DiType User Guide

DiType User Guide

RenderX

Telephone: 650-328-8000
Fax: 650-328-8008
Website: http://renderx.com
Email: support@renderx.com

Table of Contents

Preface

1. What's in this Document?

The RenderX DiType User Guide provides background information about what DiType does and explains how to use the product. The manual is divided into the following sections:

1. Overview
2. Installation
3. Standard Applications
4. Configuring DiType
5. Administration
6. Uninstalling
7. Integrating DiType

2. Prerequisites

DiType runs on a variety of platforms including Windows, Mac OS X, Linux, and FreeBSD.

In order to view PDF output, a viewer is required. Adobe provides a free one which can be downloaded and installed from the Adobe website.

To view PostScript files, one option is to use GhostView, which may be used for viewing PDF as well. Versions are available for most operating systems.

3. Technical Support

You can contact RenderX technical support by:

- Using the RenderX support portal at http://renderx.com/support/index.html
- Sending an email to support@renderx.com
- Calling 650-328-8000

Chapter 1. Overview

This section contains a brief introduction to the DiType application, including how to use it, and the environments in which it can be used.

1.1. Using DiType

RenderX XEP DiType is a versatile digital typesetting engine. The DiType engine accepts a number of file types as input, including XML source files and XSL-FO files. XML source files are first transformed into XSL-FO, and then rendered (typeset) to the selected output format. XSL-FO files are immediately rendered to the selected output format. Several output formats are supported, including PDF, PostScript and SVG.

DiType can be integrated into an organization's publishing environment in a number of ways. Here are a few examples of how organizations use DiType today:

Individual user publishes documents

This is the simplest usage scenario for DiType. An individual end user employs DiType to publish a document on an as-needed basis. Typically, the input file is a DocBook or DITA XML document that needs to be published to PDF.

Corporate documentation system automatically publishes documents fed into the system

This scenario is used by enterprises who integrate DiType into their documentation creation workflow, to allow all employees simple and easy access to DiType's typesetting capabilities. In this scenario, DiType is used to generate documentation on the fly, on demand.

Mass production of documents

Typically used by banks and phone companies in order to mass produce statements and bills for each customer on a regular basis.

Document creation in a small office/home office (SOHO)

An extension of the single user scenario, this option allows a number of users to use DiType to publish their documents.

The XEP DiType engine can be run from any of the following standard applications, which are included in the DiType installation package:

DiType Assistant

Enables easy transformation of files in a graphical environment, suitable for users who prefer an intuitive graphical user interface. For a detailed description, refer to Section 3.1, “DiType Assistant”.

DiType Command Line

For users who prefer the command line, DiType can be run from the command line. For further details, refer to Section 3.2, “DiType Command Line”.

ActiType

Enables automatic publishing of documents from a watched folder. ActiType processes input documents that are located in a specified folder (hot folder). This option is used to integrate DiType into a document publishing environment (such as a content management system). For a detailed description, refer to Section 3.3, “ActiType”.

1.1.1. Individual User Book Creation

The simplest usage of DiType is where a single user invokes DiType to publish one or more books. After installing and configuring the application as necessary, the user uses one of the bundled applications to produce documentation. Such documentation typically includes one or more of the following: educational material, technical manuals, user guides, release notes, journal articles, and books.

The most common XML dialects used to write documentation are DocBook and DITA. However, DiType can be configured to handle any XML files that comply with any Document Type Definition (DTD) or XML Schema.

The following figure shows this usage scenario:

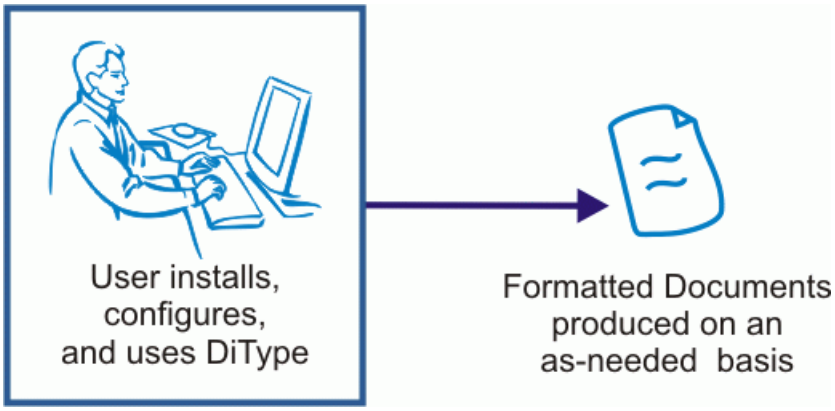

Figure 1.1. Individual User Book Creation

1.1.2. Corporate Documentation System

DiType can be used as part of a corporate content management system (CMS). When a company wishes to use DiType to produce their documentation, DiType must be integrated as a component of the CMS's publishing engine. When integrating DiType into a CMS, the administrator uses one of the available connection methods to integrate the DiType component into the CMS (see Chapter 7, *Integrating DiType*).

When integrating DiType into a corporate CMS, the information architect should assess the volume of documentation to be produced, as well as the required production rate, and match the server's hardware to achieve the desired performance and fail-safety. DiType is infinitely scalable, and the organization can add servers, CPUs, and memory as needed to ensure good output performance. DiType fully supports parallel documentation processing.

In this scenario, documentation authors have little or nothing to do with the installation, integration, or configuration of DiType. These tasks are handled by the content architect, the organization's IT group, or the CMS administrators.

The following figure shows this usage scenario:

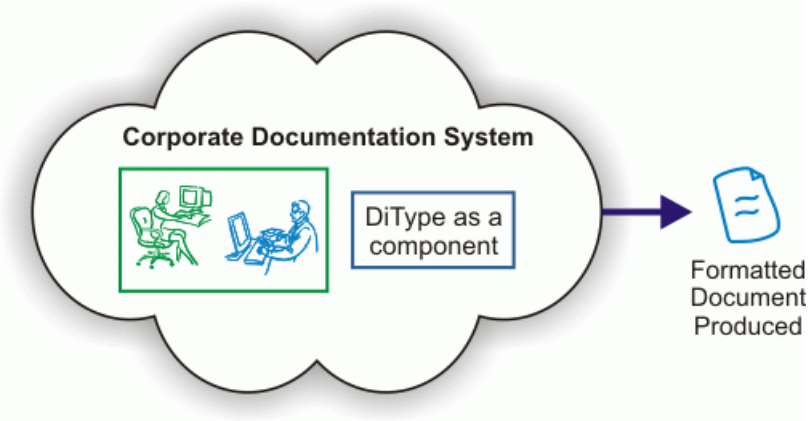

Figure 1.2. DiType in a Corporate Documentation System

1.1.3. Mass Production of Documents

Mass production of documentation is a special case of integrating DiType into a corporate CMS or documentation production system. DiType is integrated into the existing corporate content publishing system using one of the available connection methods (see Chapter 7, *Integrating DiType*). In many cases, DiType replaces a reporting application, or coexists with a reporting application that generates the XML documents DiType publishes into typeset forms (such as bills).

When mass-producing documents, it is necessary for the throughput of the system to be as high as possible. This high throughput is achieved by linear scaling; i.e., by adding more processors as the number of pages of required output documentation increases.

A bank implementing DiType in this way will require the ability to produce several million pages of text that are of a simple design, but within just a few hours. The required throughput will be X number of pages per minute, assuming that all the documents are of the same basic design, but differing in length.

To achieve the desired average throughput, the document processing must be distributed among all of the available processors, organized in a grid, with load balancing. This paralleliza-

tion of the processing can be implemented with J2EE mechanisms, or with RenderX En-Masse. Such software tools distribute the document processing jobs across all the servers that are part of the grid and implement the required load balancing. This distributed approach ensures that the failure of a single processor/server will not stop the overall document processing throughput, and will only minimally impact the average response time of the system.

The following figure shows this usage scenario:

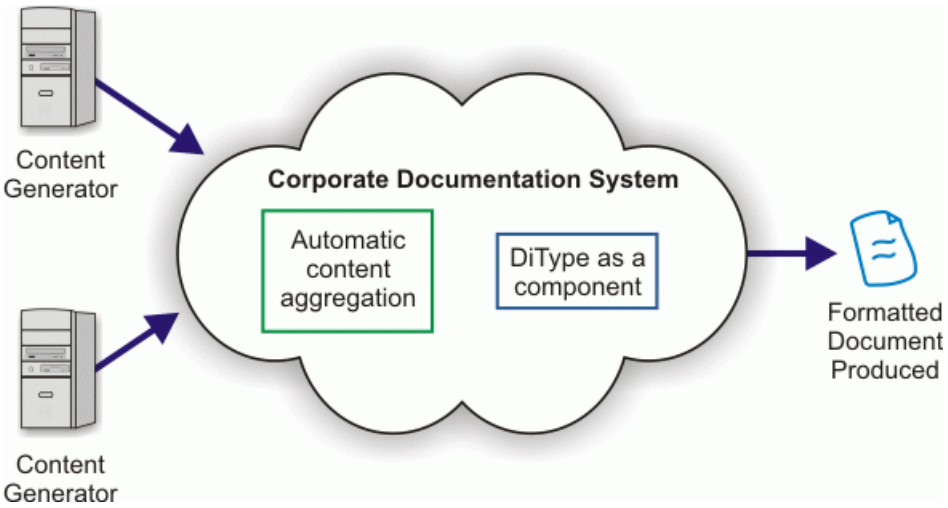

Figure 1.3. DiType in Document Mass-Production

1.1.4. Small Office/Home Office

DiType is also suitable for use in a SOHO environment. In this scenario, DiType may be installed on the local network (see Section 3.3.6, “Sharing ActiType”). In a SOHO environment, the users use one of the bundled standard applications to produce documentation. Documentation is produced on an as-needed basis; for example, creating a project plan, writing technical references, or creating a user guide for a product.

The following figure shows this usage scenario:

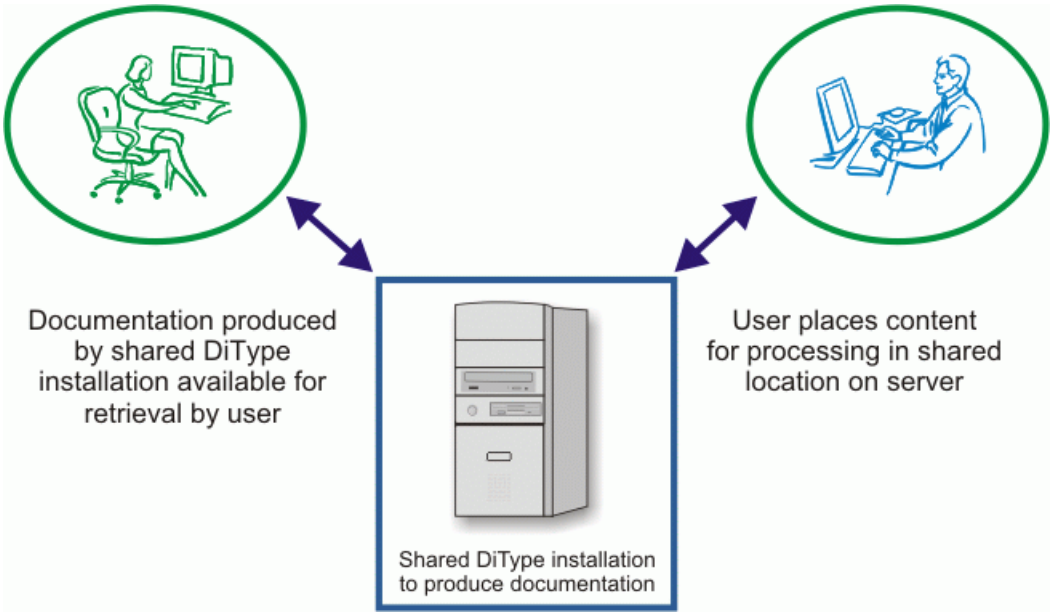

Figure 1.4. DiType in a Small Office/Home Office

Chapter 2. Installation

This section describes the DiType installation procedures for the supported platforms.

2.1. Windows

This section describes the procedure to install DiType on a Windows platform. The distribution has been tested on Windows XP. Older versions of the Windows family are not supported.

The DiType installation for Windows is contained in a Windows Installer (.msi) file named **ditype-<version>-<release date>-windows.msi**.

To install DiType on a Windows platform:

1. Launch the DiType installer.

   The Welcome screen of the DiType InstallShield Wizard appears.

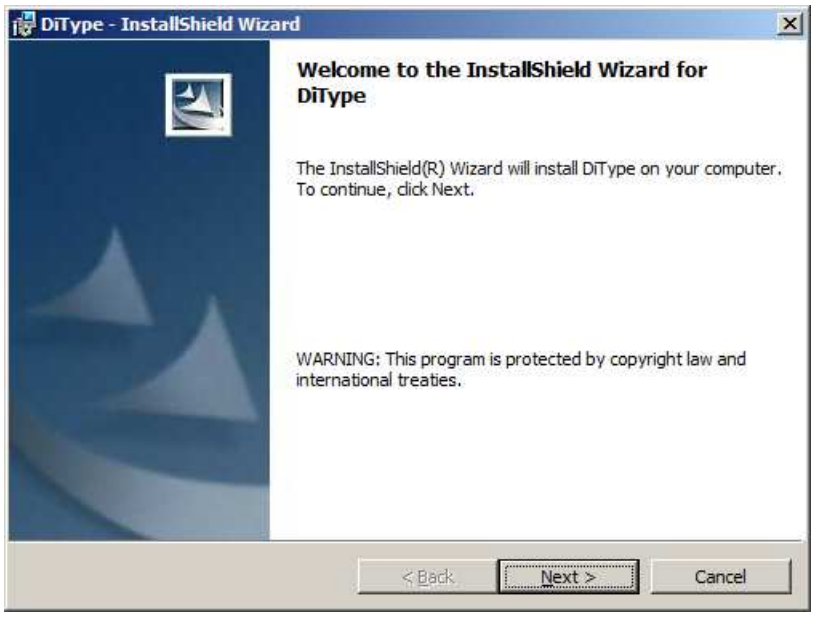

Figure 2.1. DiType Welcome Screen

2. Click **Next**.

   The License Agreement screen of the InstallShield Wizard opens.

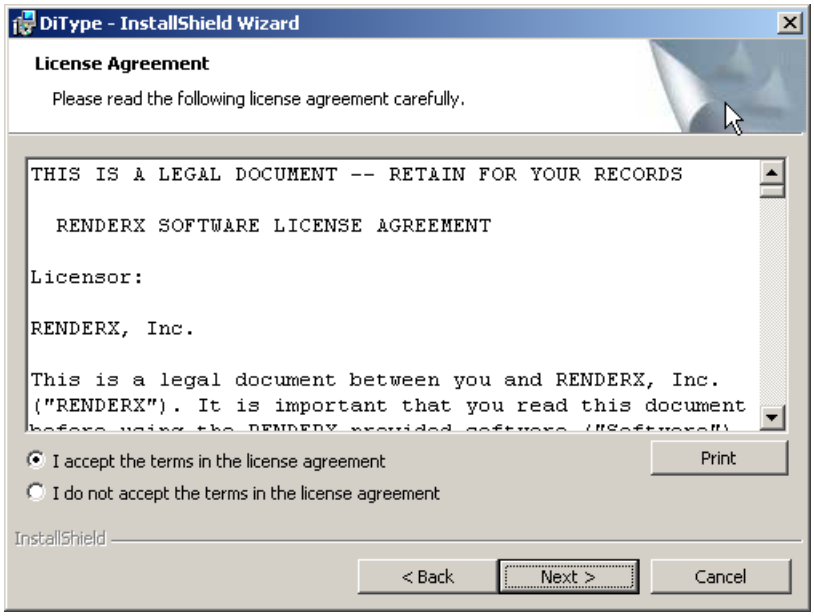

Figure 2.2. License Agreement Screen

3. (Optional) Click **Print** to print out the license agreement.
4. Select **I accept the terms in the license agreement** and click **Next**.

   The Destination Folder screen of the InstallShield Wizard opens.

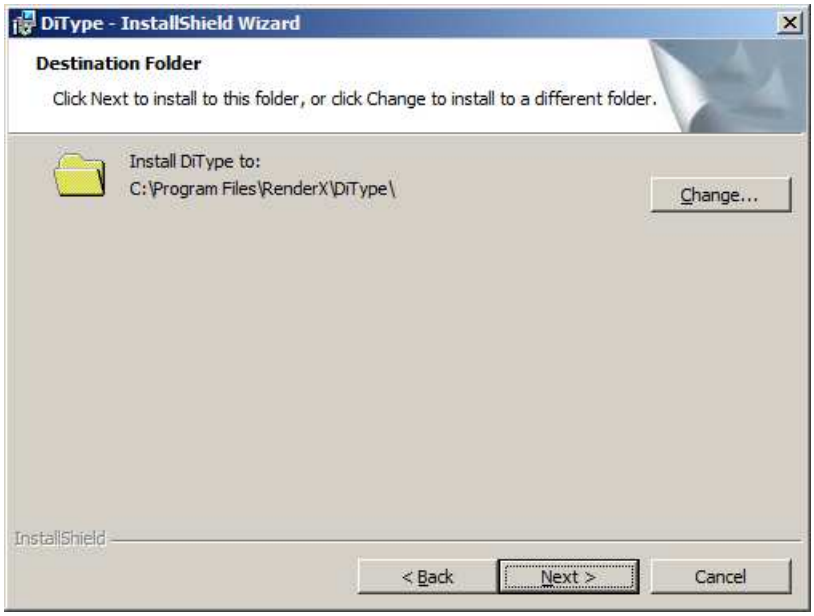

Figure 2.3. Destination Folder Screen

5. (Optional) Click **Change** and choose a different destination folder.

6. Click **Next**.

   The Setup Type screen of the InstallShield Wizard opens.

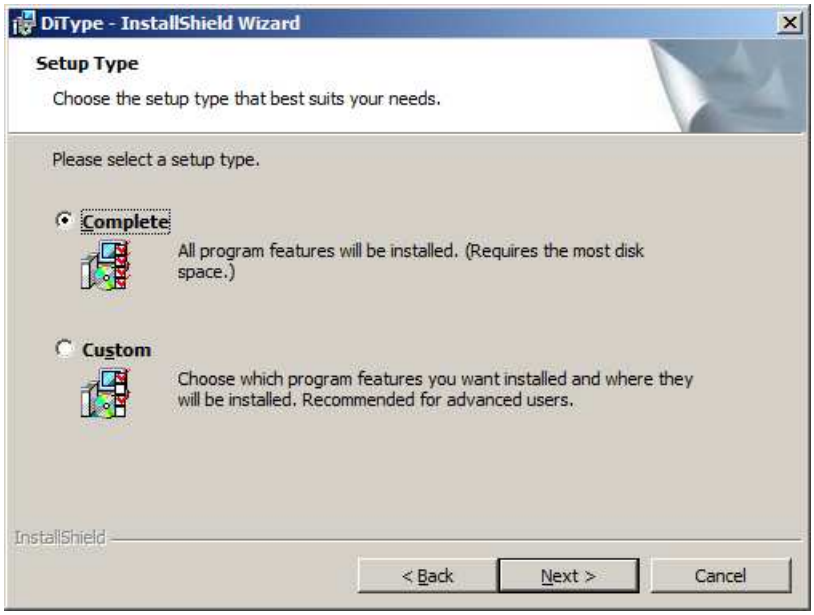

Figure 2.4. Setup Type Screen

7. Select the setup type and click **Next**.

   The Ready to Install screen of the InstallShield Wizard opens.

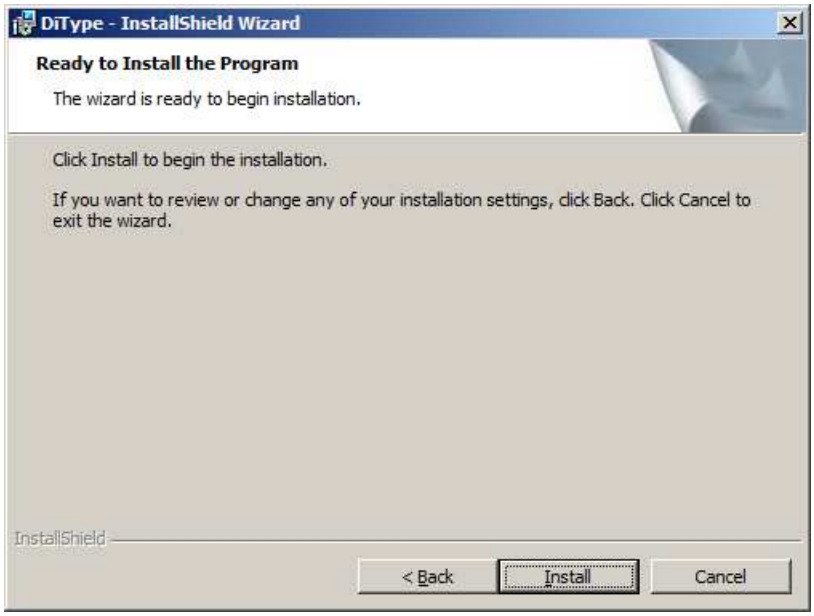

Figure 2.5. Ready to Install Screen

8. Click **Install**.

   The Installing screen of the InstallShield Wizard opens.

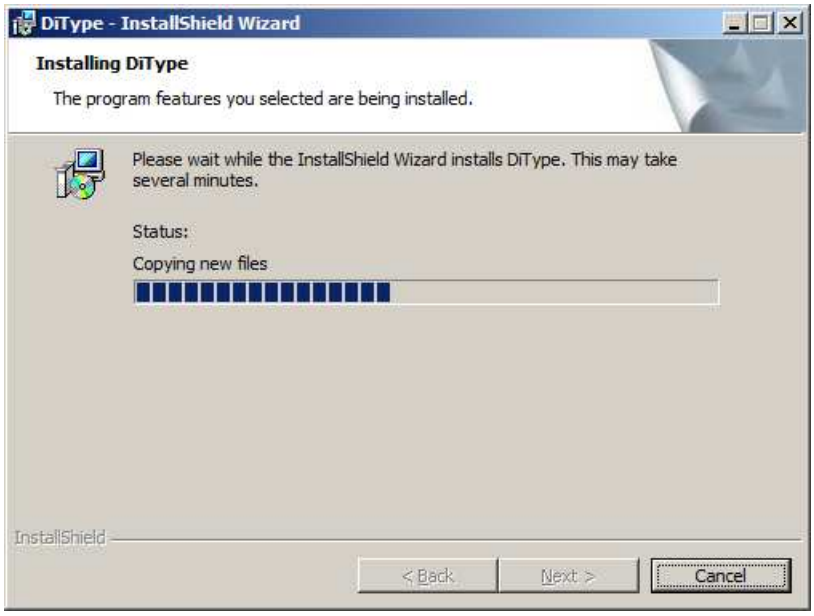

Figure 2.6. Installing Screen

The DiType License screen opens.

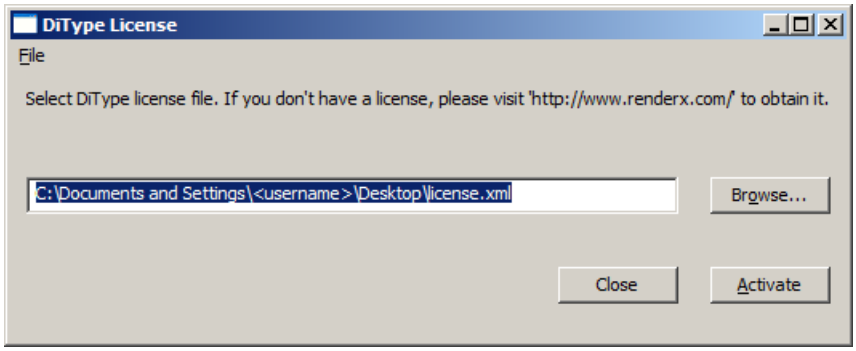

Figure 2.7. DiType License Screen

9. Click **Browse** to locate your license file and click **Activate**.

   The Installation Complete screen of the InstallShield Wizard opens.

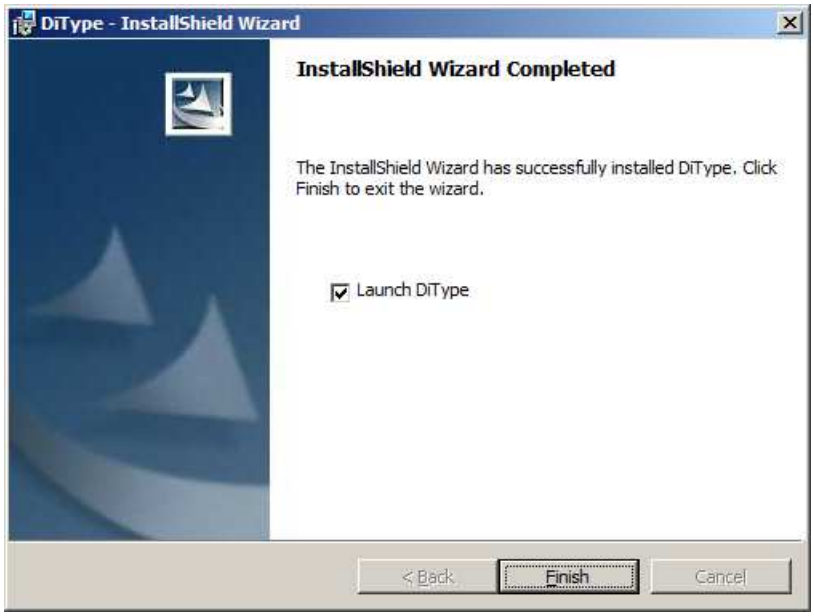

Figure 2.8. Installation Complete Screen

10. (Optional) Uncheck the **Launch DiType** option.

    Note: When the Launch DiType option is checked, the last step of the installer will be to register and start the DiType service. If you leave this box empty, you are required to start DiType manually or register the service yourself with the help of scripts provided. See Section 5.1, “Windows” for details.

11. Click **Finish**.

2.2. Mac OS X

This section describes the procedure to install DiType on Mac OS X. The DiType Mac OS X distribution can run on a PowerPC or on Intel; however, the application binaries are different.

The installation package is a disk image file named **ditype-<version>-<release date>-intel.dmg** or **ditype-<version>-<release date>-ppc.dmg** (choose the image file to match your hardware) containing the following files:

- **ditype-<version>-<release date>-intel.mpkg** - The metapackage containing DiType Framework, DiType Assistant and ActiType.
- **uninstall-ditype.sh** - The DiType uninstaller.

To install DiType on Mac OS X:

1. Open the installation package .dmg file.

Figure 2.9. Disk Image file contents

2. Open the DiType installation metapackage.

   The Welcome screen of the DiType Installer opens.

Figure 2.10. Welcome Screen

3. Click **Continue**.

   The Software License Agreement screen of the DiType Installer opens.

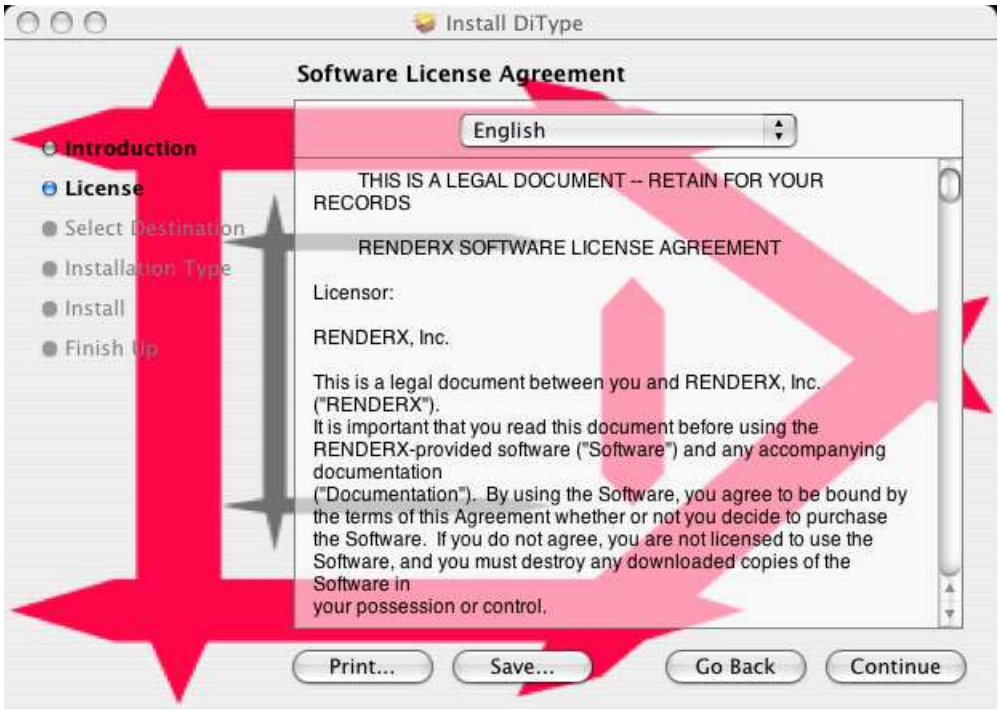

Figure 2.11. Software License Agreement Screen

4. (Optional) Click **Print** to print the license agreement, or click **Save** to save the license agreement.

5. Click **Continue**.

   The Software License Agreement confirmation prompt opens.

Figure 2.12. Software License Agreement Confirmation Prompt

6. Click **Agree**.

   The Select Destination screen opens.

Figure 2.13. Select Destination Screen

7. (Optional) Click **Choose** and choose a different destination folder.
8. Click **Continue**.

   The Install Type screen opens.

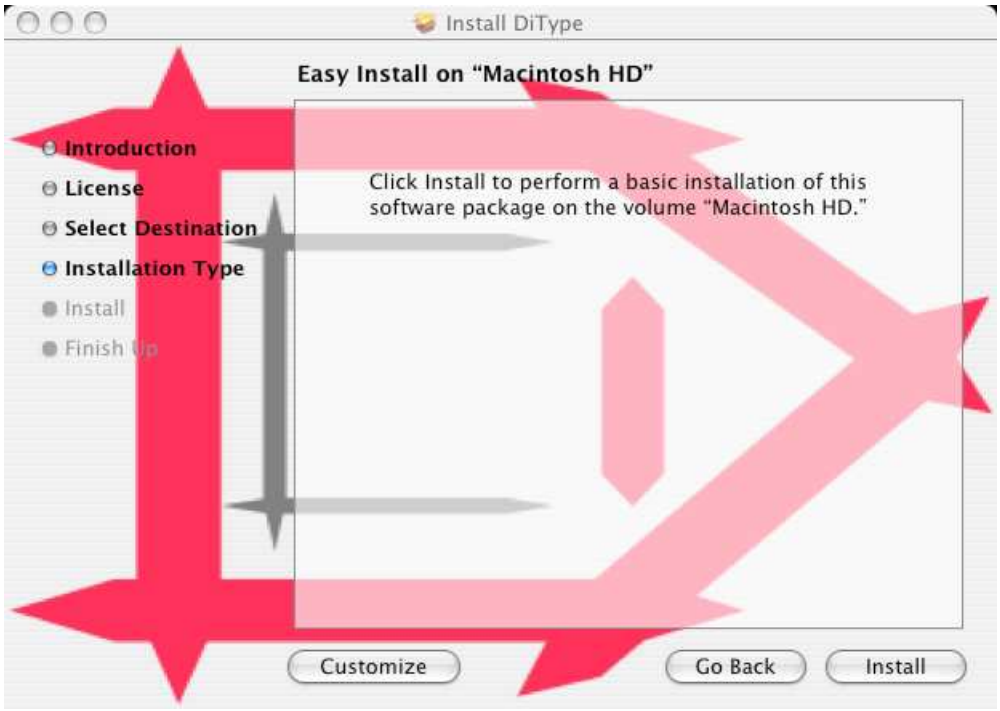

Figure 2.14. Install Type Screen

9. Click **Install**.

   The Authenticate prompt opens.

Figure 2.15. Authenticate Prompt

10. Type in your username and password and click **OK**.

    The DiType License screen opens.

Figure 2.16. DiType License Screen

11. Click **Browse** to locate your license file and then click **Activate**.

    After DiType is installed, the Finish screen opens.

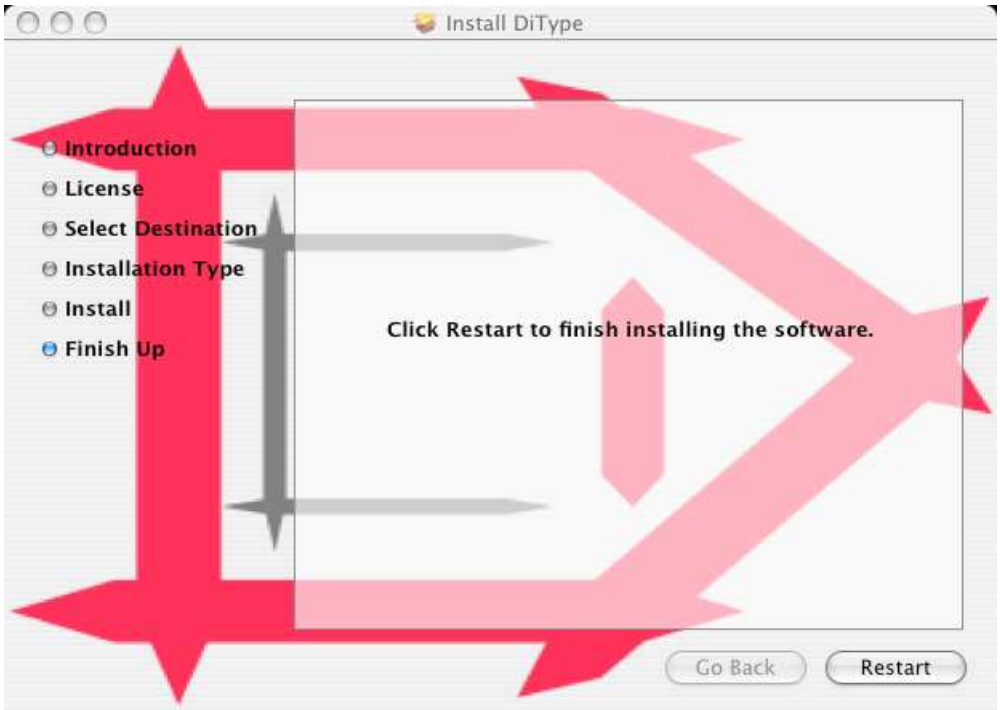

Figure 2.17. Finish Screen

12. Click **Restart** to restart your computer and complete the installation.

2.3. Unix/Linux

This section describes the procedure to install DiType on Unix/Linux platforms.

To install DiType on Unix/Linux platforms:

1. Unpack the installation files with the tar utility (or any other suitable utility).

2. Install DiType using one of the following methods (text that you enter is highlighted in **bold**):

```
$ ./install.sh
linux-gnu
Ditype installation directory [/opt/RenderX/ditype] : /path/to/ditype-linux
Directory for temporary files [/tmp/ditype] : /path/to/ditype-linux/tmp
License file [license.xml] : /location/of/license.xml
Installation directory: /path/to/ditype-linux
Temporary directory: /path/to/ditype-linux/tmp
License: /location/of/license.xml
Please wait...
Copying license file to
'/path/to/ditype-linux/etc/license.xml'
Done
$
```

or without interactive input,

```
$ D=/path/to/ditype-linux
$ T=$D/tmp
$ L=/location/of/license.xml
$ ./install.sh -d $D -t $T -l $L
Installation directory: /path/to/ditype-linux
Temporary directory: /path/to/ditype-linux/tmp
License: /location/of/license.xml
Please wait...
Copying license file to
'/path/to/ditype-linux/etc/license.xml'
Done
$
```

3. Start the server by navigating to the target directory and run the following command:

```
$ bin/ditype-server start
```

Chapter 3. Standard Applications

The DiType installation contains three simple applications that expose the functionality of the DiType engine. The three applications are:

- DiType Assistant - A user-friendly GUI tool.
- DiType Command Line - A command line interface to the DiType formatting engine.
- ActiType - A tool to aid automatic formatting of XSL files to PDF, PostScript and SVG (or ZIP) output.

These applications are intended for personal use to help new users get started using DiType. The applications are detailed in the following sections.

Note: All figures in this chapter show the applications running on a Windows platform. On other platforms, the applications contain identical features that are displayed according to the platform's user interface.

3.1. DiType Assistant

This section describes the DiType Assitant.

3.1.1. What is the DiType Assistant?

XEP DiType contains a user-friendly GUI tool, called the DiType Assistant. Using the DiType Assistant simplifies rendering from XML or XSL-FO into the desired output format.

3.1.2. Opening the DiType Assistant

To open the DiType Assistant:

- On Windows, from the **Start** menu, choose **All Programs** > **RenderX** > **DiType** > **DiType Assistant** .
- On Mac OS X, DiType Assistant is installed in the **Applications** menu.
- On Linux/Unix the script that invokes DiType Assistant is **bin/assistant** in the DiType installation directory.

3.1.3. Rendering an XML File using the DiType Assistant

Opening a File

To render a file, you must first open the XML or XSL-FO file you wish to publish.

To open an existing XML or XSL-FO file:

1. From the **File** menu, click **Open.**

   A dialog box is displayed**.**

2. Browse to the file you wish to open.

   The file is opened within DiType Assistant.

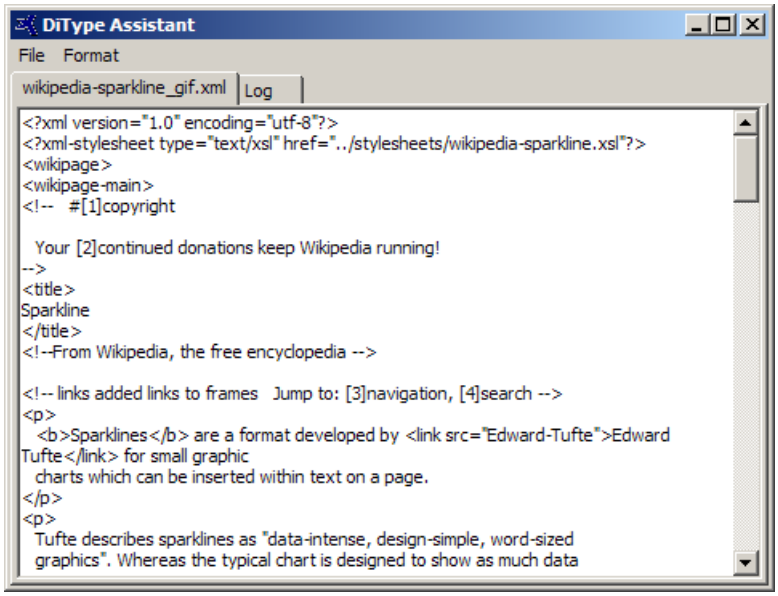

Figure 3.1. XML file displayed in the DiType Assistant

Formatting a File

Formatting a file is a two-stage process comprising transforming and rendering. After an XML file is opened, it must be transformed before it can be rendered to PDF or PostScript output.

The transformation stage of the formatting process assigns the settings required for applying an XSL stylesheet to the XML file. The XML file is transformed into an XSL-FO. The XSL-FO is then rendered to your final output format (PDF, PS or SVG).

To format an XML file:

1. From the **Format** menu, click **Start.**

The **Formatting Settings** dialog box appears.

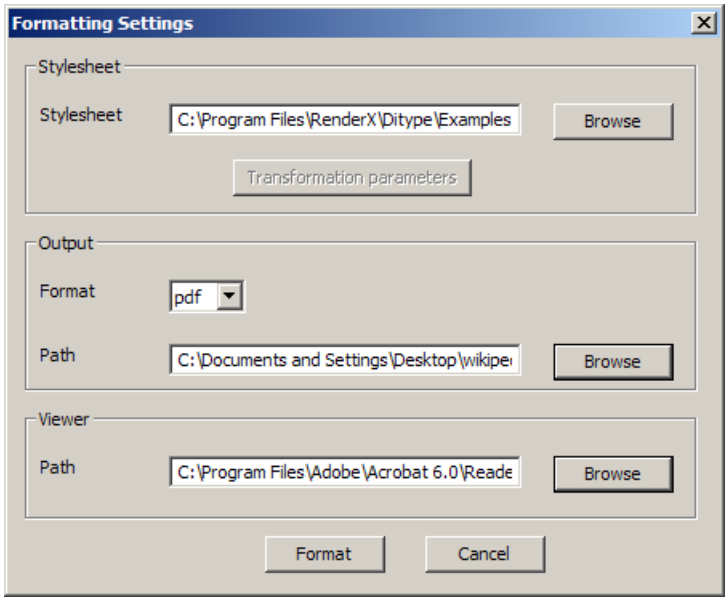

Figure 3.2. Formatting Settings dialog box

2. Set the desired settings as described in the following table.

Table 3.1. Formatting Settings

Parameter	Description
	Stylesheet
Stylesheet	Select the name and location of the stylesheet you wish to apply to your XML file.
	Output
Format	Select the format to which you want to render the XML file. Available options are PDF, XEP, PS, SVG (or ZIP).
Path	Select the location and name of the file to which the output will be saved. **Note:** If a file with the same name already exists in the chosen location, the new file will overwrite the preexisting file with no warning.
	Viewer
Path	Browse to the location of the program with which you wish to view the rendered file.

3. Click **Format** to format the file, or **Cancel** to cancel the formatting.

To cancel formatting:

- From the **Format** menu, click **Stop**.

  Formatting is canceled.

3.2. DiType Command Line

This section describes how to run DiType from the command line.

3.2.1. Running DiType

DiType can be run from the command line using the `ditype` Command Line Tool.

- On Windows, `ditype` is located in the DiType installation directory in the `Framework/Commands/` directory.
- On Mac OS X, `ditype` is located in the `/Library/Frameworks/DiType.framework/Commands` directory.
- On Linux/UNIX, `ditype` is located in the DiType installation directory in the `bin/` directory.

`ditype` is a batch/shell script that invokes `diclick.py`, which is a Python application that communicates with the DiType server.

The syntax of the Windows batch and Linux/Mac shell command is:

```
ditype {switches} {arguments}
```

This syntax assumes the full path to the Windows batch file `ditype.bat` or the Linux/Mac shell script `ditype` is specified in the `PATH` environment variable, or that the current directory is the directory containing the Windows batch file or Linux/Mac shell script.

The switches and arguments are the same whether `ditype` is run via a Windows batch file or via a Linux/Mac shell script.

3.2.2. ditype Switches

The `ditype` switches configure the behavior of the command line utility.

Table 3.2. `ditype` Switches

Switch	Description
--version	Prints the program version and the exit option.
-h, --help	Displays the usage syntax and the exit option.

Switch	Description
-H <host> --host=<host>	Sets the host name or IP address of the computer on which the DiType server is running. The command line client will connect to that host and make the DiType server do the formatting. The default value is 'local-host'.
-p <port> --port=<port>	Sets the port number of the computer running the DiType server. The default value is 19790.
-i <format> --input=<for-mat>	Sets the format for the input files to the DiType rendering engine. The possible values for this switch are 'xsl', 'fo', 'dbx', 'dita', and 'auto'. The default value is 'auto'. When set to 'auto', the DiType server automatically determines the file type and applies the corresponding transformation, if configured.
-f <format> --format=<for-mat>	Sets the format for the output files from the DiType rendering engine. The possible values for this switch are 'ps', 'pdf', 'xep', 'svg' (or 'zip') for PostScript, PDF, DiType Intemediate Format, and SVG, respectively. The default value is 'ps'. Other values that correspond to document types known to `transformer.conf` can also be specified. This assumes that there is a path over the set of types from the input format to the output format. For example, to transform an arbitrary ('auto') input to 'fo', specify '-f fo'. In this case, only transformations will occur, the file will not be rendered.

-s <SysId>
 	
--sysid=<SysId>
	When a file to format is supplied as an argument to `ditype`, the DiType server will ask the client to provide locally referenced files; for example, stylesheets or images (if any). The client will send these files to the server, resolving relative links from the input file itself.
	
	When `ditype` is invoked without arguments, it works as a pipe, reading input on `stdin` and writing output to `stdout`. In this case, the client does not have the information about the actual location of the input. In order to resolve relative links properly, the user must specify SysId so that if the input stream was read from a file located at this SysId, the relative links would point to the appropriate location.
	
	The default value of SysId is `file://$PWD/stdin`. In general, `ditype` users may omit the -s option if the input stream originates from a file in the current directory:
	
	`cat afile.fo | ditype`
	
	However, it is nesessary to specify the proper SysId in other cases:
	
	`cat /path/to/afile.fo | ditype -s "file:///path/to/afile"`

Switch	Description
-a, --append	Specify this switch to append the file extension to the output filename. If the input filename is `input.xml` and the output format is 'ps', the output filename will be `input.xml.ps`. By default, the output file extension replaces the input file extension. This option has no effect when running as a pipe.
-q	By default, `ditype` writes detailed messages to `stderr`. These messages indicate the current status and progress of the rendering process, as well as any warnings or errors that may occur during the rendering process. Specify this switch to suppress the detailed informational messages. In this case, the renderer outputs only warning and error messages.
-d	Specify this switch to include additional debugging information during the rendering process. By default, debugging is not enabled. It is possible to specify both the '-q' and '-d' switches.

3.2.3. ditype Arguments

The `ditype` arguments are one of the two ways to provide DiType with the data to process.

Note: If any argument string contains spaces, the entire string must be enclosed in quotation marks.

An xml/xsl pair of files may not be passed to ditype. Instead, add the xml-stylesheet processing instruction to the source file immediately after the XML declaration, as shown in the following example.

```
<?xml-stylesheet type="text/xsl" href="stylesheets/mystylesheet.xsl"?>
```

3.3. ActiType

ActiType is a simple application used for asynchronous formatting of XSL files to PDF, PostScript and SVG (or ZIP) output. ActiType constantly monitors its active folders. When a file is placed in an active folder for processing, ActiType formats it via DiType to a specified format according to the configured rules.

This section describes the ActiType application and how to share ActiType between multiple users.

3.3.1. Opening ActiType

To open ActiType in Windows, from the **Start** menu, choose **All Programs** > **RenderX** > **DiType** > **ActiType**.

The ActiType Window

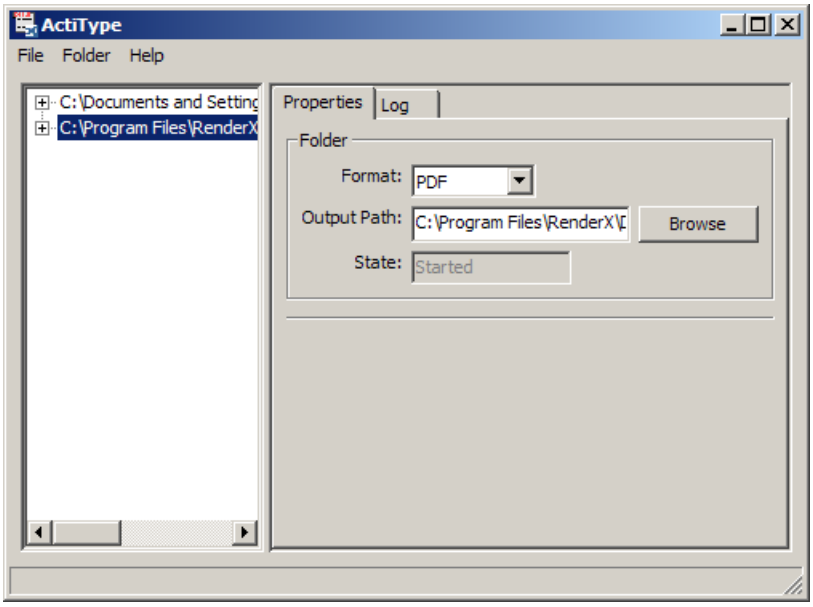

Figure 3.3. ActiType Window

The ActiType window consists of two panes:

- Left pane - All hot folders, which are folders added to the ActiType configuration file for formatting, are displayed.

  Click on a hot folder to expand its node and view all rules that are configured for the specific folder.

- Right pane - Displays properties and log of the hot folder in currently focus.

3.3.2. Formatting an XML or FO File using ActiType

In order for a document to be formatted you need to create a hot folder and to configure a rule or multiple rules to be applied to the hot folder. ActiType monitors all active hot folders: it detects new files added to the hot folders and automatically formats them via DiType according to the rules specified for the folder.

Formatting a file consists of the following steps:

1. Creating a Hot Folder
2. Configuring Rules for the Hot Folder
3. Starting the Hot Folder
4. Formatting a File

Creating a Hot Folder

To format a file, you must first create a hot folder by adding a folder to the ActiType configuration folder.

To create a hot folder:

1. From the **Folder** menu, click **Add.**

   A dialog box is displayed**.**

2. Browse to the folder you wish to create as a hot folder.

   The folder is added to ActiType as a hot folder.

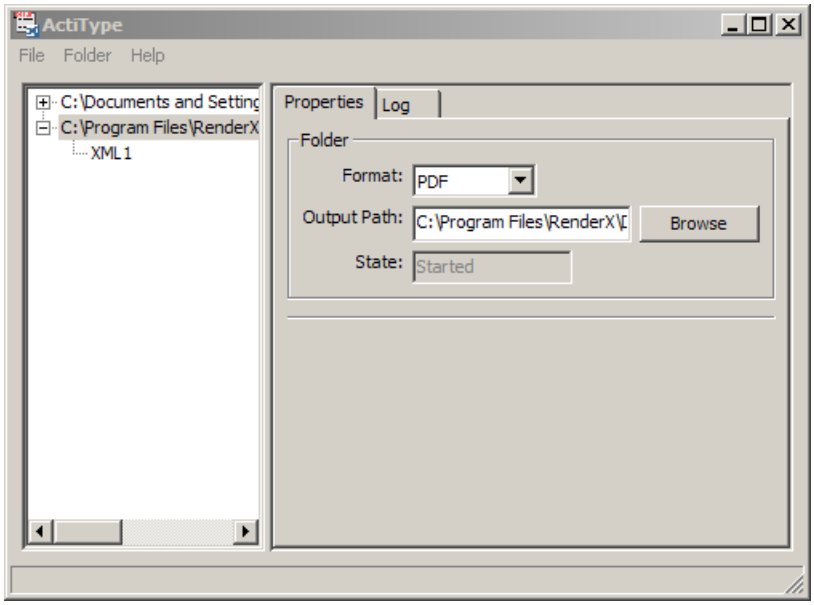

Figure 3.4. Folder Added to DiType

3. Set the desired properties, as described in the following table.

Table 3.3. Hot Folder Properties

Parameter	Description
Format	Select the format to which you want to format files within the hot folder. Available options are PDF and PostScript.
Output Path	Select the location and name of the file to which the output will be saved.

Parameter	Description
	Note: If a file with the same name already exists in the chosen location, the new file will overwrite the preexisting file with no warning.
State	The current state of the folder. This is a non-editable field that reflects the status of the hot folder. • Started - The folder is being monitored and when a new file is detected it will be formatted according to the rules associated with it. • Stopped - The folder is not being monitored and files within the folder will not be formatted.

Configuring Rules for the Hot Folder

Once a hot folder is created, you must configure rules that define which type of files inside the hot folder will be formatted and which stylesheet will be used for formatting.

To add a rule:

1. Click on the hot folder.
2. From the **Folder** menu, click **Rules > Add**.

   A new rule is added.

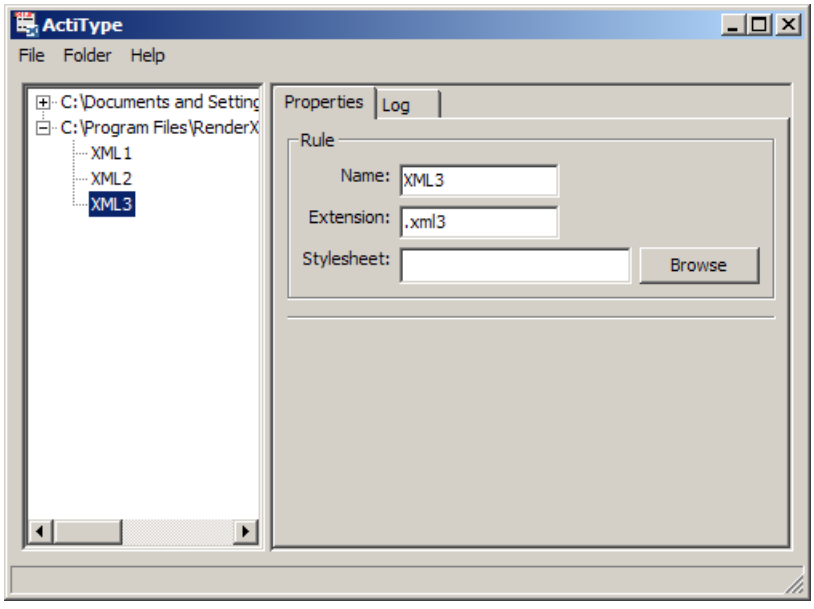

Figure 3.5. Adding a Rule

3. Set the desired properties, as described in the following table.

Table 3.4. Rule Properties

Parameter	Description
Name	Specify a name for the new rule.
Extension	Specify an extension. All files inside the hot folder with the extension specified will be formatted. All other files will be ignored.
Stylesheet	Select the name and location of the stylesheet which will be used for formatting the files within the hot folder.

To edit a rule:

1. Click on the rule you wish to edit.
2. Make any changes you wish to the name, extension, or stylesheet fields.

   An Apply button appears.

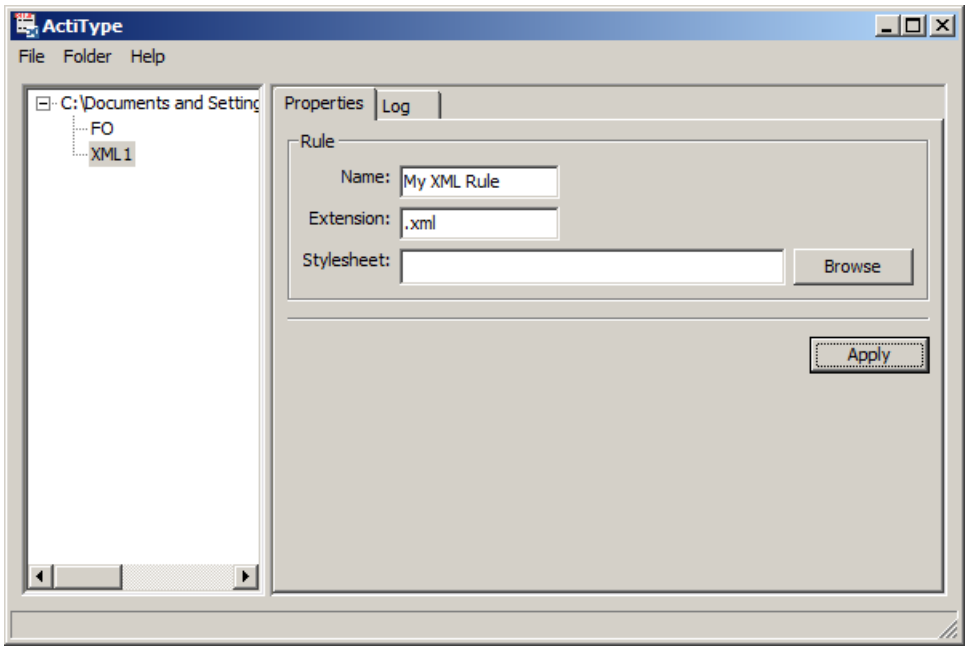

Figure 3.6. Editing a Rule

3. To apply the changes to the rule, click **Apply**.

   The rule is updated.

To delete a rule:

1. Click on the rule you wish to delete.
2. From the **Folder** menu, click **Rules > Delete**.

   The rule is deleted.

Starting the Hot Folder

In order for a hot folder to be monitored, it must be started.

To start the hot folder:

1. Click on the hot folder.
2. From the **Folder** menu, click **Start.**

   The hot folder status is now "Started." The folder will now be monitored and all new files added to the folder will be detected and formatted according to the rules configured.

Formatting a File

To format a file, simply copy it into your hot folder. It will then be detected and formatted according to the rules you defined for the hot folder.

3.3.3. Stopping a Hot Folder

A hot folder can be stopped so that it is no longer monitored.

To stop a hot folder:

1. Click on the folder you wish to stop.
2. From the **Folder** menu, click **Stop.**

   The hot folder status is now "Stopped." The folder will not be monitored and new files added to the folder will not be formatted.

To stop all hot folders:

- From the **Folder** menu, click **Stop All.**

  All hot folders are stopped.

3.3.4. Closing the Application

To close ActiType:

- From the **File** menu, click **Close.**

  The ActiType GUI is closed. However, ActiType continues to run in the system tray, monitoring all hot folders and formatting any new files added to the hot folders.

To quit ActiType:

- From the **File** menu, click **Exit.**

  ActiType is now inactive. It is no longer running in the system tray.

3.3.5. Running ActiType in Console Mode

During the DiType installation, a link to ActiType is installed in the **Startup** menu of the user. When the computer is restarted, ActiType also starts and appears as an icon in the System Tray (Windows only). Currently, the icon only supports opening the main ActiType window.

ActiType may also be run in **console mode**, that is, without the icon in the System Tray. In this mode, users may not use the GUI to change the configuration or to stop and start folders. The advantage of running ActiType in console mode is that it does not require a user log in. This is essential when ActiType is installed on an Intranet server.

If ActiType is not currently running, it can be started in console mode by running one of the following command at the command prompt:

```
python actitype.pyw -n -c actitype.conf
```

or

```
python actitype.pyw --non-gui-mode --config actitype.conf
```

Configuration File

ActiType is configured using a configuration file. The configuration file can be edited in any text editor, and the configuration can then be applied either in console mode or by using the ActiType GUI.

The following is a sample configuration file:

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

<!-- ActiType configuration file -->

<actitype>
    <folder path="C:\My Documents\ActiType Hot Folder"
```

```
            output-path="C:\My Documents\ActiType Output Folder"
            output-format="application/pdf"
            state="started"
            <rule name="XML" ext=".xml"/>
    </folder>
</actitype>
```

If the name of the sample configuration file is `MyActiTypeConfig.conf`, it can be loaded by running the following command at the command prompt:

```
python actitype.pyw -n -c MyActiTypeConfig.conf
```

3.3.6. Sharing ActiType

It is possible to share an installation of ActiType between multiple users in a Windows environment. The steps required to share an ActiType installation are described in the following section.

Note: This procedure assumes that there is a dedicated computer running Windows with DiType installed with the default options.

To share ActiType:

1. Navigate to `C:\Documents and Settings\<your_user_name>\Start Menu\Programs\Start-up` and delete `ActiType.lnk` - this prevents ActiType from being started twice.

2. Create a new shared folder (it must be viewable on the Windows network from other computers):

   ```
   C:\Documents and Settings\All Users\Documents\Shared ActiType Hot Folder
   ```

3. Add the new folder to the ActiType configuration file:

   ```
   <folder path="C:\Documents and Settings\All Users\Documents\
           Shared ActiType Hot Folder"
           output-path="C:\Documents and Settings\All Users\Documents\
           Shared ActiType Hot Folder"
           output-format="application/pdf"
           state="started"
      <rule name="FO" ext=".fo"/>
   </folder>
   ```

4. From **Control Panel**, open **Scheduled Tasks** and select **Add Scheduled Task**.

   Apply the following parameters for the task:

   - Run:

```
C:\PROGRA~1\RenderX\DiType\FRAMEW~1\Common\env\Python2.4\pythonw.exe -E
"C:\Program Files\RenderX\DiType\Actitype\actitype.pyw"
-n -c actitype.conf
```

- Start in: `C:\Program Files\RenderX\DiType\Actitype\`
- Run as: <**your credentials, password required**>
- Uncheck **Run only if logged on**.
- Check **Enabled**.
- On the **Schedule** tab, select **At System Startup**.
- On the **Settings** tab, uncheck all boxes.

5. Reboot, but do not log on.

   You can now copy source files to the shared hot folder and view the resulting output.

Chapter 4. Configuring DiType

This section describes how to configure DiType and the configuration files included with the DiType installation.

4.1. Configuring CLISER

CLISER is an internal protocol through which applications communicate with the DiType core. The two CLISER configuration parameters are described in the following table:

Table 4.1. CLISER Configuration Parameters

Parameter	Description
data port	The DITYPE_DATA_PORT environment variable determines the port through which data is communicated between applications and the DiType core. The default value is 19790.
http port	The DITYPE_HTTP_PORT environment variable determines the port through which http communications take place. The default value is 19800.

Note: You must open the selected ports, either user-defined or default value, in the firewall.

4.2. DiType Configuration Files

The DiType installation contains several configuration files that can be changed by the user. After changing any of the files, it is necessary for the file to be validated. The validation is performed using one of the build-in RelaxNG schemas.

- `ditype.conf` - This is the main configuration file for the DiType engine. It specifies the URL of the DiType root directory and license file, and the XSL-FO and XEPOUT validation schemas. It also specifies the spaces (such as lambda and pi) and services (such as the graphic-server and the transformer) that will be used and any relevant options.

  The `ditype-conf.rnc` schema validates this file.

- `arx.conf` - ARX is a tool to automatically determine the type of a document from its name and contents. This configuration file contains the list of grammars that DiType can process and the rules that the ARX tool uses to recognize documents that use one of the listed grammers.

- `font-server.conf` - Specifies the fonts to be used by DiType.

  The `font-server-conf.rnc` schema validates this file.

- `transformer.conf` - The transformer uses the ARX tool to define the type of the input document.

The `transformer-conf.rnc` schema validates this file.

- `xep.conf` - This configuration file is a supplementary configuration file for XEP-DiType.

  The `xep-conf.rnc` schema validates this file.

Chapter 5. Administration

There are a number of administrative tasks that can be performed to provide administrators with further information about the DiType installation and status, and to allow administrators to change the default characteristics of the DiType application.

Controlling the DiType server on `http://localhost:19800`	The DiType server provides a web interface to monitor and control its activity. Using this interface, it is possible to shutdown, kill, suspend, or resume the DiType server. The monitoring facility provides a table describing each session, including the following information: • The host port from which it originated • The timing information • Whether the formatting session was successful or not
Monitoring the temporary folder	The temporary folder is configured in the `ditype.conf` configuration file. The temporary folder is set to the value of the "temporary-directory" option in the "file-server" service section of the configuration file. The temporary folder is used to store cached external resources and intermediate files. When a formatting session ends, whether it was successful or not, all of the files in the temporary folder that were produced during the formatting session are removed. Thus, normally there is no requirement to manually remove temporary files from the folder. However, when the DiType server is started in debug mode, the files are not removed from the temporary folder. When the DiType server is started in debug mode, an administrator is required to manually clean the temporary folder.
Environment variables	The DiType server, the standard applications (DiType Assistant, ActiType and `ditype`), and the connectors included in the DiType DevKit are affected by the following environment variables: • DITYPE_HOST (default: localhost) • DITYPE_DATA_PORT (default: 19790) • DITYPE_HTTP_PORT (default: 19800) These variables, if specified, allow clients to locate a running DiType server to which they are able to connect. The DITYPE_DATA_PORT

	and DITYPE_HTTP_PORT variables force the server to use the specified ports for data exchange or monitoring.
Starting the server with debug-level logging	It is possible to start the DiType server in debug mode. This causes detailed messages to be sent to each session's logger, and in Lisp REPL listening on TCP port 12345 as configured in the `ditype.conf` configuration file. With debug-level logging turned on, temporary files are not removed. The server will start in debug mode if `cliser.py` is given the switch `-d`.
Firewall-related issues	The DiType server communicates data with clients on TCP port 19790 and provides a web interface on TCP port 19800. It uses TCP port 6210 and TCP port 6610 internally for inter-space data exchange and provides a Lisp REPL on TCP port 12345 when in debug mode. The port numbers given here are the default values, they may be altered via environment variables or by changing the relevant value in `ditype.conf`.
Optimizing performance	A number of performance-related options that are present in ditype.conf may alter the performance of the DiType server. These are the options for the "lambda" space, which are set by default to the optimal values.

5.1. Windows

This section describes the administration tasks that can be carried out in a Windows environment.

5.1.1. Files

On Microsoft Windows, the DiType installation creates by default the following files and directories:

- `%ProgramFiles%\RenderX\DiType\Framework`

5.1.2. Processes

The following processes are started when the system is rebooted at the last step of the installation process:

- `ditype-service.exe`
- `pythonw.exe`

- `lambda.exe`

5.1.3. Starting Manually

It is possible to start, stop, or restart the DiType server manually from the command line; e.g., after making configuration changes. This can be done with the following command:

`net start ditype`, or `net stop ditype`

Whenever the DiType server starts, it first checks if there is an instance of the DiType server already running on the system by checking for the existence of the pid file, which usually is found in `/Library/Frameworks/DiType.framework/Common/var/ditype.pid`. Upon normal shutdown of the DiType server, the pid file is removed. However, abnormal termination may leave the pid file in place, preventing further startups of the DiType server. If this happens, remove the pid file manually.

5.2. Mac OS X

This section describes the administration tasks that can be carried out in a Mac OS X environment.

5.2.1. Files

On Mac OS X, the DiType installation creates the following files and directories:

- `/Library/Frameworks/DiType.framework`
- `/Library/Frameworks/GPLGhostscript`
- `/Library/StartupItems/DiType`
- `/Library/Applications/DiTypeAssistant.app`
- `/Library/Applications/ActiType.app`
- `/Library/Receipts/ditype-*.pkg`
- `/usr/local/bin/ditype`

5.2.2. Processes

The following processes are started when the system is rebooted at the last step of the installation process:

Note: The output of `ps` is split into several lines with '\' used as a line separator in order to fit the page width.

```
/Library/Frameworks/DiType.framework/Versions/1/Common/env/Python.framework/ \
Versions/Current/bin/python \
/Library/Frameworks/DiType.framework/Versions/1/Common/pi/ditype/cliser.py -c \
/Library/Frameworks/DiType.framework/Versions/1/Configuration/ditype.conf

/Library/Frameworks/DiType.framework/Versions/1/Common/lambda \
/tmp/lambda-Yeed3w
```

The standard GUI applications, DiType Assistant and ActiType, are represented by the following processes:

```
/Library/Frameworks/DiType.framework/Common/env/Python.framework/Versions/ \
Current/Resources/Python.app/Contents/MacOS/Python actitype.pyw \
--config /Users/msulyaev/.actitype/actitype.conf

/Library/Frameworks/DiType.framework/Common/env/Python.framework/Versions/ \
Current/Resources/Python.app/Contents/MacOS/Python assistant.pyw
```

5.2.3. Starting Manually

It is possible to start, stop, or restart the DiType server manually from the command line; e.g., after making configuration changes. This can be done with the following command:

```
/Library/StartupItems/DiType/DiType <action>
```

where <action> is `start`, `stop`, or `restart`.

Whenever the DiType server starts, it first checks if there is an instance of the DiType server already running on the system by checking for the existence of the pid file, which usually is found in `/Library/Frameworks/DiType.framework/Common/var/ditype.pid`. Upon normal shutdown of the DiType server, the pid file is removed. However, abnormal termination may leave the pid file in place, preventing further startups of the DiType server. If this happens, remove the pid file manually.

5.3. Unix/Linux

This section describes the administration tasks that can be carried out in a Unix/Linux environment.

This chapter is unfinished.

Chapter 6. Uninstalling

This section describes how to uninstall DiType.

6.1. Windows

This section describes the Windows DiType uninstall procedure.

To uninstall DiType on a Windows platform:

1. From the **Start** menu, choose **Control Panel**.
2. Open the **Add or Remove Programs** control panel applet.
3. From the **Currently installed programs** list, click **DiType**.
4. Click **Remove**.

   A confirmation dialog appears.
5. Click **Yes** to continue the uninstallation, click **No** to cancel.

   A dialog box shows the uninstallation progress. The box closes automatically when the uninstallation is complete.

6.2. Mac OS X

This section describes the Mac OS X uninstall procedure.

To uninstall DiType on a Mac OS X platform:

1. Run the `uninstall-ditype.sh` script from the command line.
2. Confirm that you want to remove **/usr/local/bin/ditype**, **/Library/StartupItems/DiType**, and **/Library/Frameworks/DiType.framework**.

   DiType is uninstalled.

6.3. Unix/Linux

To uninstall DiType from Unix/Linux platforms, delete the DiType installation directory.

Chapter 7. Integrating DiType

DiType can be used in many diverse environments. For this reason, it provides a number of ways to integrate with other applications and systems. The internal functionality of DiType and its integration methods are described in the following sections.

7.1. DiType Processing Flow

The logical flow of processing a document is:

- Transform - The transformer does one of two things. Either it determines the document type and applies the necessary transformations to convert it to XSL-FO. Or it applies the stylesheet referred to in the xml-stylesheet processing instructions.
- Preprocess - The preprocessor converts the XSL into FOSI, an internal format that is suitable for the formatting process.
- Render - The document is rendered to XEPOUT and passed to the back ends to generate the final output (PostScript, PDF or SVG).

The logical flow of document processing is shown in the following figure:

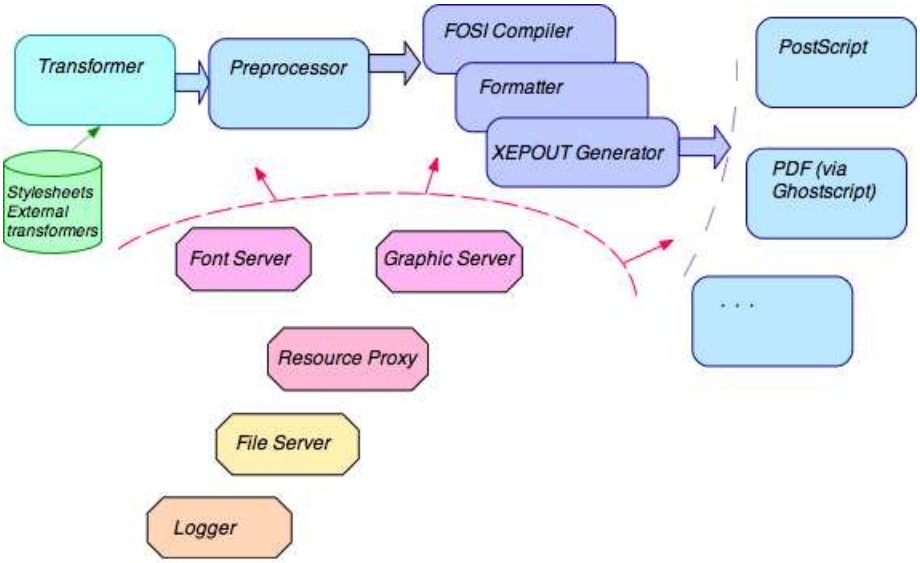

Figure 7.1. Document Processing

7.2. Universal Plugin Connector

Many interactive applications use XEP DiType uniformly by using the following procedure.

1. Initiate a session.

2. Set formatting options.

3. Send documents to DiType for formatting.

4. Retrieve formatting log and display it to the user while the formatting continues to run.

A Universal Plugin Connector (UPC) is available so that the same algorithms need not be re-implemented in each application. UPC provides the calls via XML-RPC, the protocol used in AJAX and inside XEP-DITYPE core. It is a robust way to interconnect pieces of a program using diverse technologies and even running on different computers. Applications connect to UPC and use it as a proxy to format documents and receive formatting logs. An application becomes an XML-RPC client, a simple role supported by well-developed libraries in most programming languages and environments.

7.3. API and Connectors

DiType provides a rendering API and connectors for some popular programming languages. APIs for Java and C# provide new interfaces and legacy versions that are compatible with the current interfaces. The Java and C# APIs and Connectors are described in the following sections.

7.3.1. Java API and Connectors

DiType provides a number of Java APIs that allow you to write your own code in Java to process XSL-FO and XML documents in multiple environments. The Java APIs and Connectors include a servlet component and an Ant connector.

Servlet Component

The servlet component provides the ability to use the DiType formatter for implementing web-applications. With the servlet component, it is possible to do the following:

- Design electronic libraries where all the books are stored on the server in XML/XSL format. Each e-library user has the opportunity to download any book in either PDF or PostScript format.
- Design a web service where each user can upload an XML/XSL-FO document to the server, or specify the URL of the document, and download the PDF or PostScript output.

Ant Connector

Apache Ant is a Java-based build tool used for automating software build processes. It is similar to **make**, but is written in the Java language and is extended using Java classes.

7.3.2. C# API and Connectors

DiType provides a C# API, which is a .Net based library that allows you to write your own code in the .Net environment to process XSL-FO and XML documents. The .Net API includes the following:

- `RenderX.DiType.dll`: The .Net assembly. The assembly is strongly named and can be installed to the `%SYSTEMROOT%\assembly` directory.
- DiType .Net API source code.
- DiType .Net API documentation.
- Examples directory containing sources and binaries of examples written in C# and VB.Net.
- `build.proj`: An MSBuild project file to build from the sources.

Appendix A. XSL-FO Conformance

A.1. XSL-FO Support

This appendix describes the implementation of XSL Formatting Objects in DiType — an XSL Engine for PDF developed by RenderX, Inc, version 1.9. It lists all supported formatting objects and their properties, provides information about fallbacks for unsupported objects, and discusses details of XSL spec interpretation adopted in the engine.

Note: DiType implements **Extensible Stylesheet Language version 1.1** as specified in the *XSL 1.1 Recommendation of December 5, 2006*.

A.1.1. Formatting Objects Supported by DiType

§	Object Name	Supported
6.4.2	`<fo:root>`	Yes
6.4.3	`<fo:declarations>`	No
6.4.4	`<fo:color-profile>`	No
6.4.5	`<fo:page-sequence>`	Yes
6.4.6	`<fo:page-sequence-wrapper>`	Yes
6.4.7	`<fo:layout-master-set>`	Yes
6.4.8	`<fo:page-sequence-master>`	Yes
6.4.9	`<fo:single-page-master-reference>`	Yes
6.4.10	`<fo:repeatable-page-master-reference>`	Yes
6.4.11	`<fo:repeatable-page-master-alternatives>`	Yes
6.4.12	`<fo:conditional-page-master-reference>`	Yes
6.4.13	`<fo:simple-page-master>`	Yes
6.4.14	`<fo:region-body>`	Yes
6.4.15	`<fo:region-before>`	Yes
6.4.16	`<fo:region-after>`	Yes
6.4.17	`<fo:region-start>`	Yes
6.4.18	`<fo:region-end>`	Yes

§	Object Name	Supported
6.4.19	<fo:flow>	Yes
6.4.20	<fo:static-content>	Yes
6.4.21	<fo:title>	No
6.4.22	<fo:flow-map>	No
6.4.23	<fo:flow-assignment>	No
6.4.24	<fo:flow-source-list>	No
6.4.25	<fo:flow-name-specifier>	No
6.4.26	<fo:flow-target-list>	No
6.4.21	<fo:region-name-speificier>	No
6.5.2	<fo:block>	Yes
6.5.3	<fo:block-container>	Yes
6.6.2	<fo:bidi-override>	Yes
6.6.3	<fo:character>	Yes
6.6.4	<fo:initial-property-set>	Yes
6.6.5	<fo:external-graphic>	Yes
6.6.6	<fo:instream-foreign-object>	Yes[1]
6.6.7	<fo:inline>	Yes
6.6.8	<fo:inline-container>	No[2]
6.6.9	<fo:leader>	Yes[3]
6.6.10	<fo:page-number>	Yes
6.6.11	<fo:page-number-citation>	Yes
6.6.12	<fo:page-number-citation-last>	Yes
6.6.13	<fo:folio-prefix>	Yes
6.6.14	<fo:folio-suffix>	Yes

1 <fo:instream-foreign-object> can host SVG graphics.

2 All content is placed inline.

3 In this version, only plain text can be put inside leaders with leader-pattern="use-content".

§	Object Name	Supported
6.6.15	<fo:scaling-value-citation>	Yes
6.7.2	<fo:table-and-caption>	Yes
6.7.3	<fo:table>	Yes
6.7.4	<fo:table-column>	Yes
6.7.5	<fo:table-caption>	Yes
6.7.6	<fo:table-header>	Yes
6.7.7	<fo:table-footer>	Yes
6.7.8	<fo:table-body>	Yes
6.7.9	<fo:table-row>	Yes
6.7.10	<fo:table-cell>	Yes
6.8.2	<fo:list-block>	Yes
6.8.3	<fo:list-item>	Yes
6.8.4	<fo:list-item-body>	Yes
6.8.5	<fo:list-item-label>	Yes
6.9.2	<fo:basic-link>	Yes
6.9.3	<fo:multi-switch>	-
6.9.4	<fo:multi-case>	-
6.9.5	<fo:multi-toggle>	-
6.9.6	<fo:multi-properties>	-
6.9.7	<fo:multi-property-set>	-
6.12.2	<fo:float>	Yes[4]
6.12.3	<fo:footnote>	Yes
6.12.4	<fo:footnote-body>	Yes
6.13.4	<fo:wrapper>	Yes
6.13.5	<fo:marker>	Yes[5]

4 Top-floated (`float="before"`) area is drawn on top of the **following** page.

5 In the current version, markers cannot be specified as children of <fo:wrapper>.

§	Object Name	Supported
6.13.6	<fo:retrieve-marker>	Yes
6.13.7	<fo:retrieve-table-marker>	Yes

A.1.2. Formatting Properties Supported by DiType

§	Property Name	Implemented
7.4.1	source-document	No
7.4.2	role	No
7.5.1	absolute-position	Yes[6]
7.5.2	top	Yes
7.5.3	right	Yes
7.5.4	bottom	Yes
7.5.5	left	Yes
7.6.1	azimuth	-
7.6.2	cue-after	-
7.6.3	cue-before	-
7.6.4	elevation	-
7.6.5	pause-after	-
7.6.6	pause-before	-
7.6.7	pitch	-
7.6.8	pitch-range	-
7.6.9	play-during	-
7.6.10	richness	-
7.6.11	speak	-
7.6.12	speak-header	-
7.6.13	speak-numeral	-
7.6.14	speak-punctuation	-

6 absolute-position="*fixed*" works on <fo:block-container> only.

§	Property Name	Implemented
7.6.15	speech-rate	-
7.6.16	stress	-
7.6.17	voice-family	-
7.6.18	volume	-
7.7.1	background-attachment	Yes
7.7.2	background-color	Yes
7.7.3	background-image	Yes
7.7.4	background-repeat	Yes
7.7.5	background-position-horizontal	Yes[7]
7.7.6	background-position-vertical	Yes[7]
7.7.7	border-before-color	Yes
7.7.8	border-before-style	Yes
7.7.9	border-before-width	Yes
7.7.10	border-after-color	Yes
7.7.11	border-after-style	Yes
7.7.12	border-after-width	Yes
7.7.13	border-start-color	Yes
7.7.14	border-start-style	Yes
7.7.15	border-start-width	Yes
7.7.16	border-end-color	Yes
7.7.17	border-end-style	Yes
7.7.18	border-end-width	Yes
7.7.19	border-top-color	Yes
7.7.20	border-top-style	Yes
7.7.21	border-top-width	Yes
7.7.22	border-bottom-color	Yes

[7] When the background image is repeated along an axis, its offset on this axis is ignored.

§	Property Name	Implemented
7.7.23	border-bottom-style	Yes
7.7.24	border-bottom-width	Yes
7.7.25	border-left-color	Yes
7.7.26	border-left-style	Yes
7.7.27	border-left-width	Yes
7.7.28	border-right-color	Yes
7.7.29	border-right-style	Yes
7.7.30	border-right-width	Yes
7.7.31	padding-before	Yes
7.7.32	padding-after	Yes
7.7.33	padding-start	Yes
7.7.34	padding-end	Yes
7.7.35	padding-top	Yes
7.7.36	padding-bottom	Yes
7.7.37	padding-left	Yes
7.7.38	padding-right	Yes
7.8.2	font-family	Yes
7.8.3	font-selection-strategy	Yes
7.8.4	font-size	Yes
7.8.5	font-stretch	Yes
7.8.6	font-size-adjust	Yes
7.8.7	font-style	Yes
7.8.8	font-variant	No
7.8.9	font-weight	Yes
7.9.1	country	No
7.9.2	language	Yes
7.9.3	script	No

§	Property Name	Implemented
7.9.4	hyphenate	Yes
7.9.5	hyphenation-character	Yes
7.9.6	hyphenation-push-character-count	Yes
7.9.7	hyphenation-remain-character-count	Yes
7.10.1	margin-top	Yes
7.10.2	margin-bottom	Yes
7.10.3	margin-left	Yes
7.10.4	margin-right	Yes
7.10.5	space-before	Yes
7.10.6	space-after	Yes
7.10.7	start-indent	Yes
7.10.8	end-indent	Yes
7.11.1	space-end	Yes
7.11.2	space-start	Yes
7.12.1	relative-position	No
7.13.1	alignment-adjust	Yes
7.13.2	alignment-baseline	Yes
7.13.3	baseline-shift	Yes
7.13.4	display-align	Yes
7.13.5	dominant-baseline	Yes
7.13.6	relative-align	Yes[8]
7.14.1	block-progression-dimension	Yes
7.14.2	content-height	Yes
7.14.3	content-width	Yes
7.14.4	height	Yes
7.14.5	inline-progression-dimension	Yes

[8] Supported on <fo:list-item>. On <fo:table-cell> elements, falls back to relative-align="*before*".

§	Property Name	Implemented
7.14.6	max-height	No[9]
7.14.7	max-width	No[10]
7.14.8	min-height	No[11]
7.14.9	min-width	No[12]
7.14.10	scaling	Yes
7.14.11	scaling-method	No
7.14.12	width	Yes
7.15.1	hyphenation-keep	No
7.15.2	hyphenation-ladder-count	No
7.15.3	last-line-end-indent	Yes
7.15.4	line-height	Yes
7.15.5	line-height-shift-adjustment	Yes
7.15.6	line-stacking-strategy	Yes
7.15.7	linefeed-treatment	Yes[13]
7.15.8	white-space-treatment	Yes
7.15.9	text-align	Yes[14]
7.15.10	text-align-last	Yes
7.15.11	text-indent	Yes
7.15.12	white-space-collapse	Yes[15]
7.15.13	wrap-option	Yes
7.16.1	character	Yes
7.16.2	letter-spacing	Yes

9 Maps to `height`.

10 Maps to `width`.

11 Maps to `height`.

12 Maps to `width`.

13 Value *"treat-as-zero-width-space"* for `linefeed-treatment` is not implemented. This property does not work on inlines.

14 `<string>` values for `text-align` are not implemented.

15 This property does not work on inlines.

§	Property Name	Implemented
7.16.3	suppress-at-line-break	No
7.16.4	text-decoration	Yes
7.16.5	text-shadow	Yes[16]
7.16.6	text-transform	Yes
7.16.7	treat-as-word-space	No
7.16.8	word-spacing	Yes
7.17.1	color	Yes
7.17.2	color-profile-name	No
7.17.3	rendering-intent	No
7.18.1	clear	Yes
7.18.2	float	Yes
7.18.3	intrusion-displace	Yes[17]
7.19.1	break-after	Yes
7.19.2	break-before	Yes
7.19.3	keep-together	Yes[18]
7.19.4	keep-with-next	Yes[18]
7.19.5	keep-with-previous	Yes[18]
7.19.6	orphans	Yes
7.19.7	widows	Yes
7.20.1	clip	No
7.20.2	overflow	Yes[19]
7.20.3	reference-orientation	Yes

16 Blurred shadows are not supported; blur radius is ignored.

17 *"indent"* value is not implemented.

18 `.within-page` component is treated as `.within-column`. In tables, `keep-with-previous`/`keep-with-next` traits ignore table headers and footers: e.g. `keep-with-previous` condition specified on a row will keep it with the previous one regardless of the intervening header. If specified on the first row of the first `<fo:table-body>` in a table, `keep-with-previous` will attach the whole table to the preceding block-level element.

19 Supported on side floats and absolutely positioned and rotated block-containers with fixed dimensions. When *"error-if-overflow"* is specified, a warning is issued on overflow, and the element is discarded in the same way as for *"hidden"* value.

§	Property Name	Implemented
7.20.4	span	Yes
7.21.1	leader-alignment	No
7.21.2	leader-pattern	Yes
7.21.3	leader-pattern-width	Yes
7.21.4	leader-length	Yes
7.21.5	rule-style	Yes
7.21.6	rule-thickness	Yes
7.22.1	active-state	-
7.22.2	auto-restore	-
7.22.3	case-name	-
7.22.4	case-title	-
7.22.5	destination-placement-offset	No
7.22.6	external-destination	Yes[20]
7.22.7	indicate-destination	No
7.22.8	internal-destination	Yes
7.22.9	show-destination	Yes[21]
7.22.10	starting-state	-
7.22.11	switch-to	-
7.22.12	target-presentation-context	-
7.22.13	target-processing-context	-
7.22.14	target-stylesheet	-
7.23.1	marker-class-name	Yes
7.23.2	retrieve-class-name	Yes
7.23.3	retrieve-position	Yes

[20] In PDF and PostScript generators, URLs starting with explicit "file:" protocol specification are rendered as PDF-to-PDF links ("remote go-to actions"). All other links are treated as Internet URIs, and open in a browser.

[21] show-destination is honored for creation of links between PDF documents ("remote go-to actions") in PDF and PostScript generators. In other cases, the attribute is not applicable.

§	Property Name	Implemented
7.23.4	retrieve-boundary	Yes
7.24.1	format	Yes
7.24.2	grouping-separator	No
7.24.3	grouping-size	No
7.24.4	letter-value	No
7.25.1	blank-or-not-blank	Yes
7.25.2	column-count	Yes
7.25.3	column-gap	Yes
7.25.4	extent	Yes
7.25.5	flow-name	Yes
7.25.6	force-page-count	Yes
7.25.7	initial-page-number	Yes
7.25.8	master-name	Yes
7.25.9	master-reference	Yes
7.25.10	maximum-repeats	Yes
7.25.11	media-usage	No
7.25.12	odd-or-even	Yes
7.25.13	page-height	Yes
7.25.14	page-position	Yes
7.25.15	page-width	Yes
7.25.16	precedence	Yes
7.25.17	region-name	Yes
7.26.1	border-after-precedence	Yes
7.26.2	border-before-precedence	Yes
7.26.3	border-collapse	Yes
7.26.4	border-end-precedence	Yes
7.26.5	border-separation	Yes

§	Property Name	Implemented
7.26.6	border-start-precedence	Yes
7.26.7	caption-side	Yes[22]
7.26.8	column-number	Yes
7.26.9	column-width	Yes
7.26.10	empty-cells	No[23]
7.26.11	ends-row	Yes
7.26.12	number-columns-repeated	Yes
7.26.13	number-columns-spanned	Yes
7.26.14	number-rows-spanned	Yes
7.26.15	starts-row	Yes
7.26.16	table-layout	Yes
7.26.17	table-omit-footer-at-break	Yes
7.26.18	table-omit-header-at-break	Yes
7.27.1	direction	Yes
7.27.2	glyph-orientation-horizontal	No
7.27.3	glyph-orientation-vertical	No
7.27.4	text-altitude	Yes
7.27.5	text-depth	Yes
7.27.6	unicode-bidi	Yes[24]
7.27.7	writing-mode	Yes[25]
7.28.1	content-type	Yes
7.28.2	id	Yes

22 Only *"before"* and *"after"* values are implemented: caption-side=*"start"* falls back to *"before,"* and *caption-side="end"* falls back to *"after."*

23 In the current implementation, all cells present in the source document are shown regardless of whether their content is empty; cells not present in the source are not visible at all.

24 Bidi implementation differs from Unicode Bidi algorithm: any markup element opens a new level of embedding. Consequently, unicode-bidi=*"normal"* is not supported (treated as *"embed"*); see detailed discussion below.

25 Only *"lr-tb"* and *"rl-tb"* values are supported. All other values are treated as *"lr-tb."*

§	Property Name	Implemented
7.28.3	provisional-label-separation	Yes
7.28.4	provisional-distance-between-starts	Yes
7.28.5	ref-id	Yes
7.28.6	score-spaces	No
7.28.7	src	Yes
7.28.8	visibility	No
7.28.9	z-index	Yes
7.29.1	background	Yes
7.29.2	background-position	Yes
7.29.3	border	Yes
7.29.4	border-bottom	Yes
7.29.5	border-color	Yes
7.29.6	border-left	Yes
7.29.7	border-right	Yes
7.29.8	border-style	Yes
7.29.9	border-spacing	Yes
7.29.10	border-top	Yes
7.29.11	border-width	Yes
7.29.12	cue	-
7.29.13	font	Yes
7.29.14	margin	Yes
7.29.15	padding	Yes
7.29.16	page-break-after	Yes
7.29.17	page-break-before	Yes
7.29.18	page-break-inside	Yes
7.29.19	pause	-
7.29.20	position	Yes

§	Property Name	Implemented
7.29.21	size	Yes
7.29.22	vertical-align	Yes
7.29.23	white-space	Yes
7.29.24	xml:lang	No

A.1.3. Notes on Formatting Objects Implementation

<fo:block>

According to the XSL Specification, an empty block that has a non-null padding and/or border should be visible. DiType suppresses all blocks that have no visible contents regardless of their border or padding attributes.

<fo:bidi-override>

In the current implementation of bidi algorithm, any markup element opens a new level of embedding. Consequently, `unicode-bidi="normal"` is not supported: `<fo:bidi-override>` behaves as if `unicode-bidi="embed"` were specified.

<fo:inline-container>

Unsupported; contents are placed inline.

<fo:multi-switch>
<fo:multi-case>
<fo:multi-toggle>
<fo:multi-properties>
<fo:multi-property-set>

Unsupported; contents are ignored. These elements deal with interactivity. Since PDF and PostScript are intrinsically static formats, none of them are applicable.

<fo:float>

The before-float appears at the top of the **next** page.

<fo:table-caption>

Only *"before"* and *"after"* captions are implemented. Side captions are treated as follows: `caption-side="start"` falls back to *"before"*, and `caption-side="end"` falls back to *"after."*

<fo:table-column>

In the collapsed border model, only border-start and border-end are supported on `<fo:table-column>` elements.

<fo:table-row>

In the collapsed border model, only border-before and border-after are supported on <fo:table-row> elements.

<fo:table-cell>

If a cell spans multiple rows in a table with a collapsed border model, its border-after is taken from the row where the cell begins.

<fo:leader>

In this version, leaders with leader-pattern="use-content" can contain only plain text inside; all formatting is lost.

<fo:marker>

This version cannot process markers specified as children of an <fo:wrapper>.

A.1.4. Supported Expressions

DiType implements a subset of XSL algebraic expressions. The following operators and functions are recognized:

- Arithmetical operators: +, -, *, div, mod
- floor()
- ceiling()
- round()
- abs()
- max()
- min()
- rgb()
- rgb-icc() (supported partially — see notes below)
- from-nearest-specified-value()
- from-parent()
- from-table-column()
- inherited-property-value()
- proportional-column-width()

- `body-start()` (stand-alone use only, cannot be an operand in expressions)
- `label-end()` (stand-alone use only, cannot be an operand in expressions)

Function `rgb-icc()` recognizes four predefined color profile names: `#Grayscale`, `#CMYK`, `#SpotColor`, and `#Registration` (see details below). For any other value of the fourth parameter, the function returns the fallback RGB color. ICC profiles are not supported.

Support for expressions is subject to the following limitations:

- For compound expressions, the result of evaluation of all intermediate subexpressions must be a valid XSL type. For example, the expression `(2in * 2in) div 1in` is not supported because its first subexpression yields dimensionality of square inches, which is not a valid XSL unit; while `2in * (2in div 1in)` works.
- Expressions that require knowledge of the layout to evaluate (e.g. block widths expressed in percentages) can be used only as stand-alone expressions, not as parts of a bigger expression, and cannot be referenced by property-value functions. The same limitation applies to `body-start()` and `label-end()` functions.
- Property value functions (`from-nearest-specified-value()`, `from-parent()`, `from-table-column()`, `inherited-property-value()`) cannot be used in shorthand expressions, and cannot take shorthand property names as their arguments.
- Property value functions that take `start-indent`/`end-indent` as arguments may not work correctly if the block with indents is placed into another block that has CSS-style `margin-*` attributes. For safety, use either expressions with indents or CSS margins, but not both; mixing these two coding styles in the same stylesheet may yield unpredictable results.

A.1.5. Color Specifiers

DiType can produce PDF, PostScript and SVG output using the following color types:

1. **Grayscale**. The following specifiers produce grayscale color output:
   - Predefined HTML and SVG names that correspond to RGB values with *R* = *G* = *B*: `white`, `black`, `silver`, `gray`, `grey`, `lightgray`, `lightgrey`, `darkgray`, `darkgrey`, `dimgray`, `dimgrey`, `whitesmoke`, `gainsboro`.
   - HTML-style RGB values with *R* = *G* = *B*: *#555*, *#9D9D9D*, etc.
   - `rgb-icc()` function with built-in `#Grayscale` pseudo profile. Gray tone intensity is specified as a real value in the range `0.0`–`1.0`, the 5th argument to the function. Example:

     ```
     rgb-icc (128, 128, 128, #Grayscale, 0.5)
     ```

2. **RGB**. The following specifiers produce RGB color output:

   - HTML and SVG predefined names, and RGB specifiers that are not mentioned above.
   - `rgb()` function. Values of color components are specified as real values in the range `0.0`–`255.0`. Example:

     ```
     rgb (127.5, 39.86, 255)
     ```

3. **CMYK**. The following specifier produce CMYK color output:

   - `rgb-icc()` function with built-in `#CMYK` pseudo profile. Ink values are specified as real values in the range `0.0`–`1.0`, arguments from 5th to 8th; order of inks is *cyan–magenta–yellow–black*. Example:

     ```
     rgb-icc (255, 255, 0, #CMYK, 0, 0, 1, 0)
     ```

4. **Spot colors**. The following specifiers produce spot color output:

   - `rgb-icc()` function with built-in `#SpotColor` pseudo profile. The 5th argument is the colorant name, specified as a string; use quotes if the name contains spaces. The 6th argument is the tint value, specified as a real number in the range `0.0`–`1.0`. These mandatory attributes may be followed by an optional specification of the alternate color for the colorant, in either CMYK or grayscale color space: 7th argument is the color space name (either `#CMYK` or `#Grayscale`), and the rest are component intensities (1 for grayscale, 4 for CMYK).

     Note: The alternate color specifies an equivalent representation **for the full colorant intensity**. Occurrences of the same spot color with different tints should have the same alternate color specifier.

     If the alternate color is not specified, DiType looks it up in SpotColor matching table (path to the table is defined by the `<SPOT_COLOR_TRANSLATION_TABLE>` option); if not found there, black color in grayscale color space is used.

     Examples:

     ```
     rgb-icc(255,255,0, #SpotColor,'PANTONE Orange 021 C',0.33)
     rgb-icc(255,255,0, #SpotColor,'PANTONE 169 M',0.5, #CMYK,0,0.2,0.2,0)
     rgb-icc(255,255,0, #SpotColor,MyColor,0.33, #Grayscale,0.5)
     ```

5. **Registration color**. The following specifier produces registration (all-colorants) color output:

   - `rgb-icc()` function with built-in `#Registration` pseudo profile. Tint intensity is specified as a real value in the range `0.0`–`1.0`, the 5th argument to the function. Example:

     ```
     rgb-icc (128, 128, 128, #Registration, 0.5)
     ```

A.1.6. XSL 1.1 Support

DiType implements several new features of XSL 1.1. The following is an informal table describing features and current level of support and limitations.

Feature	Comments
Bookmarks	Partial (no styling for bookmark titles). The rx:outline extension is supported via a stylesheet.
Change bars	Partial (currently, the elements are allowed only in %block; or %inline; contexts. Forking on out-of-flow elements is not implemented). The rx:change-bar-* extension is supported via a stylesheet.
Flow maps	Fully implemented.
Folio prefix and suffix	Fully implemented.
Graphic scaling	Partial (fo:scaling-value-citation is not implemented).
@id on root, static-content, flow, footnote-body, footnote, float	Partial (except on fo:root and fo:static-content).
Indexing	Partial (except @index-class). The rx: extensions for indexing is supported via a stylesheet.
inside/outside values for @clear	Fully supported.
block-container/@overflow="repeat"	Not supported.
block-container/@overflow="error-if-overflow"	Fully supported.
fo:page-number-citation-last	Partial (except @page-citation-strategy). The rx:page-number-citation-last extension is supported via a stylesheet.
page-position="only"	Fully supported.
page-sequence-wrapper	Fully supported.
Table markers	Not implemented (future).

A.1.7. Extensions to the XSL 1.0 Recommendation

DiType implements several extensions to the XSL Specification, placed into a separate namespace: `xmlns:rx="http://www.renderx.com/XSL/Extensions"`. They add support for useful functionality that cannot be expressed by XSL Formatting Objects.

Document Information

This extension permits passing a set of name/value pairs to the generator of the output format. A typical application is setting PDF document info fields (`'Author'` and `'Title'`). Implementation uses two extension elements: `<rx:meta-info>` and `<rx:meta-field>`.

`<rx:meta-info>`

This element is merely a container for one or more `<rx:meta-field>` elements. It should be the first child of `<fo:root>`.

`<rx:meta-field>`

This element specifies a single name/value pair. It has two mandatory attributes: `name` and `value`. Current implementation of the PDF and PostScript generators recognize four possible values for `name`:

- `name="author"`- fills the `'Author'` field in the resulting PDF file with a string specified by the `value` property.
- `name="creator"` - fills the `'Creator'` field.
- `name="title"` - fills the `'Title'` field.
- `name="subject"` - fills the `'Subject'` field.
- `name="keywords"` - fills the `'Keywords'` field.

All other values for `name` are ignored. The `'Producer'` field in the PDF file is set to *`"DiType <version>"`*; there is no way to control it from the source file.

In the PostScript generator module, the document info fields are added using the `pdfmark` operator. The respective fields are filled when PostScript is converted to PDF using *Adobe Acrobat Distiller* or *GhostScript*.

Document Outline (Bookmarks)

Document Outline is natively implemented in terms of XSL 1.1 bookmarks. For compatibility reasons the rx:outline extension is also supported. The rest of this section describes the extension.

This extension provides the following three elements:

- `<rx:outline>` - The top-level element of the document outline tree. It should be located before any `<fo:page-sequence>` elements, and after the `<fo:layout-master-set>` and the `<fo:declarations>` elements (if present). It contains one or more `<rx:bookmark>` elements.
- `<rx:bookmark>` - This element contains information about a single bookmark. It contains a mandatory `<rx:bookmark-label>` element as its first child, and zero or more nested `<rx:bookmark>` elements that describe nested bookmarks. Bookmark destination is expressed either by `internal-destination` property (for internal navigation), or by `external-destination` (for extra-document links). The initial presentation of the children bookmarks is controlled by `collapse-subtree` attribute. Values are either *"true"* (collapse children) or *"false"* (expand children).
- `<rx:bookmark-label>` - This element contains text of a bookmark label. It must be the first child of its parent `<fo:bookmark>`. Content of this element should be plain text.

Indexes

Indexes are natively implemented in terms of XSL 1.1 Indexes. For compatibility reasons the corresponding rx: extension is also supported. The rest of this section describes the extension.

Building page number lists for back-of-the-book indexes is a common task. It is relatively easy to collect a list of references to index terms in the text. However, to turn them into a real index entry, you should exclude repeated page numbers and merge adjacent numbers into ranges. Neither of these two operations can be done in XSL 1.0. Therefore, DiType supports an extension for this purpose.

The task of building an index can be split in two subtasks:

- Mark up occurrences of index terms in the main text.
- Specify composition and formatting of page number lists in the index.

Index Term Markup

In order to mark up occurences of the index terms in the text, DiType introduces a special extension attribute: `rx:key`. It can be specified on any element that can take an `id` attribute; unlike the latter, it need not be unique across the document. Its value is used as a key to select elements for the page number list. For example, an index term to the word "rendering" might look like this:

```
The process of converting XSL-FO to a printable format
is called <fo:inline rx:key="key.render">rendering.</fo:inline>
```

There is also a mechanism to specify an explicit range, not distinct elements. Two extension elements serve this purpose:

<rx:begin-index-range>

Starts a range. It takes two attributes, both required:

id

A unique identifier used to define the limits of the range.

rx:key

An Index key used to select the range into a page number list.

<rx:end-index-range>

Ends a range. It takes one attribute, required:

ref-id

A reference to the `id` attribute of the `<rx:begin-index-range>` that started the range.

These two elements always form a pair. These elements may be located anywhere inside `<fo:flow>`; there are no constraints on their nesting with respect to other elements.

Index Entries

In the index, the actual page reference is created by another extension element, `<rx:page-index>`. It picks elements from the text by their `rx:key` properties, and produces a sorted list of their page numbers, eliminating duplicates.

`<rx:page-index>` should contain one or more `<rx:index-item>` elements as children. Each `<rx:index-item>` has a required `ref-key` attribute, and selects elements that have an `rx:key` attribute with the same value.

A distinct element bearing the appropriate `rx:key` value is represented as follows:

- If it fits completely onto one page, it is represented as a single page number.
- If it spans multiple pages, its entry is formatted as a range from the first to the last of the spanned pages.

A range (created by a `<rx:begin-index-range>` and `<rx:end-index-range>` element pair) is represented as a range from the page where `<rx:begin-index-range>` is located to the page of its matching `<rx:end-index-range>`.

A basic entry in an index looks like this:

```
<fo:inline rx:key="key.elephant">Elephants</fo:inline> live in Africa. …
<fo:inline rx:key="key.elephant">African elephants</fo:inline> have big ears …
…
<fo:block text-align="center" font="bold 16pt Futura">INDEX</fo:block>
<fo:block>
    Elephants <rx:page-index>
                  <rx:index-item ref-key="key.elephant"/>
              </rx:page-index>
</fo:block>
```

There are other attributes of `<rx:index-item>` to control the formatting of the index entry:

`range-separator`

Specifies the string used to separate page numbers that form a continuous range. Default is en dash: `"–" (U+2013)`.

`merge-subsequent-page-numbers`

Specifies whether sequences of adjacent page numbers should be merged into ranges. Default is `"false."`

`link-back`

Specifies whether page numbers should be made into hyperlinks to the corresponding page. Default is `"false."`

Besides that, `<rx:index-item>` can take additional inline attributes, applied to each page number generated from this element. This allows for different presentation styles across the list, e.g., to make references to primary definitions bold.

Flow Sections

Flow sections permit splitting the flow into subflows, with different column counts in each subflow. The following element creates flow sections:

`<rx:flow-section>`

This element must be a direct child of `<fo:flow>`. It can be mixed with other block-level elements. It takes two attributes, both required:

`column-count`

The number of columns for the subflow.

`columngap`

The space between the columns.

Last Page Number Reference

Last Page Number Reference is natively implemented in terms of XSL 1.1 fo:page-number-citation-last. For compatibility reasons the rx:page-number-citation-last extension is also supported. The rest of this section describes the extension.

This extension element retrieves the number of the last page occupied by a particular element. Its syntax and semantics are similar to `fo:page-number-citation`.

`<fo:page-number-citation-last>`

The only required attribute, `ref-id`, specifies the `id` of the element whose last page number you want to retrieve. In particular, by referencing the `id` of the `<fo:root>` element, it is possible to retrieve the number of the last page in the document.

For compatibility, this element is recognized in rx: namespace as well.

Change Bars

Change Bars are natively implemented in terms of XSL 1.1 Change Bars, currently with certain limitations. For compatibility, the corresponding rx: extension is also supported. The rest of this section describes the extension.

DiType has support for change regions, as described in XSL 1.1.

`<rx:change-bar-begin>`
`<rx:change-bar-end>`

These elements have exactly the same meaning and properties as listed in the Working Draft for elements `<fo:change-bar-begin>` and `<fo:change-bar-end>`, sections 6.3.12 and 6.3.13, respectively.

Background Image Scaling and Content Type

In XSL 1.0, there is no provision to scale/size a background image. XEP implements this functionality via the following extension properties:

`rx:background-content-height`
`rx:background-content-width`
`rx:background-scaling`
`rx:background-content-type`

These properties have the exact same semantics as `content-height`, `content-width`, `scaling`, and `content-type`, respectively. They apply to the image specified in `background-image` property (or inside `background` shorthand).

Initial Destination

This extension allows you to specify the destination to jump to when the document is first opened. It uses a single extension attribute, `rx:initial-destination` placed on `<fo:root>`. Its syntax is the same as the `internal-destination` attribute.

Base URI Definition: `xml:base`

DiType recognizes and processes `xml:base` attribute, defined in XML Base Recommendation. It permits you to set the base for resolving relative URIs (link targets, image locations, fonts, hyphenation patterns, etc.) for the whole document or a single subtree.

Note: The use of `xml:base` in XSL is not authorized by the XSL Specification; therefore, this option should be considered a proprietary extension to XSL.

Border and Padding on Regions

In the XSL Recommendation, border and padding properties are permitted on region elements (`<fo:region-body>`, `<fo:region-before>`, `<fo:region-after>`, `<fo:region-start>`, and `<fo:region-end>`). However, they may accept values of 0 **(sic!)**. In DiType, non-zero values of these properties result in a border around the respective region area, and its content rectangle is padded by the specified amount.

Kerning: `rx:kern`

An inheritable extension attribute `rx:kern` allows you to control which kerning pairs available in the font will be honored. The values are:

Value	Effect
auto	All kerning pairs defined by the font are used. This is the default.
none	Kerning is disabled.
<space-separated list of pairs>	Those kerning pairs defined by the font that are present in the list are used. Sample value: "AT AV".

Ligaturization: `rx:ligaturize`

An inheritable extension attribute `rx:ligaturize` allows you to control which character sequences defined in the Unicode and present in the font will be converted to corresponding ligatures. The values are:

Value	Effect
auto	All possible ligatures will be used.

Value	Effect
none	No sequence will be ligaturized. This is the default.
<space-separated list of character sequences>	Those sequences for which the font provides a ligature and if present in the list will be ligaturized. Sample value: "fi fl".

Note: If the input document already contains a ligature and this ligature is available in the font, it will come out as a ligature, no matter what the value of `rx:ligaturize`.

If the input document already contains a ligature but this ligature is not available in the font, it will be replaced with the corresponding sequence. If any character of this sequence is also not available in the font, the font-family for this character will be degraded in the regular order (i.e., another font in a multiple font family, or the default font). If degrading fails, a missing glyph will be drawn.

Hyphenation does not appear in the middle of a ligature. Other parts of a word may be properly hyphenated.

Appendix B. Linguistic Algorithms

B.1. Line-Breaking Algorithm

The following rules comprise DiType's line-breaking algorithm:

1. Line-break is permitted if one of the following conditions is fulfilled:
   - Line-break is forced by the explicit linefeed characters: `U+000A, U+000D, U+2028, and U+2029`. Note, however, that the default behavior of DiType is to perform **linefeed normalization**, which treats all linefeed characters like spaces. Therefore, the linefeed characters actually force a line-break only if the `linefeed-treatment` attribute is set to "preserve."
   - Line-break is permitted at space characters: `U+0009, U+0020, U+2000 - U+200B`, and `U+3000`.
2. Line-break is not allowed in the following cases, **unless** one of the conditions of rule 1 is fulfilled:
   - Immediately preceding or following non-breaking spaces (`U+00A0`) and non-breaking hyphens (`U+200C`).
   - Immediately preceding trailing punctuation characters, closing brackets and quotes, small Katakana and Hiragana characters, superscript characters, etc.
   - Immediately after opening brackets and quotes, Spanish leading punctuation, currency symbols, etc.
3. If the `hyphenate` attribute is set to "true" and all hyphenation conditions (`hyphenation-push-character-count, hyphenation-remain-character-count`, etc.) are satisfied, then line-break is permitted after a soft hyphen (`U+00AD`). A soft hyphen at the end of a line is replaced by the text specified in the `hyphenation-character` attribute; all other soft hyphens are suppressed.
4. Unless prohibited by the above rules, line-break is permitted before or after CJK ideographic, Katakana, Hiragana, and Hangul characters.
5. In all other cases, line-break is prohibited.

The algorithm will be refined in future versions of DiType, when more feedback about non-European scripting systems is received.

B.2. Hyphenation

DiType uses Unicode soft hyphen characters (`U+00AD`) to mark possible hyphenation points. These characters either can be contained in the source XSL-FO document (e.g. from an external hyphenation application), or can be added by DiType automatically before the source is passed to the formatter.

The hyphenator implements Liang's algorithm. DiType's distribution includes patterns for the following languages: English (American and British), French, German, Spanish, Russian, Polish, Italian, Finnish, Danish, Dutch, Croatian, Czech, Irish, Catalan, Hungarian, Interlingua, Basque, Greek, Latin, Galician, Slovenian, Swedish, Portuguese, Estonian, Icelandic, Norwegian, Turkish, Mongolian, Sanskrit, Bulgarian, Serbian. All patterns are borrowed from CTAN (the Comprehensive TeX Archive Network, http://www.ctan.org/), with some modifications for non-English patterns. More patterns can be added if necessary.

B.2.1. Hyphenation Patterns

The hyphenator uses T_EX format for hyphenation patterns. It recognizes the following sections in the pattern files:

- patterns (for hyphenation patterns)
- hyphenation (for exceptions)

Any other section in the pattern file is ignored. Hexadecimal escape codes (e.g. ^^ae) and control characters (^^A) are supported; they can be used to encode non-ANSI European characters. Additionally, DiType recognizes a set of \rm macros for accented characters: \^a is â (a with circumflex accent), \l is ł (Polish barred l), etc.

B.3. Support for Right-to-Left Writing Systems

DiType supports both left-to-right and right-to-left text. To define ordering of charactes within lines and stacking direction of lines into paragraphs, the `writing-mode` attribute is used. It can be specified on the `<fo:simple-page-master>`, `<fo:region-*>`, `<fo:table>`, `<fo:block-container>`, and `<fo:inline-container>` elements. Its primary values are:

- `"lr-tb"`: left-to-right, top-to-bottom. This is the default writing mode in XSL-FO; it is used by the majority of world languages, including English.
- `"rl-tb"`: right-to-left, top-to-bottom. This mode is used in Arabic writing system (adopted by many languages of the Middle East), Hebrew, and Syriac alphabets.
- `"tb-rl"`: top-to-bottom, right-to-left. This way of writing is widely used for Japanese, but also for Chinese and other languages of East Asia.

Note: DiType supports only horizontal writing modes: `"lr-tb"` and `"rl-tb"`.

The `writing-mode` attribute defines every aspect of the document organization: binding edge, column ordering in tables, text alignment in blocks, etc. It also sets the correspondence between relative directions (`before – after – start – end`) and absolutely oriented ones (`top – bottom – left – right`).

B.3.1. Bidirectionality

Bidirectionality is the interleaving of text which is to be displayed in both directions: for example, operating instructions are in Hebrew, but the name of the product appears in the middle of the instructions, in English. In simple situations, the renderer handles the bidirectionality on its own; there are, however, many situations where there may be an ambiguity as to the exact resolution desired. For these situations, XSL defines a special element, `<fo:bidi-override>` that enables altering the bidirectional behavior of the whole text or its parts. It has two properties:

`direction`

Sets the dominant direction for a span of text. Possible values are:

- `"ltr"` — from left to right
- `"rtl"` — from right to left

`unicode-bidi`

Specifies behavior of a text span with respect to the Unicode bidi algorithm. Possible values are:

- `"normal"` — order characters by Unicode bidi.
- `"embed"` — open a new level of embedding.
- `"bidi-override"` — ignore directionality of the text and arrange characters in the order specified by the `direction` property.

B.3.2. Glyph Shaping

DiType supports contextual selection of Arabic positional glyph variants, known as **glyph shaping**. Shaping proceeds as follows: each character that belongs to Arabic Unicode range `U+0600–U+06FF` is replaced by its counterpart in the Arabic Presentation Forms ranges `U+FB50–U+FDFF` and `U+FE70–U+FEFF`, in accordance with the Unicode rules for Arabic. Only basic changes are considered:

- Substitution of initial, final, and medial forms
- Insertion of lam-alef ligatures

Shaping occurs before font selection. For the algorithm to work, the following conditions must be met:

- Fonts chosen for Arabic text spans shall cover all positional variants for glyphs used (You can specify a list of fonts. Glyphs will be searched in all of them, following the usual rules for processing of multiple font families).
- Positional variants are accessible through their Unicode codepoints.

This is the case for most TrueType fonts that support Traditional Arabic; however, DiType does not work with Simplified Arabic fonts.

Appendix C. Supported Fonts

C.1. Supported Fonts

This appendix lists font types currently supported in DiType, and describes the details of their use. The overall structure of font configuration is described in the DiType User Guide; here, details specific to particular font formats are described.

C.1.1. PostScript Type 1 Fonts

To use a Type 1 font with DiType, it is necessary to obtain an AFM (Adobe Font Metrics) file for the font, and to specify the URL to it in the `afm` attribute of the `<font-data>` element. If the font is to be embedded into the resulting PDF or PostScript documents, a font outline file in PFA or PFB format is also needed; its location is specified in the respective attribute of the `<font-data>` — either `pfa` or `pfb`. (SVG output doesn't support font embedding)

Example: suppose we have a metrics file `mybar.afm` and an outline file `mybar.pfb`. Its descriptor in the configuration file should look like this:

```
<font embed="true" subset="true">
    <font-data afm="mybar.afm" pfb="mybar.pfb"/>
</font>
```

If your Type 1 font uses non-standard glyph names, you may need an additional step — custom glyph list registration. This is discussed in more detail in the next section.

PostScript Fonts and Unicode

Type 1 font support in DiType is based on direct mapping of Unicode characters to glyph names. Built-in character codes are not used in the formatting.

DiType follows Adobe's guidelines for mapping Unicode values to glyph names, as described in the following document: *Unicode and Glyph Names, version 2.3* (http://partners.adobe.com/public/developer/opentype/index_glyph.html). By default, *Adobe Glyph List, version 2.0* (hereinafter, AGL; http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt) is used to determine the Unicode positions for Type 1 glyphs; AGL is hard-coded inside DiType.

If a font includes only glyphs comprised in the AGL and all glyphs are named according to Adobe standards, you need no additional steps to use them in XEP. (This is normally the case with most Latin-based Type 1 fonts). However, some fonts cannot be covered by the AGL:

- Some fonts define glyphs outside the scope of AGL — exotic scripts, custom dingbats, etc.

- Some others give non-standard names to glyphs; e.g., Cyrillic or Armenian fonts from TeX.

With DiType, it is possible to use such fonts and to access characters from them by their regular Unicode values. All you need to do is to write an extension to the Adobe Glyph List, and register in the font descriptor: `glyph-list` attribute of a `<font-data>` element that contains a URL to the extension glyph list. Glyph lists are ascribed to fonts individually: different fonts in your system may use different glyph naming systems.

The syntax of a custom glyph list is as follows:

- Lines starting with `'#'` are comments.
- Empty lines are ignored.
- Each non-comment and non-empty line contains information about a single glyph.
- Within a line, records are separated by semicolons.
- The first record is the Unicode value — 4 hex digits.
- The second record is the glyph name as used in the AFM file.
- The rest of the line is treated as a comment.

Note: The syntax for the glyph list follows the structure of the previous version of AGL, *Adobe Glyph List 1.2* (http://www.renderx.com/glyphlist-old.txt). Unfortunately, the two versions of AGL are not compatible with each other.

Duplicate entries are allowed in glyph lists: you can assign different Unicode values to one and the same glyph, and have more than one glyph point to the same Unicode value.

In a custom glyph list, there is no need to cover all symbols present in the font: only non-standard mappings should be included. All glyphs not found in the glyph list are processed according to AGL 2.0 (hard-coded into the formatter).

Following is a schematic example of a custom glyph list:

```
# Sample Glyph List

0020;space
0021;exclam;EXCLAMATION MARK
…
…
…
```

A registration entry for a font with custom glyph mapping looks like this:

```
<font-data afm="mybar.afm"
           pfa="mybar.pfa"
           glyph-list="my.glyphs"/>
```

Standard Adobe Fonts

An important kind of Type1 fonts are Adobe standard font families: `Times`, `Helvetica`, `Courier`, `Symbol`, and `ZapfDingbats`. They are present in every PDF or PostScript installation, and do not require embedding. The default DiType configuration includes settings for them.

All symbols from these fonts are accessed by Unicode, including `Symbol` and `ZapfDingbats` fonts. For Symbol, mapping of Unicode to glyph names is contained in the *Adobe Glyph List, version 2.0* (http://partners.adobe.com/public/developer/en/opentype/glyphlist.txt). ForZapfDingbats, the mapping is taken from a separate document, also available at the Adobe technical support site: http://partners.adobe.com/public/developer/en/opentype/zapfdingbats.txt.

DiType samples include three files where all glyphs available from standard Adobe fonts are listed, with their Adobe glyph names and Unicode values:

- `adobe-standard.fo` - lists all glyphs from Roman Extended character set.
- `symbol.fo` - lists all glyphs from Symbol character set.
- `zapf-dingbats.fo` - lists all glyphs from ZapfDingbats character set.

C.1.2. TrueType Fonts

TrueType fonts are supported in DiType, with the following limitations:

- DiType can use only Unicode-enabled TrueType fonts, i.e., those with an internal `cmap` table for mapping glyph IDs to Unicode. Most TrueType fonts now satisfy this condition, but not all. A notable exception is `Wingdings` font, commonly found on Windows machines.

DiType supports only stand-alone TrueType fonts (normally stored in files with a `*.ttf` extension). Fonts in TrueType Collection files (they normally have a `*.ttc` extension) are not supported. To use a stand-alone TrueType font with DiType, a URL to its font file should be specified in a `ttf` attribute to the `<font-data>` element, as in the following example:

```
<font-data ttf="MYBAR.TTF"/>
```

Appendix D. Supported Graphic Formats

D.1. Supported Graphic Formats

D.1.1. Bitmap Graphics

DiType supports the following raster graphics formats:

- PNG
- JPEG
- GIF
- TIFF

Bitmap graphic that have no built-in resolution or dimension data, default to a resolution of 120 dpi (5 dots of a 600-dpi printer) as prescribed by the CSS2 Spec. This is always the case for GIF images, but may also occur for other image types. The XSL Recommendation suggests using 0.28 mm as a pixel size in such cases, which corresponds to 90-dpi resolution. A smaller pixel size gives better print results because the proportion between pixel size and page width is similar to that of a computer screen. With lower resolutions, often the large GIF/JPEG images fit onto a screen but not into the printable area on the page. For interoperability with other XSL-FO implementations, it is advisable to specify image size explicitly in the XSL-FO code.

PNG

DiType recognizes all types of PNG images described in the PNG specification. Currently, DiType translates PNG images into JPEG to reproduce in the PostScript and PDF output, so that transparency is not supported and image quality is affected. Handling of PNG images will be improved in future versions.

JPEG

Grayscale, RGB, and CMYK JPEGs are supported. Data stream is copied directly from the image file to the resultant PDF or PostScript, so there is no additional loss of quality.

For CMYK JPEGs, DiType analyzes the contents of the `APP14` marker. If the marker indicates that the image is created by Adobe, color polarity is inverted: 0 means "full colorant". Otherwise, standard CMYK conventions apply: 0 is treated as "no colorant".

GIF

DiType supports both interlaced and non-interlaced GIF images, and includes implementation of the LZW algorithm.

TIFF

DiType supports the following principal TIFF flavors:

- File organization - strip-based or tiled
- Color model - monochrome, grayscale, RGB, or CMYK
- Compression type - uncompressed, CCITT Fax (monochrome images only), PackBits, or LZW

TIFF images with separate color planes (`PlanarConfiguration=2`) or an associated alpha channel (`ExtraSamples=1`) are not supported.

D.1.2. Vector Graphics

XEP supports the following vector graphics formats:

- SVG
- EPS

Note: Embedding PDF illustration is currently not supported; instead, PostScript can be embedded in both PDF and PostScript outputs.

SVG

XEP supports a subset of Scalable Vector Graphics, version 1.1. SVG images can be either referenced as external files (in `src` and `background-image` attributes) or directly embedded into the XSL-FO flow through the `<fo:instream-foreign-object>` wrapper.

XEP implements the following SVG elements:

- structure elements - `<svg>`, `<g>`, `<defs>`, `<use>`, `<symbol>`, `<image>`
- styling - `<style>`
- shapes - `<rect>`, `<circle>`, `<ellipse>`, `<polygon>`, `<polyline>`, `<path>`
- basic clipping - `<clipPath>` (see limitations below)
- text - `<text>`, `<tspan>`, `<tref>`
- conditional processing - `<switch>`

The following SVG properties are supported:

- `baseline-shift`
- `clip-path` (see below for limitations on clipping support)
- `color`
- `fill`
- `fill-opacity`
- `fill-rule`
- `font`
- `font-family`
- `font-size`
- `font-stretch`
- `font-style`
- `font-weight`
- `letter-spacing`
- `marker`
- `marker-end`
- `marker-start`
- `marker-mid`
- `stroke`
- `stroke-width`
- `stroke-linecap`
- `stroke-linejoin`
- `stroke-miterlimit`
- `stroke-dasharray`
- `stroke-dashoffset`
- `stroke-opacity`

- `text-anchor`
- `transform`
- `visibility`
- `word-spacing`
- `xml:base`
- `xml:space`

Notes on SVG support in XEP:

1. Color treatment for SVG follows the same rules as for XSL-FO. In particular, `#CMYK`, `#Grayscale`, `#SpotColor`, and `#Registration` pseudo profile names can be used in the `icc-color()` function to produce CMYK, grayscale, spot, and registration colors, respectively.
2. For an SVG image to be processed by XEP, it must have an intrinsic size. If `height` or `width` are expressed in percents, a `viewBox` attribute must be present: the intrinsic size is determined by the `viewBox` attribute, assuming 1 user space unit = 1 pixel.
3. Animation-related elements and attributes are ignored. All objects are drawn at their specified static positions; no attempt is made to reconstruct the initial state of an animated picture.
4. The `clip-path` attribute is not supported on the elements inside the `<clipPath>` element or on the `<clipPath>` element itself.
5. Remote references to the `clipPath` and `marker` elements are unsupported: only the fragment identifier is used to retrieve them. (Remote links in `use` elements are supported.)
6. Character-by-character placement and rotation in text elements are not supported. If an array is used in the `x`, `y`, `dx`, `dy`, or `rotate` attribute of the `<text>` or `<tspan>` element, only the first number is considered.
7. Bidi reordering and Arabic glyph shaping do not work in SVG text.
8. The `xml:base` attribute works only when resolving relative URLs for external images via the `<image>` element. It is ignored in the `<use>`, `<tref>`, and similar elements.
9. XEP supports SVG styling via embedded CSS stylesheets (`<style>` element, and `style` and `class` attributes). CSS support is limited to Level 1: only ancestor, class, and ID selectors are recognized. Pseudo classes and pseudo elements are not supported.

EPS

EPS images are inserted intact into PostScript output. In DiType, the PDF output is generated from PostScript output by means of GhostScript, therefore EPS images are supported in both PostScript and PDF output.

Appendix E. DiType Intermediate Output Format Specification

E.1. DiType Intermediate Output Format Specification

This section describes the DiType intermediate output format — an XML based representation of the layout that is passed from the renderer to the final output generators (PDF, PostScript, etc). All elements reside in a separate namespace, `http://www.renderx.com/DITYPE/xep`. All lengths are measured in units of 0.001 pt (1/72,000 inch), and expressed as integers.

The specification for DiType Intermediate Output Format is a RelaxNG grammar stored in a file named "xepout.rnc". This file is available within each distribution of DiType in "lib/schema" directory. The file "xepout.rnc" is used in DiType for validation of input files that pretend to be the DiType Intermediate Format.

The DiType Intermediate Format may serve both as input and as output format for DiType. The following commands will process an input file "afile.xml" to the Intermediate Format, resulting in a file named "afile.xep":

```
ditype -f xep afile.xml
```

```
cat afile.xml | ditype -f xep > afile.xep
```

The following is an example of formatting a .xep file to PostScript:

```
ditype afile.xep
```

```
cat afile.xep | ditype -f ps > afile.ps
```

It is also possible to create pipes like this:

```
cat afile.xml | ditype -f xep | ditype -f pdf > afile.pdf
```

```
cat afile.xml | ditype -f xep | 4xslt - postprocess.xsl | \
                 ditype -f pdf > afile.pdf
```

The following table provides a detailed description of the output elements of the Intermediate Format:

Table E.1. Output Elements

Element	Description	Attributes
`<document>`	The root element. At its beginning and at its end, initialization and finalization are usually performed.	• `creator` - Information about the application that created the document.

Element	Description	Attributes
		• `initial-destination` (optional) - The part of the document that is moved into focus when the document is first opened. • There are several other attributes (such as `author` or `title`) that transfer unchanged from `<rx:meta-field>` extension formatting objects (if present) in the source document. Their use is implementation-dependent; for example, PDF generator uses them to fill the fields of the `Info` object.
`<page>`	Wraps a single page of the document. There is one `<page>` element for each page in the document.	• `height` - Height of the page • `width` - Width of the page. • `page-id` - Page number label. • `page-number` - Page number as an ordinal.
`<rotate>`	Rotates the coordinate system.	• `phi` - Rotation angle, in degrees. May take values in multiples of 90: 0, 90, 180, 270 degrees. @@@@Clockwise or counterclockwise?
`<translate>`	Shifts the origin of the coordinate system.	• `x` - Horizontal shift distance. • `y` - Vertical shift distance.
`<word-spacing>`	Sets word spacing (additional spacing between words).	• `value` - The value of word spacing.
`<letter-spacing>`	Sets letter spacing (additional spacing between non-space characters).	• `value` - The value of letter spacing.
`<font-stretch>`	Sets font-stretch factor. @@@@What is a "font-stretch factor"?	• `value` - The value of font-stretch factor. @@@@Values?
`<font>`	Changes the current font.	• `family` - Font family. • `weight` - Font weight (100 to 900). @@@@What is the "normal" font weight - 500?

Element	Description	Attributes
		• `style` - Font style (normal, italic, oblique, or backslant). • `variant` - Font variant (normal or small-caps). • `size` - Font size.
`<text>`	Prints a character string.	• `x` - Horizontal position of the initial point on the baseline. • `y` - Vertical position of the initial point on the baseline. • `value` - The text string to print. • `width` - Text width.
`<more-text>`	Prints a character string adjacent to the string generated by the parent `text` and preceding `more-text` elements. The element is used for output of text fragments consisting of glyphs from multiple fonts.	• `value` - The text string to print.
`<line>`	Draws a line.	• `x-from` - Horizontal position of initial point. • `y-from` - Vertical position of initial point. • `x-till` - Horizontal position of final point. • `y-till` - Vertical position of final point. • `thickness` - The thickness of the line. • `style` - The style of the line.
`<image>`	Embeds an external image.	• `src` - The source URL of the image. • `base` - The base directory to resolve hrefs in the image. • `type` - The image MIME type.

Element	Description	Attributes
		• `x-from` - Horizontal position of the lower-left corner of the image. • `y-from` - Vertical position of the lower-left corner of the image. • `scale-x` - Horizontal scaling factor. • `scale-y` - Vertical scaling factor. • `role` - Alternate description for accessible PDFs.
`<gray-color>`	Sets the current color for stroking and filling. The color is chosen in a grayscale color space.	• `gray` - Gray color, value: *0* to *1*
`<rgb-color>`	Sets the current color for stroking and filling. The color is chosen in an RGB color space (additive).	• `red` - Red color, value: *0* to *1* • `green` - Green color, value: *0* to *1* • `blue` - Blue color, value: *0* to *1*
`<cmyk-color>`	Sets the current color for stroking and filling. The color is chosen in CMYK color space (subtractive).	• `cyan` - Cyan color, value: *0* to *1* • `magenta` - Magenta color, value: *0* to *1* • `yellow` - Yellow color, value: *0* to *1* • `black` - Black color, value: *0* to *1*
`<spot-color>`	Sets the current color for stroking and filling. The color is chosen in a spot color space (subtractive).	• `colorant` - Colorant name • `tint` - Color intensity, value: 0 to1 • `alt-gray` - Alternate gray color, value: *0* to *1* • `alt-red` - Alternate red color, value: *0* to *1* • `alt-green` - Alternate green color, value: *0* to *1* • `alt-blue` - Alternate blue color, value: *0* to *1* • `alt-cyan` - Alternate cyan color, value: *0* to *1*

Element	Description	Attributes
		• `alt-magenta` - Alternate magenta color, value: *`0`* to *`1`* • `alt-yellow` - Alternate yellow color, value: *`0`* to *`1`* • `alt-black` - Alternate black color, value: *`0`* to *`1`* To describe an alternate color, one of the following attribute sets must be present: • `alt-gray` — The default color is grayscale. • `alt-red`, `alt-green`, `alt-blue` — The default color is RGB. • `alt-cyan`, `alt-magenta`, `alt-yellow`, `alt-black` — The default color is CMYK.
`<registration-color>`	Sets the current color for stroking and filling. The color is chosen in registration color space (subtractive): it appears on all separations in the document.	• `tint` - Color intensity, value: 0 to 1
`<rectangle>`	Draws a filled rectangle.	• `x-from` - The horizontal position of the lower-left corner. • `y-from` - The vertical position of the lower-left corner. • `x-till` - The horizontal position of the upper-right corner. • `y-till` - The vertical position of the upper-right corner.
`<clip>`	Sets the clipping area.	• `x-from` - The horizontal position of the lower-left corner. • `y-from` - The vertical position of the lower-left corner.

Element	Description	Attributes
		• `x-till` - The horizontal position of the upper-right corner. • `y-till` - The vertical position of the upper-right corner.
`<polygon>`	Draws a filled polygon. All verticals but the first one are specified in the contained `<point>` elements.	• `x-from` - The horizontal position of the first vertex. • `y-from` - The vertical position of the first vertex.
`<point>`	Adds a vertex to a polygon.	• `x-till` - The horizontal position. • `y-till` - The vertical position.
`<target>`	Sets the endpoint for an internal destination.	• `name` - Internal destination name (`id` of the element that created the target). • `x` - Horizontal position. • `y` - Vertical position.
`<internal-link>`	Specifies an internal link destination.	• `destination-id` - Name of the target endpoint. It should match the `name` attribute of a `<target>` element somewhere in the document. • `destination` - The page number to point the link to. • `x-destination` - Horizontal position of the destination point. • `y-destination` - Vertical position of the destination point.
`<external-link>`	Specifies an external link destination.	• `destination` - a URL. • `show-destination` - Controls whether to create a new window when jumping to the link target.
`<internal-bookmark>`	Specifies an internal bookmark destination.	• `label` - Bookmark text. • `id` - Bookmark ID - a positive integer.

Element	Description	Attributes
		• `parent-id` - ID of the parent bookmark, or `0` if the bookmark is a top-level element. • `destination-id` - Name of the target endpoint. It should match the `name` attribute of a `<target>` element somewhere in the document. • `destination` - Page number to point the link to. • `x-destination` - Horizontal position of the destination point. • `y-destination` - Vertical position of the destination point. • `collapse-subtree` - Initial state (collapsed or expanded).
`<external-bookmark>`	Specifies an external bookmark destination.	• `label` - Bookmark text. • `id` - Bookmark ID - a positive integer. • `parent-id` - ID of the parent bookmark, or `0` if the bookmark is a top-level element. • `destination` - A URL. • `collapse-subtree` - Initial state (collapsed or expanded). • `show-destination` - Controls whether to create a new window when jumping to the link target.

Processing instructions may appear in the output. They are taken from the source file and passed straight to the generator with no modification. The placement of processing instructions meets the following conditions:

- Instructions placed before `<fo:root>` element in the source file are reproduced at the very top, before the root `<document>` element.
- Instructions placed inside an `<fo:simple-page-master>` element are reproduced on each page generated using that master, immediately after the opening tag of the `<page>` element.

All other processing instructions may vanish during formatting. Except for those specified above, ordering of instructions is not preserved.

TODO: describe differences between XEPOUT of XEP4 and of DiType; make the description table match the schema; check the PIs.

Index

R

S

T

U

V

X