

IBM SPSS Modeler 14.2 Solution
Publisher



Note: Before using this information and the product it supports, read the general information under Notices on p. .

This edition applies to IBM SPSS Modeler 14 and to all subsequent releases and modifications until otherwise indicated in new editions.

Adobe product screenshot(s) reprinted with permission from Adobe Systems Incorporated.

Microsoft product screenshot(s) reprinted with permission from Microsoft Corporation.

Licensed Materials - Property of IBM

© Copyright IBM Corporation 1994, 2011.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Preface

IBM® SPSS® Modeler is the IBM Corp. enterprise-strength data mining workbench. SPSS Modeler helps organizations to improve customer and citizen relationships through an in-depth understanding of data. Organizations use the insight gained from SPSS Modeler to retain profitable customers, identify cross-selling opportunities, attract new customers, detect fraud, reduce risk, and improve government service delivery.

SPSS Modeler's visual interface invites users to apply their specific business expertise, which leads to more powerful predictive models and shortens time-to-solution. SPSS Modeler offers many modeling techniques, such as prediction, classification, segmentation, and association detection algorithms. Once models are created, IBM® SPSS® Modeler Solution Publisher enables their delivery enterprise-wide to decision makers or to a database.

About IBM Business Analytics

IBM Business Analytics software delivers complete, consistent and accurate information that decision-makers trust to improve business performance. A comprehensive portfolio of [business intelligence](#), [predictive analytics](#), [financial performance and strategy management](#), and [analytic applications](#) provides clear, immediate and actionable insights into current performance and the ability to predict future outcomes. Combined with rich industry solutions, proven practices and professional services, organizations of every size can drive the highest productivity, confidently automate decisions and deliver better results.

As part of this portfolio, IBM SPSS Predictive Analytics software helps organizations predict future events and proactively act upon that insight to drive better business outcomes. Commercial, government and academic customers worldwide rely on IBM SPSS technology as a competitive advantage in attracting, retaining and growing customers, while reducing fraud and mitigating risk. By incorporating IBM SPSS software into their daily operations, organizations become predictive enterprises – able to direct and automate decisions to meet business goals and achieve measurable competitive advantage. For further information or to reach a representative visit <http://www.ibm.com/spss>.

Technical support

Technical support is available to maintenance customers. Customers may contact Technical Support for assistance in using IBM Corp. products or for installation help for one of the supported hardware environments. To reach Technical Support, see the IBM Corp. web site at <http://www.ibm.com/support>. Be prepared to identify yourself, your organization, and your support agreement when requesting assistance.

Contents

1	<i>About IBM SPSS Modeler</i>	1
	IBM SPSS Modeler Server	1
	IBM SPSS Modeler Options	1
	IBM SPSS Text Analytics	2
	IBM SPSS Modeler Documentation	2
	Application Examples	3
	Demos Folder	4
2	<i>IBM SPSS Modeler Solution Publisher</i>	5
	How IBM SPSS Modeler Solution Publisher Works	5
	Publishing Streams	6
3	<i>Executing Published Streams</i>	9
	Using the Parameter File to Customize Stream Execution	10
	Embedding IBM SPSS Modeler Solution Publisher into Applications	11
	IBM SPSS Modeler Solution Publisher Sample Application	12
4	<i>Installing IBM SPSS Modeler Solution Publisher Runtime</i>	13
	Installing the Runtime	13
	System Requirements	13
	Installing on Windows	14
	Installing on UNIX	14
	Starting the Runtime	16
	Changing the Temp Directory	17
	Uninstalling IBM SPSS Modeler Solution Publisher Runtime	17
	Windows Uninstallation Procedure	17
	UNIX Uninstallation Procedure	18

5 IBM SPSS Modeler Solution Publisher Runtime Library API Reference 19

Introduction 19
API Process Overview 19
API Functions 20

Appendix

A Notices 33

Index 36

About IBM SPSS Modeler

IBM® SPSS® Modeler is a set of data mining tools that enable you to quickly develop predictive models using business expertise and deploy them into business operations to improve decision making. Designed around the industry-standard CRISP-DM model, SPSS Modeler supports the entire data mining process, from data to better business results.

SPSS Modeler offers a variety of modeling methods taken from machine learning, artificial intelligence, and statistics. The methods available on the Modeling palette allow you to derive new information from your data and to develop predictive models. Each method has certain strengths and is best suited for particular types of problems.

SPSS Modeler can be purchased as a standalone product, or used in combination with SPSS Modeler Server. A number of additional options are also available, as summarized in the following sections. For more information, see <http://www.ibm.com/software/analytics/spss/products/modeler/>.

IBM SPSS Modeler Server

SPSS Modeler uses a client/server architecture to distribute requests for resource-intensive operations to powerful server software, resulting in faster performance on larger data sets. Additional products or updates beyond those listed here may also be available. For more information, see <http://www.ibm.com/software/analytics/spss/products/modeler/>.

SPSS Modeler. SPSS Modeler is a functionally complete version of the product that is installed and run on the user's desktop computer. It can be run in local mode as a standalone product or in distributed mode along with IBM® SPSS® Modeler Server for improved performance on large data sets.

SPSS Modeler Server. SPSS Modeler Server runs continually in distributed analysis mode together with one or more IBM® SPSS® Modeler installations, providing superior performance on large data sets because memory-intensive operations can be done on the server without downloading data to the client computer. SPSS Modeler Server also provides support for SQL optimization and in-database modeling capabilities, delivering further benefits in performance and automation. At least one SPSS Modeler installation must be present to run an analysis.

IBM SPSS Modeler Options

The following components and features can be separately purchased and licensed for use with SPSS Modeler. Note that additional products or updates may also become available. For more information, see <http://www.ibm.com/software/analytics/spss/products/modeler/>.

- SPSS Modeler Server access, providing improved scalability and performance on large data sets, as well as support for SQL optimization and in-database modeling capabilities.

- SPSS Modeler Solution Publisher, for real-time or automated scoring outside the SPSS Modeler environment. For more information, see the topic [IBM SPSS Modeler Solution Publisher in Chapter 2 on p. 5](#).
- Adapters to enable deployment to IBM SPSS Collaboration and Deployment Services or the thin-client application IBM SPSS Modeler Advantage. For more information, see the topic [Storing and Deploying IBM SPSS Collaboration and Deployment Services Repository Objects in Chapter 9 in *IBM SPSS Modeler 14.2 User's Guide*](#).

IBM SPSS Text Analytics

IBM® SPSS® Text Analytics is a fully integrated add-on for SPSS Modeler that uses advanced linguistic technologies and Natural Language Processing (NLP) to rapidly process a large variety of unstructured text data, extract and organize the key concepts, and group these concepts into categories. Extracted concepts and categories can be combined with existing structured data, such as demographics, and applied to modeling using the full suite of IBM® SPSS® Modeler data mining tools to yield better and more focused decisions.

- The Text Mining node offers concept and category modeling, as well as an interactive workbench where you can perform advanced exploration of text links and clusters, create your own categories, and refine the linguistic resource templates.
- A number of import formats are supported, including blogs and other web-based sources.
- Custom templates, libraries, and dictionaries for specific domains, such as CRM and genomics, are also included.

Note: A separate license is required to access this component. For more information, see <http://www.ibm.com/software/analytics/spss/products/modeler/>.

IBM SPSS Modeler Documentation

Complete documentation in online help format is available from the Help menu of SPSS Modeler. This includes documentation for SPSS Modeler, SPSS Modeler Server, and SPSS Modeler Solution Publisher, as well as the Applications Guide and other supporting materials.

Complete documentation for each product in PDF format is available under the *\Documentation* folder on each product DVD.

- **IBM SPSS Modeler User's Guide.** General introduction to using SPSS Modeler, including how to build data streams, handle missing values, build CLEM expressions, work with projects and reports, and package streams for deployment to IBM SPSS Collaboration and Deployment Services, Predictive Applications, or IBM SPSS Modeler Advantage.
- **IBM SPSS Modeler Source, Process, and Output Nodes.** Descriptions of all the nodes used to read, process, and output data in different formats. Effectively this means all nodes other than modeling nodes.
- **IBM SPSS Modeler Modeling Nodes.** Descriptions of all the nodes used to create data mining models. IBM® SPSS® Modeler offers a variety of modeling methods taken from machine learning, artificial intelligence, and statistics. For more information, see the topic [Overview of Modeling Nodes in Chapter 3 in *IBM SPSS Modeler 14.2 Modeling Nodes*](#).

- **IBM SPSS Modeler Algorithms Guide.** Descriptions of the mathematical foundations of the modeling methods used in SPSS Modeler.
- **IBM SPSS Modeler Applications Guide.** The examples in this guide provide brief, targeted introductions to specific modeling methods and techniques. An online version of this guide is also available from the Help menu. [For more information, see the topic Application Examples in IBM SPSS Modeler 14.2 User's Guide.](#)
- **IBM SPSS Modeler Scripting and Automation.** Information on automating the system through scripting, including the properties that can be used to manipulate nodes and streams.
- **IBM SPSS Modeler Deployment Guide.** Information on running SPSS Modeler streams and scenarios as steps in processing jobs under IBM® SPSS® Collaboration and Deployment Services Deployment Manager.
- **IBM SPSS Modeler CLEF Developer's Guide.** CLEF provides the ability to integrate third-party programs such as data processing routines or modeling algorithms as nodes in SPSS Modeler.
- **IBM SPSS Modeler In-Database Mining Guide.** Information on how to use the power of your database to improve performance and extend the range of analytical capabilities through third-party algorithms.
- **IBM SPSS Modeler Server and Performance Guide.** Information on how to configure and administer IBM® SPSS® Modeler Server.
- **IBM SPSS Modeler Administration Console User Guide.** Information on installing and using the console user interface for monitoring and configuring SPSS Modeler Server. The console is implemented as a plug-in to the Deployment Manager application.
- **IBM SPSS Modeler Solution Publisher Guide.** SPSS Modeler Solution Publisher is an add-on component that enables organizations to publish streams for use outside of the standard SPSS Modeler environment.
- **IBM SPSS Modeler CRISP-DM Guide.** Step-by-step guide to using the CRISP-DM methodology for data mining with SPSS Modeler.

Application Examples

While the data mining tools in SPSS Modeler can help solve a wide variety of business and organizational problems, the application examples provide brief, targeted introductions to specific modeling methods and techniques. The data sets used here are much smaller than the enormous data stores managed by some data miners, but the concepts and methods involved should be scalable to real-world applications.

You can access the examples by clicking Application Examples on the Help menu in SPSS Modeler. The data files and sample streams are installed in the *Demos* folder under the product installation directory. [For more information, see the topic Demos Folder in IBM SPSS Modeler 14.2 User's Guide.](#)

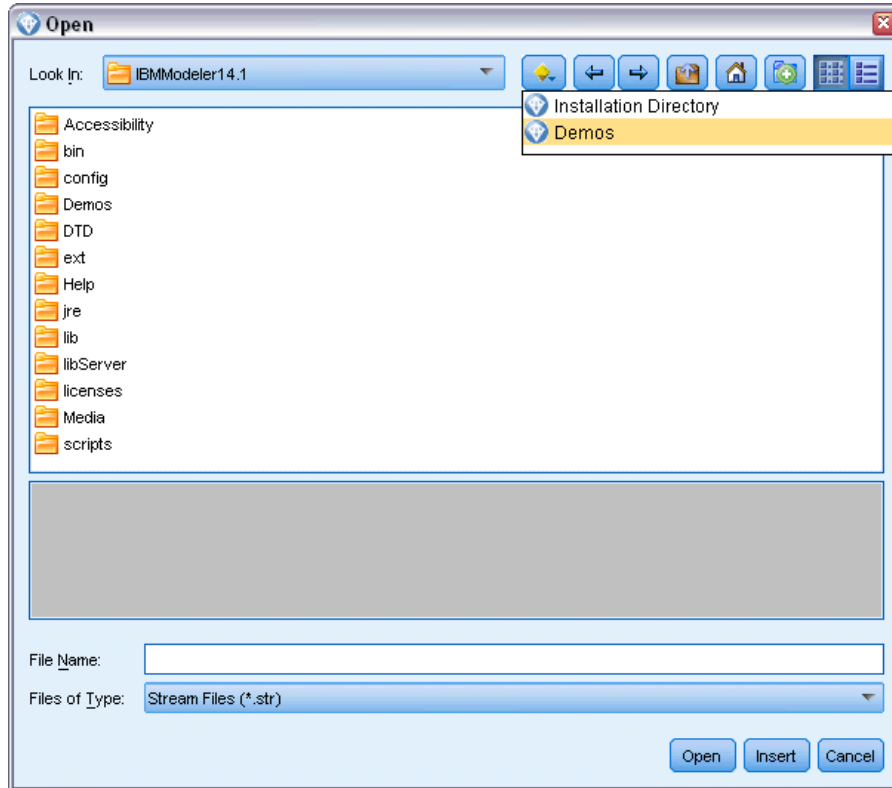
Database modeling examples. See the examples in the *IBM SPSS Modeler In-Database Mining Guide*.

Scripting examples. See the examples in the *IBM SPSS Modeler Scripting and Automation Guide*.

Demos Folder

The data files and sample streams used with the application examples are installed in the *Demos* folder under the product installation directory. This folder can also be accessed from the IBM SPSS Modeler 14.2 program group on the Windows Start menu, or by clicking *Demos* on the list of recent directories in the File Open dialog box.

Figure 1-1
Selecting the Demos folder from the list of recently-used directories



IBM SPSS Modeler Solution Publisher

IBM® SPSS® Modeler Solution Publisher is a powerful tool for integrating your data mining results into your business process to solve real-world problems. Using SPSS Modeler Solution Publisher, you can create a “packaged” version of a stream that can be executed by an external Runtime engine or embedded in an external application. This allows you to deploy your data modeling streams in a production environment to support your everyday business processes and to empower your organization’s decision makers with the knowledge gained from mining your data.

Using SPSS Modeler Solution Publisher offers more power than simply exporting the model (as PMML), because it allows you to publish and deploy *complete IBM® SPSS® Modeler streams*. That means you can perform data preparation as well as record and field operations, such as aggregating data, selecting records, or deriving new fields, before creating predictions based on a model. You can then further process the model results before saving the data—all simply by executing the published stream.

Note: Publishing is achieved by using the Publish tab of the appropriate export node. [For more information, see the topic Publishing Streams on p. 6.](#) Streams saved in releases prior to version 11.0 of the product used a separate Publisher node. The node is still supported for backwards compatibility, but is no longer available on the node palette.

Licensing IBM SPSS Modeler Solution Publisher

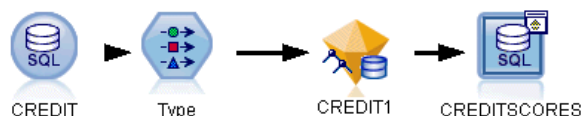
Note: SPSS Modeler Solution Publisher is distributed as part of the IBM® SPSS® Collaboration and Deployment Services Scoring Service, for which a separate license is required. <http://www.ibm.com/software/analytics/spss/products/deployment/cds/>

When you purchase a license, a separate IBM® SPSS® Modeler Solution Publisher Runtime is provided that allows you to execute published streams. [For more information, see the topic Installing the Runtime in Chapter 4 on p. 13.](#)

How IBM SPSS Modeler Solution Publisher Works

Deploying a solution using IBM® SPSS® Modeler Solution Publisher involves two phases: *publishing* a stream and *executing* a stream.

Figure 2-1
Publishing a stream



Publishing. As you work through the data mining process, you will eventually arrive at a model that provides a good solution to your business problem. At that point, you are ready to take that model and apply it to your business process. When you publish a stream, a detailed description of

the stream is written to the disk (as an *image* file and a *parameter* file). [For more information, see the topic Publishing Streams on p. 6.](#)

Executing. After you've published the stream, you can re-create the process implemented in the stream by executing the published stream. This is done either by using the standalone IBM® SPSS® Modeler Runtime (*modelerrun.exe*) or by developing an application that uses the SPSS Modeler Runtime Library to execute the stream. To execute streams outside of SPSS Modeler (using the Runtime or a custom application), you must first install the IBM® SPSS® Modeler Solution Publisher Runtime. [For more information, see the topic Installing the Runtime in Chapter 4 on p. 13.](#)

Publishing Streams

Publishing streams is done directly from IBM® SPSS® Modeler using any of the standard export nodes: Database, Flat File, Statistics Export, IBM® SPSS® Data Collection Export, SAS Export, Excel, and XML Export nodes. The type of export node determines the format of the results to be written each time the published stream is executed using the IBM® SPSS® Modeler Solution Publisher Runtime or external application. For example, if you want to write your results to a database each time the published stream is run, use a Database export node. [For more information, see the topic Overview of Export Nodes in Chapter 7 in IBM SPSS Modeler 14.2 Source, Process, and Output Nodes.](#)

To Publish a Stream

- ▶ Open or build a stream in the normal manner and attach an export node at the end.
- ▶ On the Publish tab in the export node, specify a rootname for the published files (that is, the filename to which the various extensions *.pim*, *.par* and *.xml* will be appended)..
- ▶ Click the Publish button to publish the stream, or select Publish the stream to automatically publish the stream each time the node is executed.

Figure 2-2
Publish tab



Published name. Specify the rootname for the published image and parameter files.

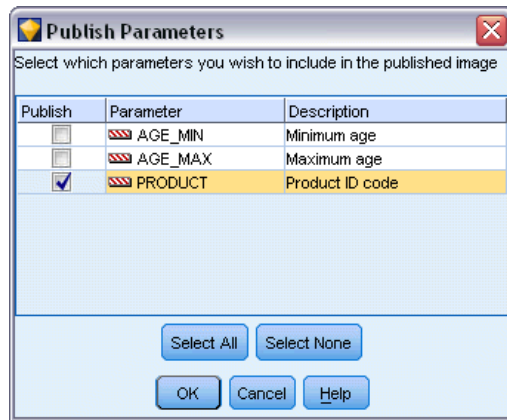
- The **image file** (*.pim) provides all of the information needed for the Runtime to execute the published stream exactly as it was at the time of export. If you are confident that you will not need to change any of the settings for the stream (such as the input data source or the output data file), you can deploy the image file only.
- The **parameter file** (*.par) contains configurable information about data sources, output files, and execution options. If you want to be able to control the input or output of the stream without republishing the stream, you will need the parameter file as well as the image file.
- The **metadata file** (*.xml) describes the inputs and outputs of the image and their data models. It is designed for use by applications which embed the runtime library and which need to know the structure of the input and output data.

Note: This file is only produced if you select the Publish metadata box.

Publish parameters. If required, you can include stream parameters in the *.par file. You can change these stream parameter values when you execute the image either by editing the *.par file or through the runtime API.

This check box enables the Parameters button. The Publish Parameters dialog box is displayed when you click the button.

Figure 2-3
Parameter selection



Choose the parameters you want to include in the published image by selecting the relevant check box in the Publish column.

On stream execution. Specifies whether the stream is automatically published when the node is executed.

- **Export data.** Executes the export node in the standard manner, without publishing the stream. (Basically the node executes in SPSS Modeler the same way it would if IBM® SPSS® Modeler Solution Publisher were not available.) If you select this option, the stream will not be published unless you do so explicitly by clicking the Publish button in the export node dialog box. Alternatively, you can publish the current stream using the Publish tool on the toolbar or by using a script.
- **Publish the stream.** Publishes the stream for deployment using SPSS Modeler Solution Publisher. Select this option if you want to automatically publish the stream every time it is executed.

Comments

- If you plan to run the published stream with new or updated data, it is important to note that the order of fields in the input file must be the same as the order of fields in the source node input file specified in the published stream.
- When publishing to external applications, consider filtering extraneous fields or renaming fields to conform with input requirements. Both can be accomplished using a Filter node prior to the export node. [For more information, see the topic Filtering or Renaming Fields in Chapter 4 in IBM SPSS Modeler 14.2 Source, Process, and Output Nodes.](#)

Executing Published Streams

The main effect of executing a published stream is always to generate a set of data. The data may be stored to a disk file or written to a database. Published streams cannot generate graphs, tables, generated models, or other non-data output.

Streams published using IBM® SPSS® Modeler Solution Publisher can be executed using the IBM® SPSS® Modeler Solution Publisher Runtime program. The Runtime program, *modelerrun.exe*, is started from the command line, with options indicating the image file and (optional) parameter file to execute. The Runtime command is as follows:

```
modelerrun -nobanner -p <parameter-file> -o <options> <image-file>
```

where

-nobanner	suppresses the startup text message for the Runtime (optional).
<parameter-file>	is the filename of the parameter file (optional).
<options>	is a comma-separated list of option=value pairs specifying execution options for the Runtime. Options are detailed below (optional).
<image-file>	is the filename of the image file to execute (required).

The following execution options are available:

Table 3-1
Valid Runtime options

Name	Default Value	Description
max_file_size	-1	Maximum size of saved files. A value of -1 indicates no limit.
max_sql_string_length	2,048	Maximum length of a string imported from the database with SQL. String values longer than this are truncated on the right without warning. The valid range is between 1 and 65,535 characters.
memory_usage	100	Multiplier for dynamic memory allocation. Adjust this value up or down to regulate the server's total memory usage.
temp_directory	"	Specifies the directory to be used for temporary files.
request_passwords	Y	Controls whether you are prompted for a database password when running a published stream that requires database access (Y/N).

Using the Parameter File to Customize Stream Execution

When a stream is published, the image file contains information about the data source and output data destination that were selected in the original stream. However, it is often convenient to run a stream against data from a different data source or to reroute the output data to a different destination. This can be done by editing the stream's parameter file.

This is particularly important when you will be executing the published stream on a computer different from the one on which it was created. You will need to update the locations of the input and output files to reflect the structure of the target computer. Note that published streams do not support relative paths, making it especially important to verify the data locations in your parameter file. (If you specify relative paths in the Export node options when publishing the stream, IBM® SPSS® Modeler will convert them to absolute paths before creating the image and parameter files for the published stream.)

You can also use the parameter file to set execution options and to change the values of stream parameters.

Parameter values are specified in the parameter file as <name.attribute>=<value> pairs. For some parameters, the parameter name contains an ID digit to distinguish nodes of the same type in the stream. For example, in a stream where both input and output refer to a database connection, the parameters describing the input connection might be dbconn0.datasource, dbconn0.user, etc., and the parameters associated with the output connection might be dbconn1.datasource, dbconn1.user, etc. Values are always enclosed in double quotes. The parameter file contains the following parameters, all of which can be changed to customize stream execution:

Table 3-2
Execution parameters

Parameter	Purpose
dbconnx.datasource	Data source name (DSN).
dbconnx.user	User name for restricted access databases.
dbconnx.password	Password for restricted access databases.
dbconnx.epassword	Encoded password for restricted access databases. To generate an encoded password, select Encode Password from the Tools menu of the SPSS Modeler user interface. Copy and paste the encoded password as needed.
dbconnx.password_required	Flag indicating whether to prompt for a password for this database connection. (The value is "Y" or "N".) Automatically set to "N" when there is no password specified at publish time.
dbconnx.catalog	Catalog associated with the database connection.
dbtablex.name	Base table name. <i>Note:</i> The order of fields in a new table must be the same as the order of fields in the originally published stream.
dbtablex.schema	Schema name.
dbtablex.catalog	Catalog associated with the current table.
filex.name	Filename for data file. <i>Note:</i> The order of fields in a new input file must be the same as the order of fields in the originally published stream.
filex.path	File location (directory name).
filex.field_names_included	For text format files, the flag indicating whether the field names are included as the first line of the file. (The value is "Y" or "N".)
filex.field_separator	For text format files, the character(s) used to separate field values.

Parameter	Purpose
filex.decimal_separator	For text format files, the character used for the decimal point.
options.angle_in_radians	Flag indicating whether radians are used as the unit of measurement in trigonometric CLEM expressions. (The value is “Y” or “N”.)
options.date_2digit_baseline	Defines the century for dates specified with two-digit years.
options.date_baseline	The baseline year (always January 1) used by CLEM date functions that work with a single date.
options.time_rollover	Flag indicating whether negative time differences refer to the past.
options.decimal_separator	The default decimal point character in text files.
options.time_format	Time format used when strings are interpreted as times by CLEM time functions.
options.date_format	Date format used when strings are interpreted as dates by CLEM date functions.
options.timestamp_format	Format used when reading timestamp fields as strings from ODBC data sources.
paramx.value	Stream parameter value. <i>Note:</i> The parameter name, description, and storage type are included for reference but cannot be changed. The value must be valid for the type.

Embedding IBM SPSS Modeler Solution Publisher into Applications

In addition to the standalone Runtime engine, IBM® SPSS® Modeler Solution Publisher provides a Runtime programming library (CLEMRTL) that allows other programs to control SPSS Modeler Solution Publisher stream execution. You can call CLEMRTL procedures in client programs written in C and C++. To use the CLEMRTL, you need to include the header file *clmrtl.h* (available from the *..\installation\clmrtl\include* folder) and be sure to link the appropriate library file for your development platform into your application when you build it. The *clmrtl.dll* file is available from the *..\installation\bin* folder.

Any source file that references library procedures must include the header *clmrtl.h*. This header file provides ANSI C prototypes for the library procedures and defines useful macros. It does not require any other headers to be included beyond what your program requires. To protect against name clashes, all library type and function names start with *clmrtl_* and all macro names are prefixed with *CLEMRTL_*.

CLEMRTL produces reports containing useful information that should be communicated back to the application in some way. The CLEMRTL provides three mechanisms for dealing with such messages:

- Messages can be written to a log file specified with the *clmrtl_setLogFile()* function.
- The application can retrieve details of the last error using the *clmrtl_getErrorDetail()* function.
- The application can provide its own report-handling procedure using the *clmrtl_setReportHandler()* function.

The Runtime system has a localized message catalog for reports. Any report passed to the application would include the localized message string. Reports also include the report code so that the application could choose to interpret and present the message differently. In order to get

detailed (localized) messages, the application needs to ship the appropriate *messages.cfg* file in a *config* directory. Different versions of *messages.cfg* for different locales can be found in the directory *<installdir>/config/<locale>*, where *<installdir>* is the directory in which you installed SPSS Modeler Solution Publisher, and *<locale>* is the desired locale. Select the appropriate version of *messages.cfg* and add it to a *config* subdirectory in your project directory.

The Runtime system needs no special environment or registry settings to operate correctly. Dependent libraries have to be distributed with an application linked to the CLEMRTL. These files are included in the *<installdir>/bin* directory.

Requirements

Following are the platforms and compilers tested with the Runtime programming library:

Table 3-3
Platforms and compilers

Platform	Suggested compiler(s)
AIX	IBM XL C/C++ Enterprise Edition V8.0 for AIX
HP-UX 11.0	aC++/ANSI C B3910B A.06.05
Solaris 64-bit	Sun Studio 11 5.8 Patch 121017-06
Linux 32-bit and 64-bit	gcc 3.2.3, Intel C++ compiler 9.1.043
Windows	Visual Studio 2005

Note: On HP-UX, the application must be built with the aCC compiler flag “-AA” to enable linking with the CLEMRTL libraries.

IBM SPSS Modeler Solution Publisher Sample Application

An example application, *dlltest.c*, is included in the *\clemtrl\demo* folder in the IBM® SPSS® Modeler Solution Publisher installation directory. Use the example application to help you get started embedding the Runtime library into your own applications.

The information used by the example application is included in the *\clemtrl\demo* folder; see the *README.txt* file for instructions. Compiling an application varies by platform, so example makefiles are also included.

Installing IBM SPSS Modeler Solution Publisher Runtime

Installing the Runtime

When you purchase IBM® SPSS® Modeler Solution Publisher as an add-on component for IBM® SPSS® Modeler, you will receive a separate installation disk. Use this disk to install the IBM® SPSS® Modeler Solution Publisher Runtime, which allows you to execute published streams. Published streams are independent of the operating system, so you can execute any published stream on any platform supported by the Runtime.

Note: The functionality that allows you to publish streams from SPSS Modeler is installed with IBM® SPSS® Modeler and can be enabled by purchasing a separate authorization code. Note that Published files must be re-exported with each release of SPSS Modeler. For example a file exported using an earlier release will not work with version 13 of SPSS Modeler Solution Publisher Runtime.

System Requirements

The system requirements for installing the IBM® SPSS® Modeler Solution Publisher Runtime are:

- **Operating system.** Windows Server 2008 R2 for 64-bit x64 systems; Windows Server 2003 Standard Edition R2 for 32-bit x86 or 64-bit x64 systems
IBM AIX® 6.1 or 7.1 for 64-bit POWER systems; HP-UX 11i V3 for 64-bit Itanium systems; Oracle Solaris™ 9.x or 10 for 64-bit SPARC systems; Red Hat Enterprise Linux 5.x for 32-bit x86 systems; Red Hat Enterprise Linux 5.x or 6.x for 64-bit x64 or IBM System z systems; Red Hat Enterprise Linux Advanced Platform 5.x for 32-bit x86 systems; Red Hat Enterprise Linux Advanced Platform 5.x or 6.x for 64-bit x64 or IBM System z systems; SuSE Linux Enterprise Server 10 or 11 for 64-bit x64 or IBM System z systems
- **Hardware.** Pentium® or Pentium-class processor or higher (for 32-bit Windows); x64 (AMD 64 and EM64T) processor family (for 64-bit Windows), running at 1GHz or faster
PowerPC processor, 233MHz or faster and IBM System p for IBM AIX; Itanium processor for HP-UX; UltraSPARC II (or better) for Solaris; Pentium or Pentium-class processor or higher for 32-bit Linux; x64 (AMD 64 and EM64T) processor family or IBM System z for 64-bit Linux
- **Minimum free disk space.** 1 gigabyte (GB) is recommended. 2 GB recommended if running on HP-UX.
- **RAM.** 4 GB is recommended.
- **Virtual environment.** The following virtual environments support IBM® SPSS® Modeler Server.
 - IBM PowerVM Hypervisor for IBM AIX
 - IBM PR/SM or z/VM for Linux for IBM System z

- VMWare ESX Server 4.1
- VMWare vSphere 4.0
- **Cluster environment.** The following cluster environments support SPSS Modeler Server.
 - IBM HACMP/RSCT for IBM AIX
 - IBM WebSphere App Server 64-bit
 - Microsoft Cluster Service for Windows
 - Oracle Solaris Cluster
 - Oracle WebLogic App Server 64-bit
 - Red Hat Cluster Suite for Red Hat Enterprise Linux

Installing on Windows

The following section describes the procedure for installing the SPSS Modeler Solution Publisher Runtime on the Windows operating system.

- ▶ Insert the IBM SPSS Modeler Solution Publisher installation disk into the drive.
- ▶ On the installation menu, choose Install IBM® SPSS® Modeler Solution Publisher Runtime or Install IBM® SPSS® Modeler Solution Publisher Runtime for Win64, depending on the system to which you are installing.
- ▶ On the installation wizard, click Next to begin.
- ▶ Follow the instructions that appear on the screen. To continue, click Next.
- ▶ Once you have specified all options, you are ready to install. Click Install to begin transferring files.
- ▶ After all files have been installed, click Finish.

Installing on UNIX

The following sections describe additional system requirements and the procedure for installing the IBM® SPSS® Modeler Solution Publisher Runtime on the UNIX operating system.

Additional Requirements for UNIX

You must ensure that kernel limits on the system are sufficient for the operation of IBM® SPSS® Modeler Server. The data, memory, and file ulimits are particularly important and should be set to unlimited within the SPSS Modeler Server environment. To do this:

- ▶ Add the following commands to *modelersrv.sh*:
 - `ulimit -d unlimited`
 - `ulimit -m unlimited`
 - `ulimit -f unlimited`

- ▶ Restart SPSS Modeler Server.

Solaris.

In addition to aforementioned system requirements, installing on Solaris also requires the following version specific patches. Patches can be downloaded from <http://sunsolve.sun.com/pub-cgi/show.pl?target=patchpage>.

- *Solaris 2.9 Patch IDs* : 111711-16, 111712-16, and 111722-05 (C++)
- *Solaris 2.10 Patch IDs*: 119963 (C++)

You must also have the library `/usr/lib/iconv/646%UTF-8` installed on your system. If this is not currently installed, please install the *SUNWiniu8* or *SUNWiniu8x* packages, which will install this library as part of the Indic iconv modules for UTF-8.

Linux 32-bit.

The following additional packages are needed to run properly. You can download these packages from the Red Hat Network. You may need to register to access the network (<https://rhn.redhat.com/network/software/search.pxt>). Use the search feature to search and drill down to each specific package version.

- The *libstdc++* package (version: *libstdc++-3.2.3-X.Y.i386* where *X.Y* is 22.1 or later).
- The *libgcc* package (version: *libgcc-3.2.3-X.Y.i386* where *X.Y* is 22.1 or later).
- The *compat-libstdc++* package (version: *compat-libstdc++-33-3.2.3-X.Y.i386* where *X.Y* is 47.3 or later).

Linux 64-bit.

The following additional packages are needed to run properly. You can download these packages from the Red Hat Network. You may need to register to access the network (<https://rhn.redhat.com/network/software/search.pxt>). Use the search feature to search and drill down to each specific package version.

- The *libstdc++* package (version: *libstdc++-3.2.3-X.Y.x86_64* where *X.Y* is 22.1 or later).
- The *libgcc* package (version: *libgcc-3.2.3-X.Y.x86_64* where *X.Y* is 22.1 or later).
- The *compat-libstdc++* package (version: *compat-libstdc++-33-3.2.3-X.Y.x86_64* where *X.Y* is 47.3 or later).

AIX.

In addition to aforementioned system requirements, installing on AIX also requires the following:

- XL C++ runtime should be at least at level 8.0.0.9 (lspp -l xlC.rte). The filesets that contain the libraries is called IBM C++ Runtime Environment Components for AIX and can be downloaded from <http://www.ibm.com/software/awdtools/xlcpp/support/>.
- XL FORTRAN runtime should be at least at level 10.1.0.4 (lspp -l xlfrte). The filesets that contain the libraries is called XL Fortran Enterprise Edition V10.1 for AIX Runtime Environment and can be downloaded from <http://www.ibm.com/software/awdtools/fortran/xlfortran/support/>.

HP-UX.

In addition to aforementioned system requirements, installing on HP-UX also requires that C++ Runtime libraries be installed. You can download them as patch PHSS_26946, as shown in the following table. To access the patches, go to <http://www4.itrc.hp.com/service/patch/mainPage.do>. Registration is required but is free of charge.

UNIX Installation Procedure

These installation instructions apply to the IBM® SPSS® Modeler Solution Publisher Runtime for UNIX. You need read and write permissions to the target installation directory, so log on with an account that has sufficient permissions.

- ▶ Insert the installation disk into the drive and ensure that you can access the drive. On Solaris, the volume manager (the vold daemon) must be running.
- ▶ Change directories to the DVD mount point.
- ▶ Change directories to the *modelrun* directory.
- ▶ Change to the relevant platform directory.
- ▶ Run the *.bin* install script. For example:

```
./modelersolutionpublisherhpia64.bin -i console
```
- ▶ Introduction details are displayed. Press Enter to continue.
- ▶ Licensing information is displayed. Read the license, type 1 to accept it, and press Enter to continue.
- ▶ A list of available languages is displayed. Enter the number for the language version that you want to install and press Enter to continue.
- ▶ You are prompted to type the installation location. To use the default directory, */usr/IBM/SPSS/ModelerSolutionPublisher<nn>* (where *<nn>* is the version number), press Enter. The Runtime will be installed in the specified directory.
- ▶ You are prompted to confirm the installation location. When it is correct, type *y* and press Enter.
- ▶ A pre-installation summary is displayed to confirm your entries so far. Press Enter to continue.
- ▶ A message is displayed to say the installation routine is ready to run. Press Enter to continue.
- ▶ A progress bar is displayed whilst the installation routine runs. When the installation is complete, press Enter to exit from the installer.

Starting the Runtime

Once you have installed the Runtime, you can use it to execute streams that have been published from IBM® SPSS® Modeler using any of the export nodes. [For more information, see the topic Publishing Streams in Chapter 2 on p. 6.](#) Start the Runtime from the command line, with options indicating the name of the published image file and an optional parameter file to execute.

To execute the Runtime, type the following line at the command prompt:

```
modelerrun -p <parameter-file> -o <options> <image-file>
```

where

<parameter-file>	is the filename of the published parameter file (optional).
<options>	is a comma-separated list of option=value pairs specifying execution options for the Runtime.
<image file>	is the filename of the published image file (*.pim) to execute.

Note: When using multibyte languages such as Chinese or Japanese in a UNIX environment, you must specify the language codes to be used. The following example shows the language codes for Simplified Chinese:

```
./modelerrun -o locale="zh_CN.GB18030",encoding="GB18030" -p
```

where -p is the location of the pim and par files.

For more information, see the topic [Executing Published Streams in Chapter 3 on p. 9](#).

Changing the Temp Directory

Some operations performed by the IBM® SPSS® Modeler Solution Publisher Runtime may require temporary files to be created. By default, the Runtime use the system temporary directory to create temp files. You can alter the location of the temporary directory using the following steps.

Note: It is very important to use the correct syntax, being especially careful of spaces and the directory path.

Suppose you are using the SPSS Modeler Solution Publisher Runtime with the following published files—*webtest.pim* and *webtest.par*. In this situation, you want modify the location of temporary files to use the directory *C:\published\temp*. To alter the location of the temp directory used by the executable:

- From the directory where the *.pim* and *.par* files are saved, run the command:

```
modelerrun -o temp_directory="C:\\published\\temp" -p webtest.par webtest.pim
```

This will direct the Runtime to use “*C:\published\temp*” as the temp directory for that specific process only.

Uninstalling IBM SPSS Modeler Solution Publisher Runtime

The following section describes the procedures for uninstalling the IBM® SPSS® Modeler Solution Publisher Runtime.

Windows Uninstallation Procedure

- In the Windows Control Panel, open Add or Remove Programs.
- From the list, select IBM® SPSS® Modeler Solution Publisher <nn>.

- ▶ Click Remove.
- ▶ Follow the instructions that appear on the screen to remove the Runtime.

Note: You must be logged on to your computer with administrator privileges to remove programs.

UNIX Uninstallation Procedure

- ▶ Remove the installation directory, including all of the installed files.

IBM SPSS Modeler Solution Publisher Runtime Library API Reference

Introduction

The CLEMRTL API allows you to control the execution of published streams from your application. This section describes the available API functions.

The following are general issues in using the API:

- The API has C-linkage for maximum compatibility, but the library has C++ dependencies. On some platforms, this might mean that it can be used only with a C++-aware linker.
- `clemrtl_initialise_ext()` must be called before any other functions in the library are used.
- The type `clemrtl_image_handle` is used as an image identifier.
- Every function returns a status indicator code. The values of the status indicator are:

CLEMRTL_OK	Success
CLEMRTL_FAIL	Failed with no further details available
CLEMRTL_ERROR	Failed with additional information about the error available through the <code>clemrtl_getErrorDetail()</code> function

API Process Overview

The general outline of an application using the API is as follows:

- ▶ Initialize the library using `clemrtl_initialise_ext()`.
- ▶ Check and change execution options using `clemrtl_getOption()` and `clemrtl_setOption()`.
- ▶ Open an image using `clemrtl_openImage()` and receive an image handle.
- ▶ Check and change image parameters using `clemrtl_enumerateParameters()`, `clemrtl_getParameter()`, and `clemrtl_setParameter()`.
- ▶ Execute the image using `clemrtl_execute()`. If the same image is to be executed multiple times without changing parameters, use `clemrtl_prepare()` first.
- ▶ Close the image using `clemrtl_closeImage()`.
- ▶ To cancel an execution in progress, use `clemrtl_interrupt()`.
- ▶ To retrieve information about the last error, use `clemrtl_getErrorDetail()`. To receive error messages and other diagnostic messages as they arrive, use `clemrtl_setReportHandler()`.

The library is designed for multithreaded use under the following conditions:

- Global functions, those not taking an image handle as an argument, cannot be called concurrently.
- Image-local functions, those taking an image handle as an argument, cannot be called concurrently when applied to the same image handle, with the exception of `clemrtl_interrupt()`.
- Image-local functions can be called concurrently when applied to different image handles.

In particular, this means that separate threads can prepare and execute images concurrently, provided that they are using different image handles. Different image handles may refer to the same image (`clemrtl_openImage()` creates a new handle each time it is called, even for the same image), but be sure when executing multiple instances of the same image to change the image parameters to redirect output as required. If two images executing concurrently both try to write to the same output file or database table, the results will be unpredictable.

API Functions

Following is a complete list of functions exposed in the API.

initialise

This function is deprecated and is equivalent to

```
initialise_ext(flags, 0, 0);
```

New programs should call *initialise_ext*.

initialise_ext

```
int clemrtl_initialise_ext(
    unsigned flags,
    int arg_count,
    const clemrtl_init_arg* args);
```

Initializes the IBM® SPSS® Modeler Solution Publisher Runtime. *This function must be called before any other API functions.*

Parameter	Description
flags	Controls some aspects of the initialization process. The value is constructed as a bitwise OR of the flags described below.
arg_count	The number of additional initialization arguments. This must be less than or equal to the length of the args array.
args	Additional initialization arguments. Arguments are described by the type: <pre>typedef struct _clemrtl_init_arg { const char* name; const char* value; } clemrtl_init_arg;</pre>

The name field is the name of a configuration property and the value field is its value. The following configuration properties are recognized:

Name	Value
installation_directory	The full path to the IBM® SPSS® Modeler Solution Publisher installation folder. The installation folder is that containing the bin and config folders.

Returns one of the following error codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed to initialize, with no further details available.

The flags parameter provides control over certain aspects of the initialization process. A value of 0 (or CLEMRTL_INIT_DEFAULTS) specifies the default behavior as follows:

- Set the default time zone by calling tzset().
- Set the LC_NUMERIC locale to “C”.
- Set the global new handler to throw an exception when memory is exhausted.

This behavior can be modified by specifying combinations of the following flags:

Flag	Description
CLEMRTL_INIT_NO_TZ	Don't set the time zone.
CLEMRTL_INIT_NO_LOCALE	Don't change the locale.
CLEMRTL_INIT_NO_NEW_HANDLER	Don't set the new handler.
CLEMRTL_INIT_LOCAL_NEW_HANDLER	Localize the new handler within each API call.

The library requires that the LC_NUMERIC locale should be set to “C” in order to properly convert between numbers and strings (the IBM® SPSS® Modeler number format uses the rules of the “C” locale). This includes the conversion of numbers read from and written to text files.

- Specify NO_LOCALE if your application depends on the LC_NUMERIC locale and does not require conversion between numbers and strings.
- NO_NEW_HANDLER and LOCAL_NEW_HANDLER are mutually exclusive. The library requires that new should throw an exception when memory is exhausted.
- Specify NO_NEW_HANDLER if your application sets its own new handler or if you are certain that your compiler/Runtime system throws an exception by default.

Specify LOCAL_NEW_HANDLER only if your application relies on new returning a null pointer when memory is exhausted. *Note:* This is not safe for use in a multithreaded environment.

getOption

```
int clemrtl_getOption(
    const char* name,
    char* value,
    int value_size);
```

Retrieves the value of an execution option.

Parameter	Description
name	The option name.
value	A buffer to receive the option value.
value_size	The size of the value buffer.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the option name is not one of the execution option names listed above—for example, `memory_usage`—or if the value buffer is not large enough to receive the value including a terminating null character. Option values are always strings, even when the interpretation is numeric. [For more information, see the topic Executing Published Streams in Chapter 3 on p. 9.](#)

setOption

```
int clemrtl_setOption(
    const char* name,
    const char* value);
```

Sets the value of an execution option.

Parameter	Description
name	The option name.
value	The option value.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the option name is not one of the execution option names listed above—for example, `memory_usage`. Option values are always strings, even when the interpretation is numeric.

setLogFile

```
int clemrtl_setLogFile(
    int mode,
    const char* log_file);
```

Redirects log messages from all subsequently opened images.

Parameter	Description
mode	The logging mode must be one of the following values: CLEMRTL_NULL_LOG—suppress messages. CLEMRTL_STDERROR_LOG—send messages to the standard error output. CLEMRTL_FILE_LOG—send messages to the file named by the log_file parameter.
log_file	String containing the filename.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the mode is not one of the recognized modes or if file logging is requested and the filename is not specified or is invalid.

openImage

```
int clemrtl_openImage(
    const char* image_file,
    const char* param_file,
    clemrtl_image_handle* handle);
```

Opens a published stream image.

Parameter	Description
image_file	The image filename.
param_file	The parameter filename.
handle	Receives the image handle.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The parameter file can be NULL if a separate parameter file is not required. The operation fails if the image file is not specified, if either filename is invalid, or if the content is unreadable. If the operation succeeds, the returned image handle can be used to identify the image instance in subsequent API calls.

closeImage

```
int clemrtl_closeImage(
    clemrtl_image_handle handle);
```

Closes an image handle and frees its resources.

Parameter	Description
handle	The image handle.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if another API call is in progress on the image, particularly if the image is still executing. If the operation succeeds, the image is closed and the handle cannot be used in any subsequent API call except another call of `clemrtl_closeImage()`, which has no effect.

enumerateParameters

```
int clemrtl_enumerateParameters(
    clemrtl_image_handle handle,
    clemrtl_parameter_proc proc,
    void* data);
```

Applies a callback procedure to each image parameter name and value.

Parameter	Description
handle	The image handle.
proc	The parameter callback procedure.
data	User-specified data for the callback.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

The callback procedure has the following type:

```
typedef void (*clemrtl_parameter_proc)(
    void* data,
    const char* name,
    const char* value);
```

Parameter	Description
data	User-specified data passed to <code>clemrtl_enumerateParameters()</code> .

Parameter	Description
name	The parameter name.
value	The parameter value.

The procedure is applied exactly once to each image parameter in an arbitrary order.

getParameter

```
int clemrtl_getParameter(
    clemrtl_image_handle handle,
    const char* name,
    char* value,
    int value_size);
```

Retrieves the value of an image parameter.

Parameter	Description
handle	The image handle.
name	The parameter name.
value	A buffer to receive the parameter value.
value_size	Size of the value buffer.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the parameter name does not match the name of any parameter in the image or if the value buffer is not large enough to receive the parameter value including a terminating null character. Parameter names use the `name.attribute` format described above—for example, `file0.name`—and parameter values are always strings, even when the interpretation is numeric.

setParameter

```
int clemrtl_setParameter(
    clemrtl_image_handle handle,
    const char* name,
    const char* value);
```

Sets the value of an image parameter.

Parameter	Description
handle	The image handle.
name	The parameter name.
value	The parameter value.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the image handle is invalid or if the parameter name does not match the name of any parameter in the image. Parameter names use the `name.attribute` format described above—for example, `file0.name`—and parameter values are always strings, even when the interpretation is numeric.

getFieldCount

```
int clemrtl_getFieldCount(
    clemrtl_image_handle handle,
    const char* key,
    size_t* field_count);
```

Returns the number of fields in an input source or an output target.

Parameter	Description
handle	The image handle.
key	The name of the input or output to be examined, as used in the parameters file. The key may refer to a file or a database.
field_count	Receives the number of fields.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

getFieldTypes

```
int clemrtl_getFieldTypes(
    clemrtl_image_handle handle,
    const char* key,
    size_t field_count,
    int* field_types);
```

Returns the field types for an input source or an output target.

Parameter	Description
handle	The image handle.
key	The name of the input or output to be examined, as used in the parameters file. The key may refer to a file or a database.

Parameter	Description
field_count	The number of fields to examine. This must be less than or equal to the length of the field_types array.
field_types	An array of length at least field_count which receives the field types. The types of the first field_count fields in the input or output are copied into the array. If field_count is greater than the actual number of fields the extra elements in the array are left undefined. Values for the field types are listed in the Data Types table below.

Data Types

The field_types must be one of the following:

Type	Interpretation	Typical 'C' Declarator
STRING	UTF-8 null-terminated character string.	const char*
INTEGER	32-bit signed integer.	int
LONG	64-bit signed integer.	long long
REAL	64-bit floating point.	double
TIME	64-bit signed integer (seconds since midnight).	long long
DATE	64-bit signed integer (seconds since midnight 01/01/1970).	long long
TIMESTAMP	64-bit signed integer (seconds since midnight 01/01/1970).	long long

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

setAlternativeInput

```
int clemrtl_setAlternativeInput(
    clemrtl_image_handle handle,
    const char* key,
    size_t field_count,
    int field_types,
    void** (*iterator)(void* arg),
    void* arg);
```

Replaces a file input source with an alternative input source.

Parameter	Description
handle	The image handle.

Parameter	Description
key	The name of the original input source as used in the parameters file. The input source must be a file so the key will always have the form “fileN” for some integer $N \geq 0$.
field_count	The number of fields in the input. The value must match exactly the number of fields in the original file input or the call will fail.
field_types	An array of types of length at least field_count. The value field_types[i] is the type of the i'th field and must be one of the values listed in the Data Types table below. The types must be compatible with those of the original file input or the call will fail.
iterator	A function which produces the alternative input data. The function is applied to its argument as follows: <pre>void** row = iterator(arg);</pre> <p>The function is called during execution (inside a call to <code>clemrtl_execute</code>) and is called once for each input record. A return value of NULL indicates the end of input in which case the function is not called again and execution will eventually terminate. Otherwise, the result is an array of data of length at least field_count where row[i] provides the value of the i'th field. A value may be NULL, otherwise it must be a pointer to a datum whose type is determined by the corresponding field_types[i]. Pointers must remain valid until the next call of the iterator, or until the end of execution if execution terminates prematurely.</p>
arg	An opaque argument which is passed to the iterator on each call.

Data Types

The field_types must be one from a restricted list of types. [For more information, see the topic getFieldTypes on p. 26.](#)

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the field types are not compatible with those of the original file input, or if the field count does not exactly match the number of fields in the original file input.

setAlternativeOutput

```
int clemrtl_setAlternativeOutput(
    clemrtl_image_handle handle,
    const char* key,
    size_t field_count,
    int field_types,
```

```
void (*iterator)(void* arg, void** row),
void* arg);
```

Replaces a file output target with an alternative output target.

Parameter	Description
handle	The image handle.
key	The name of the original output target as used in the parameters file. The output target must be a file so the key will always have the form " <i>fileN</i> " for some integer $N \geq 0$.
field_count	The number of fields in the output. The value must match exactly the number of fields in the original file output or the call will fail.
field_types	An array of types of length at least field_count. The value field_types[i] is the type of the i'th field and must be one of the values listed in the Data Types table below. The types must be compatible with those of the original file output or the call will fail.
iterator	A function which consumes the image output. The function is applied to its argument and to a data row as follows: iterator(arg, row); The function is called during execution (inside a call to clemrtl_execute) and is called once for each result row produced by the image. A row value of NULL indicates the end of output after which the function is not called again; an application should not rely on this final call and should flush and close any external resources, etc. when execution has terminated. Otherwise, the row is an array of data of length at least field_count where row[i] provides the value of the i'th result. A value may be NULL, otherwise it must be a pointer to a datum whose type is determined by the corresponding field_types[i]. The function must copy any data values it needs because the memory may not be preserved after the call has returned.
arg	An opaque argument which is passed to the iterator on each call.

Data Types

The field_types must be one from a restricted list of types. [For more information, see the topic getFieldTypes on p. 26.](#)

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_FAIL	Failed with no further details available.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the field types are not compatible with those of the original file output, or if the field count does not exactly match the number of fields in the original file output.

execute

```
int clemrtl_execute(clemrtl_image_handle handle);
```

Executes an image.

Parameter	Description
handle	The image handle.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

If the image has not been prepared, it is prepared first using the current parameter values. The operation fails if the image handle is invalid or if an error occurs during preparation or execution. The call does not return until execution is complete.

prepare

```
int clemrtl_prepare(clemrtl_image_handle handle);
```

Prepares an image for execution.

Parameter	Description
handle	The image handle.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

The operation fails if the image handle is invalid or if an error occurs during preparation.

An image must be prepared before it is executed. Preparing an image freezes parameter values into the image. `clemrtl_execute()` prepares an image automatically on each call if it hasn't been prepared already; using `clemrtl_prepare()` to prepare an image is useful if the image is to be executed multiple times with the same parameter values, and it can significantly improve performance. Once an image has been prepared, subsequent changes to parameter values are ignored; use `clemrtl_prepare()` again to update the image with the new parameter values.

interrupt

```
int clemrtl_interrupt(clemrtl_image_handle handle);
```

Terminates execution in progress on an image.

Parameter	Description
handle	The image handle.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

The operation has no effect if the image handle is invalid or if the image is not executing.

This function is safe to call concurrently with another API call on the same image handle.

getErrorDetail

```
int clemrtl_getErrorDetail(
    clemrtl_image_handle handle,
    char* severity,
    int* code,
    char* text,
    int text_size);
```

Retrieves detailed information about the last error that occurred on an image.

Parameter	Description
handle	The image handle.
severity	Receives the severity code as a single character: I—information W—warning E—error X—system error
code	Receives the error number.
text	A buffer to receive the message text.
text_size	Size of the text buffer.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

Results will be unreliable if the image handle is invalid. The message text is truncated if necessary to fit in the text buffer.

If an API call fails with CLEMRTL_ERROR and the error is unrelated to a particular image handle, passing 0 as the image handle will retrieve details of the last non-image-specific error but is unreliable in a multithreaded environment.

setReportHandler

```
int clemrtl_setReportHandler(
    clemrtl_image_handle handle,
    clemrtl_report_proc proc,
    void* data);
```

Installs a report callback procedure for an image.

Parameter	Description
handle	The image handle.
proc	The report callback procedure.
data	User-specified data for the callback.

Returns one of the following status codes:

Result	Description
CLEMRTL_OK	Success.
CLEMRTL_ERROR	Failed with further details available.

The callback procedure has the following type:

```
typedef void (*clemrtl_report_proc)(
    void* data,
    char severity,
    int code,
    const char* text);
```

Parameter	Description
data	User-specified data passed to <code>clemrtl_setReportHandler()</code> .
severity	The severity code as a single character: I—information W—warning E—error X—system error
code	The message number.
text	The message text.

The procedure is applied to each message as it arrives. Setting the procedure to NULL removes any existing handler installed on the image.

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 1623-14, Shimotsuruma, Yamato-shi Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group Attention: Licensing 233 S. Wacker Dr. Chicago, IL 60606 USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, and SPSS are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.html>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Other product and service names might be trademarks of IBM or other companies.

Index

- API
 - alternative input, 27
 - alternative output, 28
 - error codes, 19
 - execution options, 21–22
 - field count, 26
 - field types, 26
 - overview, 11, 19
 - reference, 19
 - setReportHandler, 31
 - status codes, 19
- API functions
 - closeImage, 19, 24
 - enumerateParameters, 19, 24
 - execute, 19, 30
 - getErrorDetail, 19, 31
 - getFieldCount, 19, 26
 - getFieldTypes, 19, 26
 - getOption, 19, 21
 - getParameter, 19, 25
 - initialising, 20
 - initializing, 19
 - interrupt, 19, 30
 - openImage, 19, 23
 - prepare, 19, 30
 - setAlternativeInput, 19, 27
 - setAlternativeOutput, 19, 28
 - setLogFile, 19, 22
 - setOption, 19, 22
 - setParameter, 19, 25
 - setReportHandler, 19
- application examples, 2

- closeImage
 - API function reference, 24
 - API process overview, 19
- closing
 - streams, 24

- deployment, 5
- documentation, 2

- enumerateParameters
 - API function reference, 24
 - API process overview, 19
- error codes, 19
- error reporting
 - API function reference, 22, 31
- examples
 - Applications Guide, 2
 - overview, 3
- execute
 - API function reference, 30
 - API process overview, 19

- executing streams
 - using IBM SPSS Modeler Solution Publisher, 9–10

- getErrorDetail
 - API function reference, 31
 - API process overview, 19
- getFieldCount
 - API function reference, 26
 - API process overview, 19
- getFieldTypes
 - API function reference, 26
 - API process overview, 19
- getOption
 - API function reference, 21
 - API process overview, 19
- getParameter
 - API function reference, 25
 - API process overview, 19

- IBM SPSS Modeler, 1
 - documentation, 2
- IBM SPSS Modeler Solution Publisher, 5–6
 - embedding in applications, 11
 - overview, 5
- IBM SPSS Modeler Solution Publisher Runtime
 - installing, 13
 - starting, 16
 - system requirements, 13–14
 - temp directory, 17
 - uninstalling, 17
- IBM SPSS Text Analytics, 2
- initialise
 - API function reference, 20
 - flags, 20
- initialise_ext
 - API function reference, 20
 - flags, 20
- initialize
 - API process overview, 19
- interrupt
 - API function reference, 30
 - API process overview, 19

- legal notices, 33
- locale
 - set using the API, 20
- log files, 22

- openImage
 - API function reference, 23
 - API process overview, 19
- opening
 - streams, 23

- output nodes, 5
- .par* file, 6
- parameters
 - API callback procedures, 24
 - API image parameters, 24–25
 - API parameter files, 23–24
 - parameters for stream execution, 10
- .pim* file, 6
- prepare
 - API function reference, 30
 - API process overview, 19
- publishing streams
 - IBM SPSS Modeler Solution Publisher, 5–6
- Runtime
 - options, 9
 - starting, 16
 - temp directory, 17
 - uninstalling, 17
 - UNIX installation, 14
 - Windows installation, 14
- Runtime programming library (CLEMRTL), 11
- setAlternativeInput
 - API function reference, 27
 - API process overview, 19
- setAlternativeOutput
 - API function reference, 28
 - API process overview, 19
- setLogFile
 - API function reference, 22
 - API process overview, 19
- setOption
 - API function reference, 22
 - API process overview, 19
- setParameter
 - API function reference, 25
 - API process overview, 19
- setReportHandler
 - API function reference, 31
 - API process overview, 19
- SPSS Modeler Server, 1
- status codes, 19
- temporary directory
 - IBM SPSS Modeler Solution Publisher Runtime, 17
- time zone
 - set using the API, 20
- trademarks, 34
- uninstalling
 - IBM SPSS Modeler Solution Publisher Runtime, 17
- UNIX
 - installing IBM SPSS Modeler Solution Publisher Runtime, 14
- Windows
 - installing IBM SPSS Modeler Solution Publisher Runtime, 13