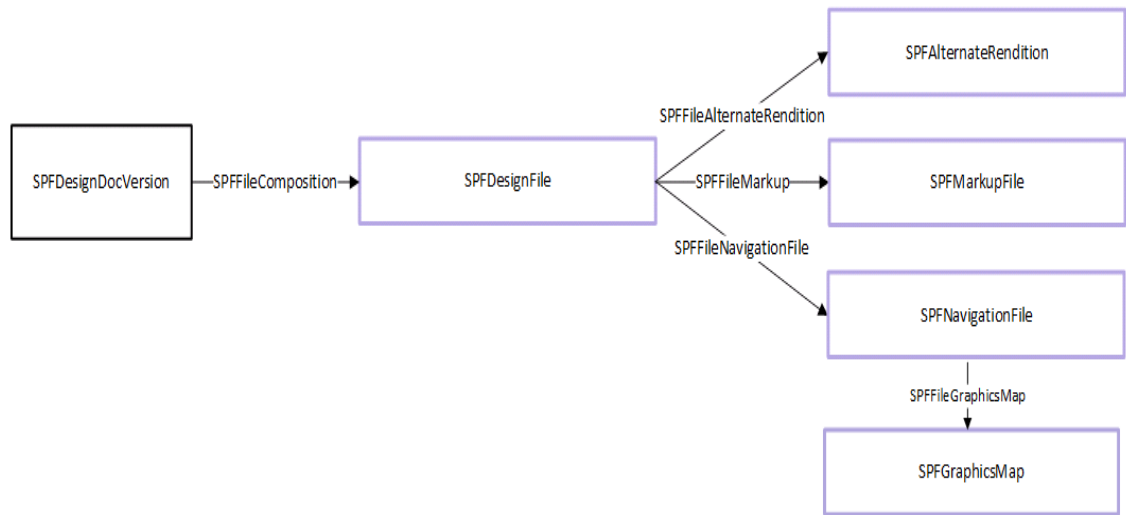
When you export files to a target system, the import mappings search for the design file, markup file, navigation file, graphics file, and alternate rendition files. The design file is then attached to

the revision object in the staging system and the other files are attached to the design file, except for the graphics map file, which is attached to the navigation file.



During the export process, local staging files are uploaded to the target system file services, and various design files are created and saved in a vault.

This mapping uses the **Check Target System File Exists** action, which ensures that the file is uploaded correctly using the path accessible by the target system server.



## CSV file example

DOCUMENT_NUM	REV	DOCUMENT	ISSUE_D	DOCUM	File 1	NavigationFile	MarkupFile	AlternateRendition	GraphicsMap	FileDirecto	REVISI
DOC-001	1A	Contract	#####	Docum	1F1.txt	navigation.igr	DWG_Mark	AlternateRendition	DWG_GM.xm	C:\Files\	Rev1A

## Object mappings example

Column Header	Column Type	Object Mappings	Object Name	Optional interface
File 1	Physical	SPFDesignFile	File 1	ISPFNavigationFileComposition
MarkupFile	Physical	SPFMarkupFile	MarkupFile	
AlternateRendition	Physical	SPFAlternateRendition	AlternateRendition	
NavigationFile	Physical	SPFNavigationFile	NavigationFile	ISPFGraphicsMapComposition
GraphicsMap	Physical	SPFGraphicsMap	GraphicsMap	



## Property mappings example

Column Header	Column Type	Property Mappings	Parent Objects
File 1	Physical	Name	File 1
MarkupFile	Physical	Name	MarkupFile
AlternateRendition	Physical	Name	AlternateRendition
NavigationFile	Physical	Name	NavigationFile
GraphicsMap	Physical	Name	GraphicsMap
DesignFile Map	Computed (func.Concat( [FileDirectory], [File 1]))	SPFLocalFileName	File 1
MarkupFileName	Coputed (func.Concat ([FileDirectory],[M arkupFile]))	SPFLocalFileName	MarkupFile
AlternateRendition	Computed(func.Conca t ([FileDirectory], [AlternateRendition ]))	SPFLocalFileName	AlternateRendition
NavigationFileNam e	Computed(func.Conca t ([FileDirectory], [NavigationFile]))	SPFLocalFileName	NavigationFile
GraphicsMap	Computed(func.Conca t ([FileDirectory], [GraphicsMap]))	SPFLocalFileName	GraphicsFile

## Relationships mappings example

Relationship Mappings	End Column 1	End Column 2
SPFFileComposition	File 1	Document_NUMBER
SPFFileMarkup	File 1	MarkupFile
SPFAlternateRendition	File 1	AlternateRendition
SPFFileNavigationFile	File 1	NavigationFile
SPFGraphicsMap	NavigationFile	GraphicsFile

### NOTES

- To identify supporting file objects, use query definition as **.Name** for SPFDesignFile, SPFMarkupFile, SPFAlternateRendition, SPFNavigationFile, and SPFGraphicsMap.
- The design file must exist before the markup and navigation files in the Target system. The export weightings must be set to export the design file before the markup file and navigation file, and the navigation file before the graphics map. The navigation file must be exported before the graphics map file.

## APPENDIX H

# Smart Data Validator Mapping Export

You can export the mappings, validation rulesets, and job definitions created in one Smart Data Validator system to another system using the Smart Data Validator Mapping Export Utility. Mappings are exported as XML load files using SmartPlant Foundation Desktop Client, but the objects and relationships must be converted before the mappings can be loaded into another system.

The Smart Data Validator Mapping Export Utility converts the XML load files to NLF format and removes UUIDs from the files. This avoids clash problems in databases with existing mappings. You need to delete the column headers of an existing import map before loading the NLF file using the Loader in SmartPlant Foundation Desktop Client.

The Smart Data Validator Mapping Export Utility and accompanying documentation is delivered with Smart Data Validator at this default location: [*installation location*]\Smart\SDV\2015\Utilities\UnSupported\MappingExportUtility.

# Delete Job Definitions Using SmartPlant Foundation

You can delete a job definition from the Smart Data Validator system using SmartPlant Foundation Desktop Client. However, if any relationships exist between the job definition and jobs being run through a Smart Data Validator Job Management workflow, the option to delete is not available. You can also delete a job definition using Smart Data Validator Administration. For more information, see *Delete a job definition* (on page 70).

1. In Smart Data Validator Administration, clear the **Is Enabled** option in the job definition so it cannot be used to create any new jobs in Smart Data Validator Job Management. For more information, see *Edit a job definition* (on page 70).
2. In Smart Data Validator Job Management, delete any jobs that used the job definition to be deleted. For more information, see *Smart Data Validator Job Management User's Guide*.
3. Open SmartPlant Foundation Desktop Client on a computer which has Smart Data Validator fully installed.
4. Click **Find > SDV > SDV Job Definitions**.
5. Using the **Find SDV Job Definition** dialog box, enter the name of the job definition or a wildcard character, and click **OK**.
6. Right-click the job definition to be deleted, and click **Delete**.

## APPENDIX J

# Supported CSV File Formats

Smart Data Validator supports the data import from two types of CSV file formats, which allows the same validation rules to be used regardless of the imported data file and the exported data to be in the same invariant format as the target system.

- CSV file format.
- CSV raw attribute file format (inverted file format).

**NOTE** For more information on using computed columns and functions, see *Use the Computed column* (on page 71).

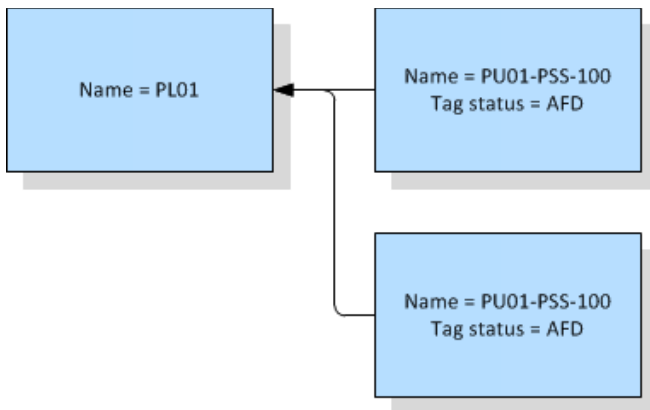
Each CSV file format is explained as follows:

### CSV file format

The CSV file format is a standard delimited text file that uses a comma, or other separators, to separate values for columns in a spreadsheet. For example:

PLANT_CODE	TAG_NAME	TAG_STATUS
PL01	PU01-PSS-100	AFD
PL01	PU01-PSS-101	AFD

This information is shown in the staging area as shown in the following illustration:

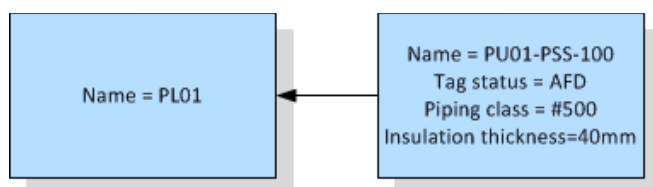


## CSV Raw Attribute file format (inverted file format)

CSV Raw Attribute file format is used to load design properties. This is when a single object has many properties that makes it impractical to represent each property in its own column. Each row in the CSV Raw Attribute file format refers to a different property definition, whereas the property definition is named in the column mapping for a standard CSV file. For example:

PLANT_CODE	TAG_NAME	PROPERTY_NAME	PROPERTY_VALUE	PROPERTY_VALUE_UOM
PL01	PU01-PSS-100	pipng class	#500	
PL01	PU01-PSS-100	insulation thickness	40	mm

This information is shown in the staging area as shown in the following illustration:



## CSV files not in invariant format

### What is invariant format?

Invariant format is a property that does not change when a specified transformation is applied on a CSV file. For example, Smart Data Validator uses the invariant format for thousand separator “,” and for decimal separator “.”. So a value of 1.001 would always be interpreted as 1 with three decimal places even if your intention may have been 1001, as the decimal point was really a thousand separator. Smart Data Validator will not return any validation errors on properties using the invariant format, but you run the risk of not getting the intended data into the system.

### Invariant format CSV files

If the data in a CSV file is not supplied in invariant format, it is possible that Smart Data Validator will interpret the data as valid data and not return any validation errors on the export of the data. Therefore, you run a risk of not getting the intended data into the target system. For example, submitting a value of 1.001 would be interpreted as 1 with three decimal places, when the intention may have been 1001, as the decimal point was really a thousand separator. Similar conditions can occur with dates.

If you import a CSV file that is not in invariant format, you must use computed columns to convert the data to invariant format. This applies to all dates, time offsets, and numerics in localized format.

**TIP** Smart Data Validator uses the invariant format as defined by SmartPlant Foundation as follows:

- Date Time Format: yyyy/MM/dd - HH:mm:ss:fff
- Timezone: GMT with no day light savings adjustment
- Year Month Day Format: yyyy/MM/dd
- Numeric format:
  - Thousand separator “,”

- Decimal separator "."

For more information on using computed columns and functions, see *Use the Computed column* (on page 71) and *Functions* (on page 89).

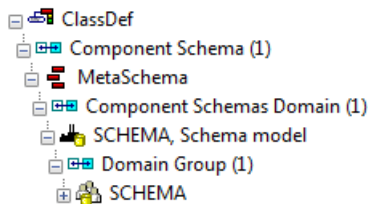
## APPENDIX K

# Clash Detection

Smart Data Validator has advanced clash detection capabilities, which ensure that clashes between the imported schema and the existing local schema are detected so that corrective action can be applied. Smart Data Validator works only with the data in the SDV domain, because it has control over those database tables and can transfer the data for each job into a new set of tables.

When you build an import mapping from the target system that maps any reserved class definitions, property definitions, relationship definitions, (such as an interface definition or a property definition), or relationship property definitions, the system automatically detects clashes and it displays a warning message to take corrective action, such as renaming the class definition.


For example, the following image is an example of a definition for **ClassDef**, which is a reserved type that is unusable for Smart Data Validator.



Here, **ClassDef** is linked to a component schema, which in turn is linked to a domain, which in turn links to a domain group. The domain group has a property that names the table prefix. For this example, the domain group is named SCHEMA.

This example does not point to the SDV tables, but points to the SCHEMA tables, which means that Smart Data Validator cannot use this class definition, as doing so would create a clash between what is being mapped and what is already in the Smart Data Validator database.

If you do intend to introduce a new class definition into your target system, use the following example workflow:


1. Create an import mapping by querying the target system.
2. Query for the object type, for example, **ClassDef**.
3. Click **Add**  to create the mapping, and the application will check for clashes.
4. If a class definition, property definition, or relationship property definition clashes, the application creates a list of alternatives by searching the stage to target map to determine whether the class definition, property definition, or relationship property definition has already been targeted. If it has, the local schema is presented as the alternative. For example, if a map exists from the local class definition **ClassDef** to a target class definition **ClassDef**, then the local class definition **ClassDef** will be suggested as an alternative.




5. If a clash is found, choose an alternative class definition from the list provided by the application. If no list of alternatives is provided, use free text to rename the object type, renaming **ClassDef** to **MyClassDef**.
6. Click **Save**.
7. When you make use of the **Auto Generate Export Mapping** option, the local object is named **MyClassDef**, but when **MyClassDef** is exported, it is exported as **ClassDef**.

Similar to clashes between class or property definitions, clashes can occur between relationship definitions, because the Smart Data Validator object types do not have the same interfaces as the objects in the target system. Relationship definitions link instances of interfaces, so when importing a relationship definition local to Smart Data Validator, the relationship definition is mapped to the primary interfaces of the two objects being linked. A clash occurs when the relationship definition is found locally, but links two different interfaces. Smart Data Validator looks at the end objects and matches to the objects you want to use that already exist in the Smart Data Validator database.

This condition can also occur for properties, which should only be exposed by one interface. If you target a property that exists in the Smart Data Validator database, but belongs to an interface not found on the associated object, a clash occurs. You will be prompted to alter the property name. Similar clash detection and resolution can occur in the case of relationship properties.

 **TIP** This clash detection functionality is also used when mapping using the free text mode.

 **NOTE** You can create new relationship definitions in addition to the existing relationship definition in the system between different interface definitions.

## APPENDIX L

# Export Objects configured with ENS in Target System

The Engineering Numbering System (ENS) provides automated object identification, which allows project-specific identification formats to be configured as templates. SmartPlant Foundation uses these templates to guide the user when creating new objects.

The ENS definition is composed of various components called ENS items that can extract data from constants and from other properties on the object or related objects. These components are configured in sequence with additional delimiter characters to define the required format.

If a classdef is configured to use ENS in the target system database, then the object name is set by the ENS definition instead of using the actual name of the object provided in the CSV file during export.

If the ENS definition is related to a property definition, such as Name, and this property definition is referred to in a UID or unique key, then Smart Data Validator fails to identify the object and leads to export failures. This is because the relationships are processed as the UIDs or unique keys and these get changed according to the ENS definition.

**⚠ CAUTION** It is not recommended to run jobs in parallel when using ENS because it might lead to data corruption and unexpected results.

### Configuration for tag object

In order to map a tag object available in the CSV file with the tag object in the target system database, you have to do the following:

- Create a new property definition to store the tag name from the CSV file.
- Create import mapping with a computed column and map it to the name property of the target system.

**📘 NOTE** The new property definition used to store the original name of the tag object from the CSV file helps Smart Data Validator identify the tag object within the target system during the update, delete, and terminate object actions.

For example, if ENS is configured on the Name property of a tag, such as `[Current Config][-][TG][-][Sequence]`, and TAG1 is exported to the target system where the configuration is PlantA and 'TG' is a constant, then the name of the tag in the target system is set to PlantA-TG-00001.

### Example CSV File

A	B	C	D	E	F
TAG NAME	DESCRIPTION	CLASS	AREA	SYSTEM	CONTRACT
TAG1	Safety Valve	DEV Electrical Equipment	AR-01	SYS-01	CTR-01

### Object Mapping Example

Column Header	Column Type	Object Mappings	Object Name
TAG NAME	Physical	DevTag	DevTag
CLASS	Physical	SPFPrimaryClassification	SDVPrimaryClassification
AREA	Physical	SPFFunctionalArea	SDVFunctionalArea
SYSTEM	Physical	SPFSystem	SDVSystem
CONTRACT	Physical	SPFContract	SDVContract

### Property Mapping Example

Column Header	Column Type	Property Mappings	Parent Objects
ActualTSTagName	Computed	Name (Create Update action)	TAG NAME
TAG NAME	Physical	New property definition added by user (Create Update action)	TAG NAME

### Computed Column for ActualTSTagName

```
func.GetTargetSystemValueIfEmptyReturnDefault("true", [TAGNAME],
"#DEVTag,.newpropertydef=[TAGNAME]", ".Name")
```

**NOTE** The unique identifier for the tag name column should be changed to contain the column header of the computed column. For example, *[Current Config],[Area],TAG NAME* becomes *[Current Config],[Area],ActualTSTagName*.

### Configuration for document object

If you have configured ENS for a document master, revision, and/or version, the property definition must be created on the master and revision. The document number must be set on the master. The document number, major revision, and minor revision must be set on the revision.

In order to map a document object available in the CSV file with the document object in the target system database, you have to do the following:

- Create two new property definitions - one for master and one for revision to store the document number and the revision code respectively.
- Create import mapping with computed columns and map them to the document master and document revision of the target system.

### Example CSV File

A	B	C	D	E
DOCUMENT_NUMBER	REVISION_CODE	ISSUE_DATE	DOCUMENT_TYPE	DOCUMENT_TITLE
DocENS2	1A	14-Jan-14	Civil Plan	Document 002 Civil Plan

### Object Mapping Example

Column Header	Column Type	Object Mappings	Property Mappings	Object Name
DOCUMENT_NUMBER	Physical	SPFDesignDocRevision		SDVDesignDocRevision

Column Header	Column Type	Object Mappings	Property Mappings	Object Name
REVISION_CODE	Physical		SPFExternalRevision	SDVExternalRevision
ISSUE_DATE	Physical			
DOCUMENT_TYPE	Physical	SPFDocumentClass		SPFDocumentClass
DOCUMENT_TITLE	Physical	SPFDesignDocMaster		SPFDesignDocMaster
REVISION_SCHEME	Constant	SPFRevisionScheme		SPFRevisionScheme
OWNING_GROUP	Constant	SPFDataAccessGroup		SPFDataAccessGroup
ISSUE_DATE_CONV	Computed		SPFRevIssueDate	SDVRevIssueDate
MAJORREVISION	Computed		SPFMajorRevision	SDVMajorRevision
MINORREVISION	Computed		SPFMinorRevision	SDVMinorRevision

### Property Mapping Example

Column Header	Column Type	Property Mappings	Parent Objects
MasterCustomProperty	Computed	New property definition to store the master (Create Update action)	DOCUMENT_TITLE
MasterActualTSName	Computed	Name (Create Update action)	DOCUMENT_TITLE
RevActualTS Name	Computed	New property definition to store the revision (Create Update action)	DOCUMENT_NUMBER
RevisionCustomProperty	Computed	Name (Create Update action)	DOCUMENT_NUMBER

### Computed Column for MasterCustomProperty

```
func.Concat({[DOCUMENT_NUMBER], ""})
```

### Computed Column for MasterActualTSName

```
func.GetTargetSystemValueIfEmptyReturnDefault("true",
[DOCUMENT_NUMBER],
"#SPFDesignDocMaster,.NewPropDefToStoreMaster=[DOCUMENT_NUMBER]",
".Name")
```

### Computed Column for RevActualTSName

```
func.Concat({[MasterCustomProperty], "_", [MAJOR_REVISION], "_",
[MINOR_REVISION]})
```

### Computed Column for RevisionCustomProperty

```
func.GetTargetSystemValueIfEmptyReturnDefault("true",
[MasterActualTSName],
```

```
"#SPFDesignDocRevision,.NewPropDefToStoreRevision=[RevActualTSName]  
", ".Name")
```

### NOTES

- The Unique Identifier for the document number column must be changed to contain the column header of the computed column. For example, change DOCUMENT\_NUMBER, [*Major Revision*], [*Minor Revision*] to RevisionCustomProperty, [*Major Revision*], [*Minor Revision*].
- The Unique Identifier for the document title column must be changed to contain the column header of the computed column. For example, change DOCUMENT\_NUMBER to MasterActualTSName.

# Case-Insensitive Support for Oracle Database

The case-insensitivity of the database interactions in Smart Data Validator are dependent on the value set for the **CaseInsensitiveOracle** property in the **Settings** node for a site in SmartPlant Foundation Server Manager. It is recommended to have the same value set for the **CaseInsensitiveOracle** property in the staging and target systems.

★ **IMPORTANT** In order to use case-insensitive support, you must configure the export mapping using query definitions to find objects in the target system.

## NOTES

- The value set for the **CaseInsensitiveOracle** property does not apply to the schema items, UoM values, and enum values.
- Keyword searches with the criteria specified using the unique key and UID are case-sensitive.
- The **Run Inconsistency Property Values** option in the **Create Job Definition** dialog box checks the value of the **CaseInsensitiveOracle** property of the staging system for the case-sensitivity of the rule execution.

## Validation rules

The following table lists all the validation rules supplied with Smart Data Validator and the system types that are considered for the case-sensitiveness of the rule execution.

Rule Name	System Type
Check Claimed to Sub Configuration	Target system
Check File Exists	Not applicable (case-sensitive)
Check For Cyclic Relationships	Staging system
Check Object Exists	Target system
Check File Exists for Document Revision	Not applicable (case-sensitive)
Check Parallel Claim	Target system
Check Is Claimable	Target system
Check Is Revisable	Target system

Rule Name	System Type
Check Revised to Sub Config	Target system
Check Revised to Parallel Config	Target system
Check Properties Against Schema	Not applicable (case-sensitive because it searches for schema items)
Check Relationship Exists	Target system
Check Unique Key	Not applicable (case-sensitive because the search criteria uses the unique key)
Compare All Property Values For Objects	Staging system and target system
Compare Value	Staging system and target system
Date Time	Not applicable
Double	Not applicable
Integer	Not applicable
Regular Expression	Not applicable (case-sensitive)
Staging System Cardinality	Not applicable (case-sensitive because it searches for schema items)
String Does Not Start With	Staging system
String Length	Not applicable
String Not Equal To	Staging system
Target System Cardinality	Not applicable (case-sensitive because it searches for schema items)
UOM	Target system

### Computed columns

The following table lists all the computed columns supplied with Smart Data Validator and the system types that are considered for the case-sensitiveness of the rule execution.

Function Name	System Type
AddDefaultUOMIfMissing	Not applicable (case-sensitive because the search criteria uses UoM values)
Concat	Not applicable (case-sensitive)
ConvertToHash	Not applicable
DateTimeColumn	Not applicable
Decode	Not applicable (case-sensitive)
Divide	Not applicable
GetDefaultUOMIfMissing	Not applicable (case-sensitive because the search criteria uses UoM values)
GetFileName	Not applicable
GetJobDetails	Not applicable
GetMajorRevisionCodeFromTargetSystem	Not applicable (case-sensitive because it searches for schema items)
GetMinorRevisionCodeFromTargetSystem	Not applicable (case-sensitive because it searches for schema items)
GetTargetSystemValueIfEmptyReturnDefault	Target system
GetValueFromTargetSystem	Target system
IndexOf	Not applicable (case-sensitive)
Join	Not applicable (case-sensitive)
Left	Not applicable
Length	Not applicable
Minus	Not applicable
Multiply	Not applicable
PadLeft	Not applicable (case-sensitive)



Function Name	System Type
PadRight	Not applicable (case-sensitive)
RegexMatch	Not applicable (case-sensitive)
RegexReplace	Not applicable (case-sensitive)
Replace	Not applicable (case-sensitive)
Right	Not applicable
Split	Not applicable (case-sensitive)
SplitAlphaNumericSequence	Not applicable (case-sensitive)
SubString	Not applicable
Sum	Not applicable
ToLower	Not applicable (case-sensitive)
ToUpper	Not applicable (case-sensitive)
Translate	Not applicable (case-sensitive)
Trim	Not applicable (case-sensitive)
TrimEnd	Not applicable (case-sensitive)
TrimStart	Not applicable (case-sensitive)
YMDColumn	Not applicable

### Prompted APIs

The following table lists all the prompted APIs supplied with Smart Data Validator and the system types that are considered for the case-sensitiveness of the rule execution.

Parameter Name	System Type
GetEnumEntries	Not applicable (case-sensitive because the search criteria uses enumerated values)
GetObjectPropertiesByType	Target system
GetValueFromTargetSystemObjects	Target system

Parameter Name	System Type
GetValuesFromTargetSystemRelExpan sion	Target system

### Implicit Delete Rules

The implicit delete rules use the value set in the **CaseInsensitiveOracle** property of the target system for the case-sensitivity of the rule execution.

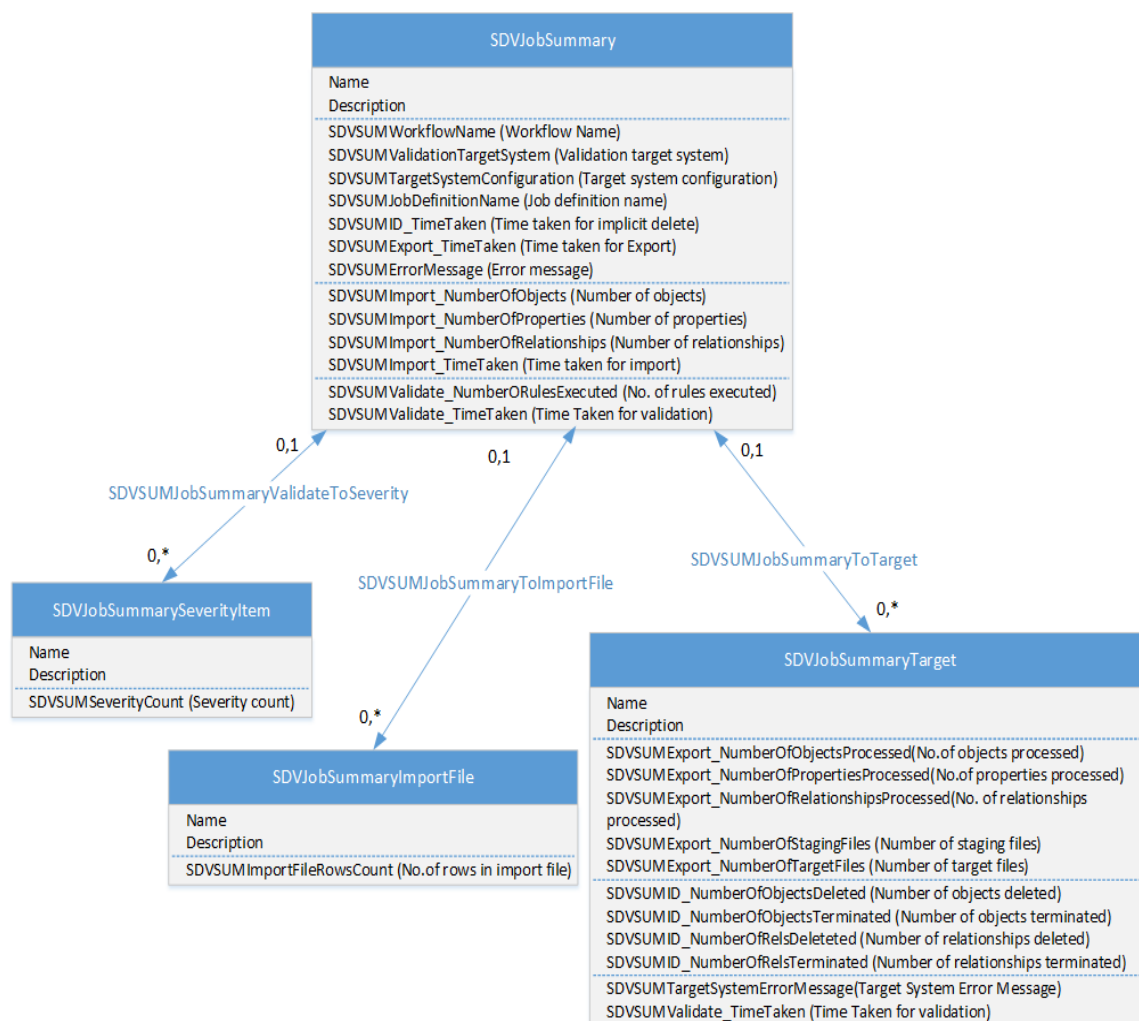
★ **IMPORTANT** For SQL Server databases, the case-sensitivity of Smart Data Validator is based on the collation settings. You must set the case-sensitive or case-insensitive settings while installing the database. This setting will determine whether or not Smart Data Validator will consider case during rule execution.

## APPENDIX N

# Job Reports

You can use Smart Data Validator to store the information related to a job in the database. The stored information can be used as a metadata for the generation of reports. For this purpose, a new schema is provided to record the job information, such as Import Details, Validation, Implicit Delete, and Export Details. If you delete a job from the target system, only that job is deleted, its details are still stored in the database. This happens only if you select the **Record Job Details** option when creating a job definition, and associate it with that specific job. You can find the job summary details in the staging system.

The following image shows the new class definitions and the corresponding property definitions that are modeled to hold job details.



The following class definitions are delivered with the software.

- **SDVJobSummary** - Stores the basic details of a job, such as Import Stage Details and Validation Stage Details.
- **SDVJobSummarySeverityItem** - Displays the severity items of a job, such as warnings, errors, and the count.
- **SDVJobSummaryTarget** - Stores export stage details and implicit delete stage details of each target system.
- **SDVJobSummaryImportFile** - Stores import file that is used to execute a Job.

Each class definition contains property definitions that allows you to define the properties of a job, such as name, description, and other properties. The **SDVJobSummarySeverityItem**, **SDVJobSummaryTarget**, and **SDVJobSummaryImportFile** class definitions are associated with the main class definition **SDVJobSummary**.

#### **NOTES**

- The job details are stored in the staging system even after deletion of the job from Smart Data Validator.
- To view job details, you must be working in a role associated with the VTLAdmin access group.

For example, create a job definition, named **Test** by applying **Record Job Details** option. For more information, refer to *Create a job definition* (on page 68). Use this job definition to create a new job named **Job1**.

The following job details are stored in the staging system. If you delete the job **Job1**, its job details will still be available in the staging system.

<b>Context</b>	
Query Configuration	Scope Not Set
<b>Object</b>	
Creation Date	21/04/2016 11:42:02
Creation User	superuser
Description	
Domain UID	ADMIN
Last updated date	21/04/2016 11:43:12
Name	210_LoadFolders
Object Configuration	
Termination Date	
Termination User	
<b>SDV Job Summary Detail</b>	
Job definition name	Test
Target system configuration	CF_ConfigurationTop
Time taken for Export	28 Second(s)
Time taken for implicit delete	16 Second(s)
Validation target system	SPFOOTB
Workflow name	Import Validate Delete Export
<b>SDV Job Summary Import</b>	
Number of objects	4
Number of properties	8
Number of relationships	0
Time taken for import	2 Second(s)
<b>SDV Job Summary Validate</b>	
Number of rules executed	3
Time Taken for validation	5 Second(s)

**NOTE** Select the **Record Job Details** option to store the job information such as Import details, Validation, Export details and other job information.

## APPENDIX O

# EPC Database Registration After Project Completion

When a project completes, an EPC may hand over the project database to the Owner Operator. The number of EPC databases that are handed over, as well as the tools that the databases are from determine whether the tool signature for the database can be re-registered to the Owner Operator's SmartPlant Foundation site.

- If only one EPC is handing over the database, then the tool signature can be re-registered to the Owner Operator's SmartPlant Foundation site.
- If more than one EPC is handing over databases for tools other than Smart 3D, then only one of the databases can be registered to the Owner Operator's SmartPlant Foundation site.
- If more than one EPC is handing over databases with a Smart 3D tool signature, all of the EPC databases can be registered to the Owner Operator's SmartPlant Foundation site.

**NOTE** Smart 3D is the only tool that supports multiple registrations to the same SmartPlant Foundation site.

# Glossary

## ***actions***

An indicator of what Smart Data Validator will do with the object data in the validation and export process, such as update and delete. These kinds of operations vary, depending on whether the column header is mapped to an object, a property, or a relationship.

## ***auto-generate***

An option to automatically generate validation rule definitions, export mappings, and rules, based on actions. The validation rules are used to validate imported and exported data and the export mappings ensure that the column headers match the objects and properties found in the target system.

## ***brownfield***

An existing project or area that has constraints imposed due to prior work and contains existing data.

## ***cardinality***

A setting on a relationship definition that specifies how many instances of a relationship are valid for the objects at the end of the relationship. For example, a tag cannot exist without a primary classification relationship, and also cannot have more than one primary classification.

## ***column headers***

They are used in Smart Data Validator as the basis for mapping new data in columns to match an existing structured database.

## ***configuration tree***

A representation in a tree list, which may include plant, areas, units, and projects, that indicates the structure in which the data is stored in SmartPlant Foundation.

## ***CSV file***

A comma-separated value (CSV) file, which stores tabular data in plain-text format.

## ***data files***

A job can process multiple input files and therefore use multiple input mappings. Import mappings are defined on the job definition.

## ***export mapping***

The process that maps the imported objects, classes, and properties to the existing structure of the objects, classes, and properties found in a target system.

## ***export process***

The Export process uses a defined mapping, based on the structure of the target system, to manage the loading of the data into the final destination system.

***function***

A computed functional code run at import, where the output value of a function depends only on the arguments that are input to the function.

***greenfield***

A project or area that is completely new and does not have any constraints imposed by prior work or existing data.

***implicit delete***

A component process used by Smart Data Validator, where a user can decide to implicitly delete or terminate a group of objects from the target system, because they are no longer in the supplied input data submission.

***import definition***

A defined mapping of imported file objects, properties, and relationships from existing column headers to objects, properties, and relationships found in the staging area database during the import process.

***import process***

The process that manages the import of data in the data file or files to the staging area database using a defined mapping.

***inverted CSV file***

See *raw attribute format* (on page 177).

***job***

A defined object that carries information for the progress of data through a selected workflow in Smart Data Validator Job Management, such as when data is imported, validated, and exported to a target system.

***job definition***

A combined set of components configured for the import, validation, and export of data to a specific target system or multiple target systems.

***mapping***

A defined process where data is correlated from existing column headers for the objects, properties, and relationships in the imported data to the correct column headers for the objects, properties, and relationships found in another database, such as the staging database or target system. For more information, see *import definition* (on page 176) and *export mapping* (on page 175).

***object weighting***

A process that emphasizes the contribution of an object in a set of data to a final effect or result, thereby deciding its weight in the analysis. This affects the order in which the data is exported, helping to achieve the desired result.



***query definition***

A comma separated list of properties that can be entered in the **Target System Query Definition** field, which identifies the object in the target system along with the class definition. These properties can be retrieved by navigating the relationships.

***raw attribute format***

Where the data in a CSV file is organized in a vertical format and each row contains properties and relationships for the same object. The data in a standard CSV file is typically organized horizontally. Also referred to as an inverted CSV file.

***relationship definition***

A defining object that relates items together in a database which are stored in different tables.

***relationship property***

A property on the link interface which is part of the relationship definition. The link interface allows properties to be created on the relationship itself.

***rules***

A logical formula used to evaluate and verify whether the data in an imported record meets the requirements and data standards specified for the target database system.

***rulesets***

A combined set of rules that can be run as a set during job processing.

***SDV***

Smart Data Validator, the software used for importing data into the staging area, validating the data, and/or exporting data to a target system.

***staging***

The staging or staging area or staging database used is the first part of a combined set of processes in Smart Data Validator, where imported data is held and validated before being exported to a Smart Data Validator site.

***target system***

The system or database used as the destination for exported data that has been validated before export. It is also used as the basis for creating import mappings, implicit delete, and validation rules.

***terminate***

The action of changing an object's status to terminated without removing it from the database. Terminating objects, instead of deleting them, allows you to continue to see the history of the object after termination.

***UID***

Unique Identifier is used to uniquely identify an object within a system. UID definitions must be set on all objects that Smart Data Validator exports from the staging area if the unique key or query definition is left blank. The staging area UID definitions must match the UID definitions in the target system.

***unique key***

A set of values guaranteed to be unique for each object in a relation, and can be used to identify objects in the target system.

***URL***

A Uniform resource locator is a web address that contains a specific character string that references a resource.

***validation process***

A process that evaluates the imported data against a defined set of rules to ensure the validity of the data.

***validation rule***

A logical formula used to evaluate the imported data in one or more fields to determine whether it matches the existing criteria and hierarchy found in the target system database. There are two types of validation rules:

1. Rules that determine if the data is valid for the target system schema (these rules can be autogenerated).
2. Rules that determine if the data meets specific business criteria, such as naming format.

# Index

## A

- actions • 180
- Actions • 119
- Attach the Convert Import Validate Export workflow automatically • 87
- auto-generate • 180

## B

- brownfield • 180

## C

- cardinality • 180
- Case-Insensitive Support for Oracle Database • 171
- Clash Detection • 165
- column headers • 180
- Column types • 34
- Computed column .NET methods • 109
- Computed column functions • 91
- configuration tree • 180
- Configure the automatic generation of validation rules • 59
- Connect to a target system • 36
- Converting a SmartPlant Foundation Object to a Smart Data Validator Job • 83
- ConvertToHash function • 108
- Copy a validation rule set • 58
- Copy an export mapping • 48
- Copy an import definition • 31
- Create a job definition • 69
- Create a new column header • 35
- Create a new job details item object • 85
- Create a new job details object • 86
- Create a new object or property using free text • 37
- Create a new target system • 26
- Create a new validation rule set • 57
- Create a stage object to target object mapping • 49
- Create a submittal • 87
- Create a validation rule • 58
- Create a validation rule relationship • 63
- Create an export mapping • 48
- Create an implicit delete rule • 66
- Create an import definition • 30
- Create column header mappings • 37

- Creating new versions and copying files from the previous version • 155
- CSV file • 180
- CSV file example • 143, 145, 157
- CSV files not in invariant format • 163
- CSV mapping example • 153

## D

- data files • 180
- Date and time functions • 108
- Define a raw attribute map • 42
- Define Column Headers • 33
- Define Export Mapping • 47
- Define Implicit Delete Rules • 65
- Define Import Mapping • 28
- Define Job Definitions • 68
- Define stage object to target object mappings • 49
- Define stage properties and relationships to target object mappings • 50
- Define Target Systems • 26
- Define Validation Rules and Rulesets • 56
- Delete a job definition • 71
- Delete Job Definitions Using SmartPlant Foundation • 161
- Document function examples • 107
- Document functions • 102
- Document Mappings • 135
- Documents with concurrent engineering • 139

## E

- Edit a job definition • 71
- Edit a raw attribute map • 44
- Edit a stage object to target object mapping • 50
- Edit a target system • 27
- Edit a validation rule • 59
- Edit a validation rule set • 57
- Edit an export mapping • 48
- Edit an implicit delete rule • 67
- Edit an import definition • 31
- Edit column headers • 41
- Environment variable examples • 133
- Environment Variables • 131
- EPC Database Registration After Project Completion • 179

Export File Mappings • 152  
export mapping • 180  
Export mapping actions • 121  
Export mappings example • 151  
Export Objects configured with ENS in  
  Target System • 167  
export process • 180  
Exporting multiple revisions • 140

## F

Files with custom target systems • 154  
function • 181  
Functions • 91

## G

Generate raw attribute property mappings  
  for export • 44  
greenfield • 181

## I

implicit delete • 181  
import definition • 181  
Import mapping examples • 138  
import process • 181  
inverted CSV file • 181

## J

job • 181  
job definition • 181  
Job Reports • 176

## L

Load files in SmartPlant Foundation  
  Desktop Client using Loader • 84

## M

Manage column header order • 42  
Manage validation rules • 59  
mapping • 181  
Mappings for published documents • 142  
Mappings for supported files • 156  
Monitor the submittal • 89

## N

Naming convention for rule auto-generation  
  • 63  
Non-target system computed columns • 106

## O

Object mapping • 37  
Object mappings example • 143, 145  
object weighting • 181

## P

Preface • 9  
Prompted API column functions • 102  
Property mapping • 39  
Property mappings example • 144, 147  
Published document mappings • 144

## Q

query definition • 182

## R

raw attribute format • 182  
Relationship cardinality rules • 123  
relationship definition • 182  
Relationship mapping • 39  
Relationship mappings example • 144, 149  
relationship property • 182  
Relationship property mapping • 40  
Rename action in Smart Data Validator •  
  123  
Reservation and activation of document  
  masters • 142  
Rule auto-generation model • 60  
rules • 182  
rulesets • 182

## S

SDV • 182  
Search using a local search • 36  
Set illegal characters for import mapping •  
  25  
Set Smart Data Validator Default Options •  
  23  
Smart Data Validator Mapping Export • 160  
Smart Data Validator process architecture •  
  16  
Smart Data Validator rule severity levels •  
  63  
Smart Data Validator work process • 20  
Stage property to target property mappings  
  • 51  
Stage relationship property to target  
  relationship property mappings • 54  
Stage relationship to target property  
  mapping • 52

- Stage relationship to target relationship
  - mappings • 53
- staging • 182
- Summary map • 40
- Supported CSV File Formats • 162

## T

- target system • 182
- Target system computed columns • 103
- Target system query definitions • 113
- Target system unique keys • 112
- terminate • 182
- Tool signature mappings • 143
- Trigger the job • 81
- Triggering a Job External to the Job Management Module • 77

## U

- UID • 182
- UID definitions mapping example • 149
- UID Definitions mappings example • 144
- Understand Implicit Delete Functionality • 126
- Understanding the control file format • 78
- unique key • 183
- Unique Keys and Query Definitions • 112
- URL • 183
- Use Function Column Type Mappings • 72
- Use planned revisions of documents with an MDR submittal • 89
- Use the Computed column • 72
- Use the Prompted API column • 76
- Using silent workflows and job scheduler • 78
- Using Smart Data Validator • 21

## V

- validation process • 183
- validation rule • 183
- Validation rule description • 116
- Validation Rules and Actions • 115
- View mapping summary • 32
- View the job • 89

## W

- Welcome to Smart Data Validator • 16
- What's New in Smart Data Validator Administration? • 11
- Workflow • 18