ChangeMan® ZDD

.NET Programming Interface Guide

Product version: 8.2

Publication date: September 2018

Table of Contents

Welcome to ChangeMan® ZDD

ChangeMan® ZDD is a network file system that operates on a PC networked with a z/OS® operating system. From your PC, you can access data sets, job output, and ChangeMan® ZMF components that reside on a z/OS server.

This document describes the .NET programming interface for ChangeMan ZDD. See the Readme file for the latest updates and corrections for this manual. The Readme file is available through the Micro Focus SupportLine website at https://www.microfocus.com/support-and-services/documentation/.

Audience and scope

This manual is intended for System Administrators or any other users who want to perform ChangeMan ZDD operations from their own programs and scripts using any language that supports the .NET CLI (Common Language Infrastructure).

Using the .NET interface to access the functionality of ChangeMan ZDD allows you to simplify some common tasks, such as:

- Automating configuration tasks for setting up ChangeMan ZDD on multiple desktops.
- Logging on to a z/OS server from your program or script.
- Submitting JCL to a z/OS server from your program or script.

Change bars

Change bars in the left margin mark the substantive changes that have been made to this manual for this release.

Manual Organization

This manual is organized as follows:

This chapter...	Contains this information...
1	Overview of ChangeMan ZDD and the .NET interface.
2	Description of the ChangeMan ZDD object model and how to use the .NET interface to access ChangeMan ZDD functions.
3	Class reference.
4	Examples of how to use ChangeMan ZDD functions within scripts.
Index	Index of ChangeMan ZDD subjects.

Guide to ChangeMan ZDD Documentation

The following sections provide basic information about ChangeMan ZDD documentation and related documents. These manuals are available through the Micro Focus Supportline website at https://www.microfocus.com/support-and-services/documentation/.

ChangeMan ZDD Documentation Suite

The ChangeMan ZDD documentation set includes the following manuals in PDF format.

Manual	Description
ChangeMan ZDD User's Guide	Explains how to: ■ Install and configure the client components on your PC ■ Access and perform operations on mainframe data from your desktop
ChangeMan ZDD .NET Programming Interface Guide	Describes how to use the .NET programming interface to access ChangeMan ZDD functionality from your own programs and scripts.
ChangeMan ZDD Tools Guide	Describes the following tools that you can use to assist in your development: ■ ChangeMan Edit ■ ChangeMan Diff These tools use the Template Manager to control how your code is displayed.
ChangeMan ZDD COM Programming Interface Guide	Describes how to access ChangeMan ZDD functionality, using COM automation, from your own programs and scripts.
ChangeMan ZDD Server Installation Guide	Instructions for installing the server components of ChangeMan ZDD on the mainframe.
ChangeMan ZDD Edit Reference Card	Provides a summary of keyboard shortcuts that you can use with ZDD editing facilities.
SER10TY User's Guide	Instructions for applying licenses to enable ChangeMan ZDD servers on the mainframe.

Related Documents

The following documents provide additional information that may be useful to ChangeMan ZDD users.

Manual	Description
ChangeMan ZMF User's Guide	Provides instructions for using functions and facilities of ChangeMan ZMF to manage changes to application software. Many of these functions are available through ChangeMan ZDD.
ChangeMan ZMF Messages Guide	Provides explanations for informational, warning, and error messages for ChangeMan ZMF. These messages may be displayed when accessing ChangeMan ZMF through ChangeMan ZDD.
ChangeMan ZMF: XML Services User's Guide	Describes how to use XML Services, an XML programming interface to ChangeMan ZMF.

Using the Manuals

The ChangeMan ZDD manuals use the Adobe Portable Document Format (PDF). To view PDF files, use Adobe® Reader®, which is freely available from www.adobe.com.

TIP Be sure to download the *full version* of Reader. The more basic version does not include the search feature.

This section highlights some of the main Reader features. For more detailed information, see the Adobe Reader online help system.

The PDF manuals include the following features:

- **Bookmarks.** All of the manuals contain predefined bookmarks that make it easy for you to quickly jump to a specific topic. By default, the bookmarks appear to the left of each online manual.
- **Links.** Cross-reference links within a manual enable you to jump to other sections within the manual and to other manuals with a single mouse click. These links appear in blue.
- **Printing.** While viewing a manual, you can print the current page, a range of pages, or the entire manual.
- **Advanced search.** Starting with version 6, Adobe Reader includes an advanced search feature that enables you to search across multiple PDF files in a specified directory. (This is in addition to using any search index created by Adobe Catalog—see step 3 below.)

To search within multiple PDF documents at once, perform the following steps (requires Adobe Reader version 6 or higher):

1. In Adobe Reader, select Edit | Search (or press CTRL+F).
2. In the text box, enter the word or phrase for which you want to search.
3. Select the **All PDF Documents in** option, and browse to select the folder in which you want to search.
4. Optionally, select one or more of the additional search options, such as **Whole words only** and **Case-Sensitive**.
5. Click the **Search** button.

NOTE Optionally, you can click the **Use Advanced Search Options** link near the lower right corner of the application window to enable additional, more powerful search options. (If this link says **Use Basic Search Options** instead, the advanced options are already enabled.) For details, see Adobe Reader's online help.

Accessing Online Help

The online help is the primary source of information about ChangeMan ZDD. The online help includes:

- Overviews of key elements within the application
- Detailed procedures for completing tasks

- Context-sensitive descriptions of fields and buttons

Viewing Help Topics

You can Help topics by clicking the Help button in the dialog box in which you are working. From there, you can do the following:

To	Do This
View a list of topics in the Contents	Click **Contents**.
Locate a topic in the Index	Click **Index**.
Locate an overview or procedure by searching on a word or words	Click **Search**.

Viewing Context-Sensitive Help

To view field-level help for an item in a dialog box:

- Click ? and then click the field or button for which you want a description.

  or

- Position the cursor in the field and press F1.

Accessing Help for the ChangeMan Utilities

When you are using the ChangeMan Edit and ChangeMan Diff utilities, you can open the online Help by:

- Pressing F1 from anywhere in the screen.
- Holding the left or right mouse button down on a toolbar icon or menu command and pressing F1.

Chapter 1

Introduction

Overview

The .NET programming interface for ChangeMan ZDD allows you to access the functionality of ChangeMan ZDD from your own programs and scripts using any language that supports the .NET CLI (Common Language Infrastructure).

NOTE The ChangeMan ZDD .NET programming interface is similar to the older COM programming interface, but is more powerful and flexible. It is recommended that new programs utilize the .NET interface rather than the older COM interface.

Through the .NET interface, ChangeMan ZDD exposes its functionality as a set of programmable *objects*. Each object can be programmatically examined and controlled. Examples of ChangeMan ZDD objects are: *network*, *servers*, *ChangeMan instances*, and *folders*.

Each object exposes a set of *properties* and *methods*. A *property* is an attribute of an object that can be set or retrieved. A *method* is a function that performs some action on an object. For a server object, examples of properties are *server name* or *IP address*; examples of methods for a server object are *logon* or *logoff*.

A special type of object is a *collection* object, which contains a set of other objects. A *servers* object is a collection of *server* objects, and a *folders* object is a collection of *folder* objects.

Languages

The ChangeMan ZDD .NET interface allows ChangeMan ZDD operations to be performed from C#, C++/CLI, Visual Basic .NET, J#, JScript .NET, or any other language that supports .NET. Following are some typical operations you can perform from a program or script:

- Configure ChangeMan ZDD for a new user (see "Configuring ChangeMan ZDD for a New User" on page 196).
- Submit JCL to a server (see "Submitting JCL to a Server" on page 192).
- Log on to a server (see "Logging on to a Server" on page 188).

In this document, examples are shown in the following order:

- C#
- C++/CLI
- Visual Basic
- JScript .NET

Other languages may be used as well, but examples are not given.

NOTE JScript .NET files must be compiled using the “jsc” compiler. The WSH (Windows Script Host) that was used to run older JScript files does not support the .NET environment.

Programming Samples

There are several small programming samples installed in the ChangeMan ZDD “Samples” subdirectory. Some of these samples are described in Chapter 4, "Examples" on page 187.

Additionally, there is a very large C# sample called “TestApi”, which is a console application that can be used to test all of the various API features. In “TestApi” you can find coding examples for virtually every function available in the programming interface. There is a pre-built copy of “TestApi.exe” in the ChangeMan ZDD installation directory.

Security

ChangeMan ZDD is compatible with RACF®, CA-ACF2®, and CA-Top Secret®.

Access to mainframe objects and functions is granted through your mainframe security system. You are required to provide your user ID and password in ChangeMan ZDD to connect to the mainframe.

The operation of ChangeMan ZDD does not affect the existing operation of either mainframe-based applications or PC network operations.

Compatibility

PC Requirements

- Windows® operating system
- Microsoft® .NET Framework 4.0

NOTE Refer to the Readme for the supported versions.

Mainframe Server Requirements

- ChangeMan ZDD server installed on the mainframe LPARs to be accessed by ChangeMan ZDD on your PC.
- IBM® z/OS® operating system (any version supported by IBM).
- TCP/IP must be installed and running.

ChangeMan ZMF Requirements

One of the following releases are required for accessing ChangeMan ZMF functionality from your program or script:

- ChangeMan ZMF 8.1 - any release
- ChangeMan ZMF 7.1 - any release

- ChangeMan ZMF 6.1 - any release

NOTE When using ChangeMan ZDD 8.1.4 with earlier releases of ChangeMan ZMF, only the functionality supported within that ChangeMan ZMF release will be available.

Chapter 2

Using the Programming Interface

This chapter describes the ChangeMan ZDD object model and how to access ChangeMan ZDD functionality from your own programs and scripts. You may use any language that supports .NET, such as C#, C++/CLI, Visual Basic.NET, J#, and JScript .NET.

NOTE JScript .NET files must be compiled using the "jsc" compiler. The WSH (Windows Script Host) that was used to run older JScript files does not support the .NET environment.

Accessing ChangeMan ZDD

The ChangeMan ZDD .NET programming interface is implemented in ZosApi.dll. All of the ChangeMan ZDD classes belong to the ZosApi namespace. To use the ChangeMan ZDD .NET classes, you must copy ZosApi.dll into the directory where your application programs or scripts reside. ZosApi.dll must match the version of ChangeMan ZDD installed on your computer.

Your program must import the ZosApi namespace as shown in the examples below.

For this language . . .	Use this code . . .
C#	`using ZosApi;`
C++	`using namespace ZosApi;`
Visual Basic	`Imports ZosApi`
JScript	`import ZosApi;`

Accessing ChangeMan ZDD begins with creating a ZosNetwork object. The ZosNetwork object is created in the standard way for creating any object in the language being used, for example:

For this language . . .	Use this code . . .
C# J# JScript	`new ZosNetwork()`
C++	`gcnew ZosNetwork()`
Visual Basic	`Dim ... As New ZosNetwork()`

For an example of how to access the ChangeMan ZDD network, see "ZosNetwork Constructor" on page 108.

Object Model

This sections lists the object types and illustrates the relationships between the objects. Detailed specifications and examples for each object are documented in Chapter 3, "Class Reference" on page 59.

Object Types

The following table summarizes the types of objects available in the ChangeMan ZDD object model:

Object	Description
ZosApplication	A ChangeMan ZMF application.
ZosBaselineLibrary	A ChangeMan ZMF baseline library.

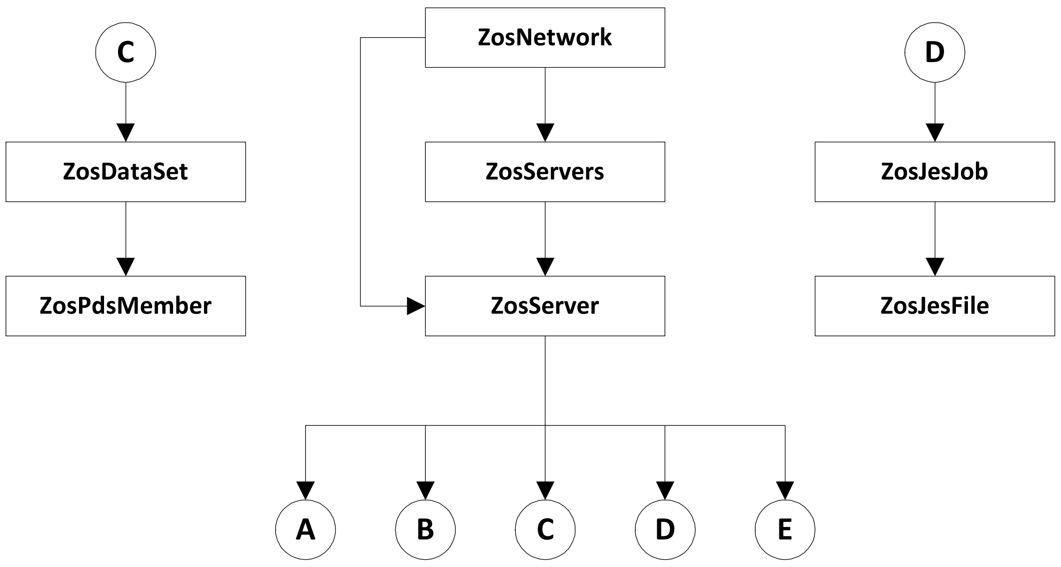
Object	Description
ZosBuildInfo	A set of build information used to build a component.
ZosChangeManInstance	A ChangeMan ZMF instance.
ZosChangeManInstances	A collection of all ChangeMan ZMF instances on the same server.
ZosCheckInStatus	Status of a check in operation.
ZosComponentHistory	ChangeMan component history record
ZosComponentPromotionHistory	ChangeMan component promotion history record
ZosConnectionLock	Obtains and locks a server connection ID
ZosDataSet	A data set.
ZosDataSetFolder	A data set folder.
ZosDataSetFolders	A collection of all data set folders with the same parent folder.
ZosDataSetInfo	A set of data set properties that can be used to create a new data set.
ZosDataSetProfile	A data set profile.
ZosDataSetProfiles	A collection of all data set profiles for a server.
ZosFileExtensionMapping	A file extension mapping.
ZosFileExtensionMappings	A collection of all file extension mappings for a server.
ZosFileFormatMapping	A file format mapping.
ZosFileFormatMappings	A collection of all file format mappings for a server.
ZosJesFile	A JES spool file.
ZosJesJob	A JES job.
ZosJobFolder	A job folder.
ZosJobFolders	A collection of all job folders with the same parent folder.
ZosLibTypeMapping	A library type mapping.
ZosLibTypeMappings	A collection of all library type mappings for a server.
ZosNameFilters	A collection of all name filters for a folder.
ZosNameType	Name/type pair used for component names and types.
ZosNameValue	Name/value pair used to represent user variables.
ZosNetwork	The entire Network.
ZosPackage	A ChangeMan package.
ZosPackageApprover	ChangeMan package approver.
ZosPackageComponentDirectory	Unix subdirectory for ChangeMan package component files.

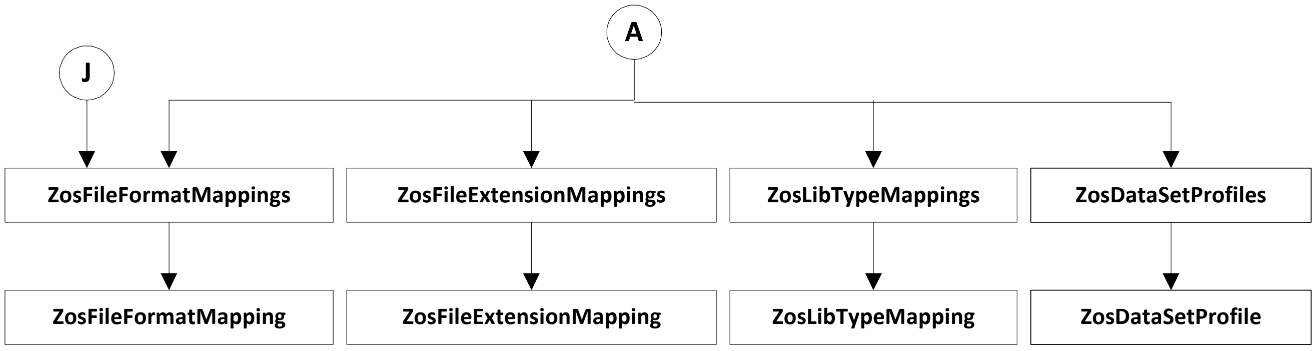
Object	Description
ZosPackageComponentFile	ChangeMan package component (PDS member or Unix file).
ZosPackageComponentObject	Base for ChangeMan package component members, files, and directories (ZosPackageComponentFile, ZosPackageComponentDirectory).
ZosPackageInfo	A set of properties that can be used to create a new package.
ZosPackageLibrary	ChangeMan package library.
ZosPackagePromotionHistory	ChangeMan package promotion history record
ZosPackageSite	Package site information.
ZosPdsMember	Partitioned data set member.
ZosPrefixMapping	A data set name prefix mapping.
ZosPrefixeMappings	A collection of all data set name prefix mappings for a folder.
ZosPromotionLevel	A ChangeMan promotion level.
ZosPromotionOverlay	ChangeMan component promotion overlay information
ZosPromotionLibrary	A ChangeMan promotion library.
ZosPromotionSite	A ChangeMan promotion site.
ZosRelease	ChangeMan release.
ZosReleaseApprover	ChangeMan release approver.
ZosReleaseArea	ChangeMan release area.
ZosReleaseComponentDirectory	Unix subdirectory for ChangeMan release component files.
ZosReleaseComponentFile	ChangeMan package component (PDS member or Unix file).
ZosReleaseComponentObject	Base for ChangeMan release component members, files, and directories (ZosReleaseComponentFile, ZosReleaseComponentDirectory).
ZosReleaseLibrary	ChangeMan release library.
ZosRetrieveStatus	Status of a retrieve operation.
ZosScratchRenameInfo	A ChangeMan request to scratch or rename a component.
ZosServer	A server.
ZosServers	A collection of all servers in the network.
ZosUnixDirectory	Unix directory (derived from ZosUnixObject).
ZosUnixFile	Unix File (derived from ZosUnixObject).
ZosUnixFolder	Unix folder, which is a local Windows alias for a Unix directory.
ZosUnixFolders	Collection of Unix folders for a server.

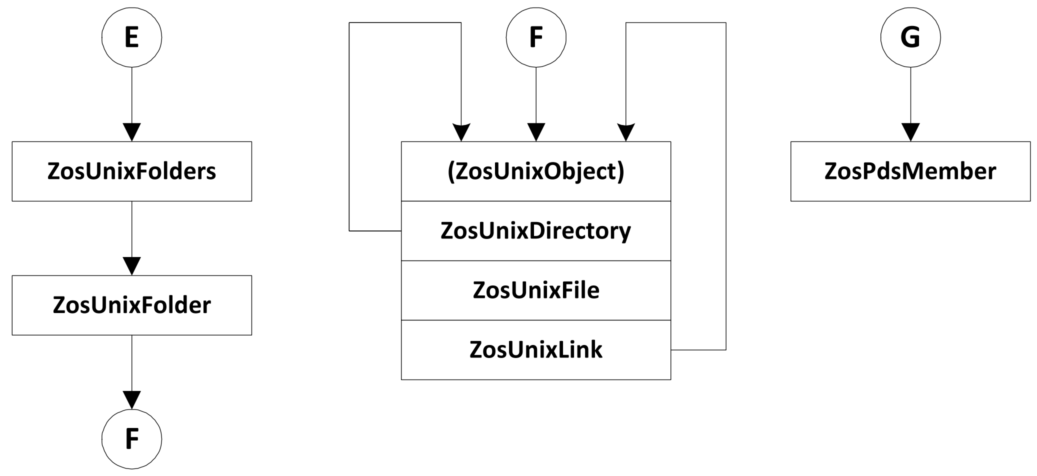
Object	Description
ZosUnixLink	Unix symbolic link (derived from ZosUnixObject).
ZosUnixObject	Unix file system object (ZosUnixDirectory, ZosUnixFile, ZosUnixLink).

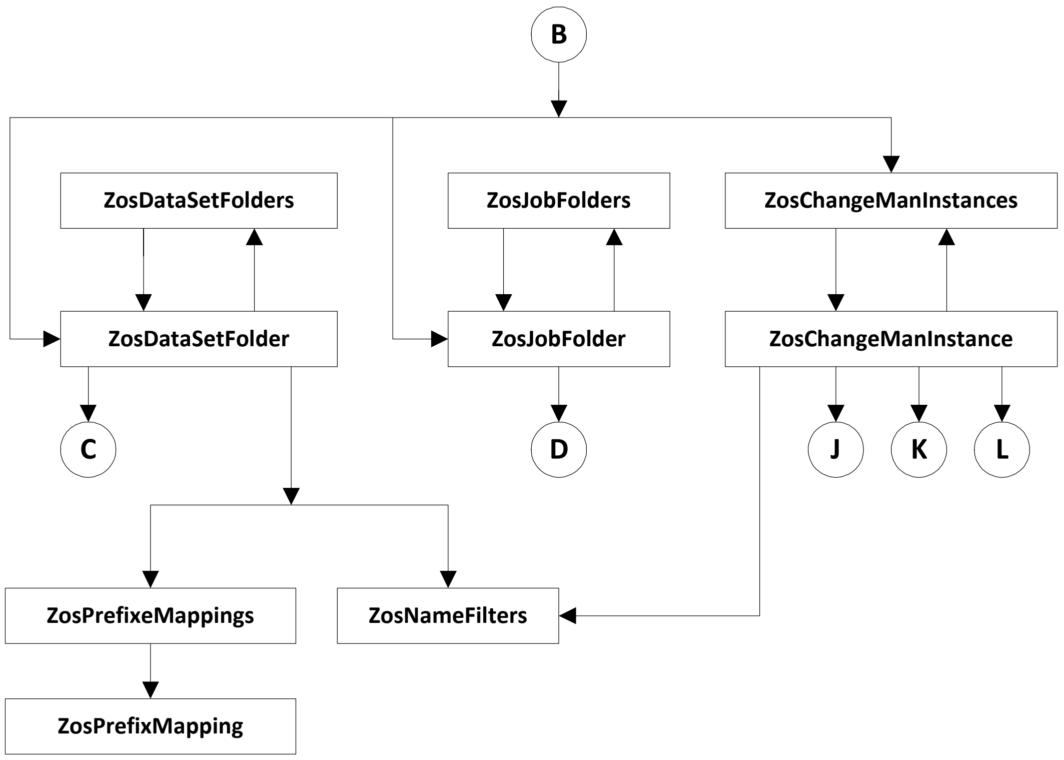
Object Model Diagrams

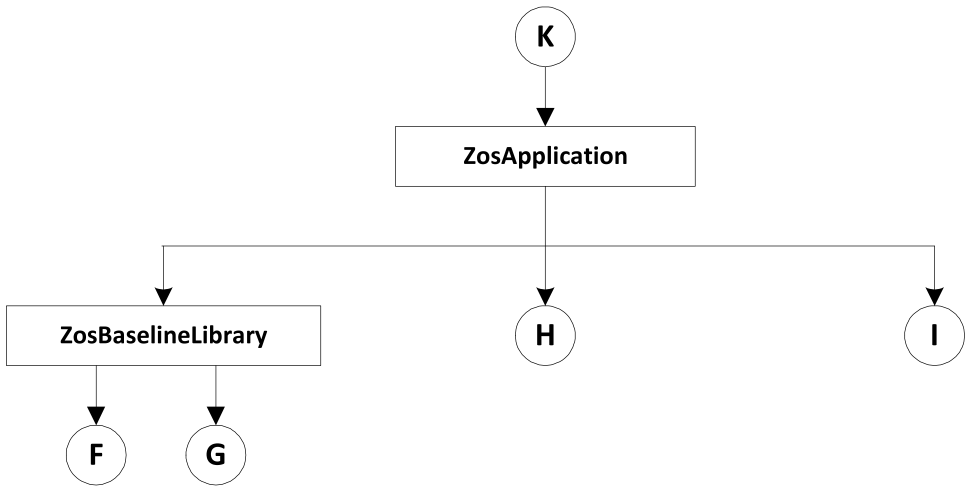
The ChangeMan ZDD object model is illustrated in the following diagrams. The *ZosNetwork* object is always the starting point. All of the other objects are obtained as properties of another object. The arrows show how each object is obtained from another object.

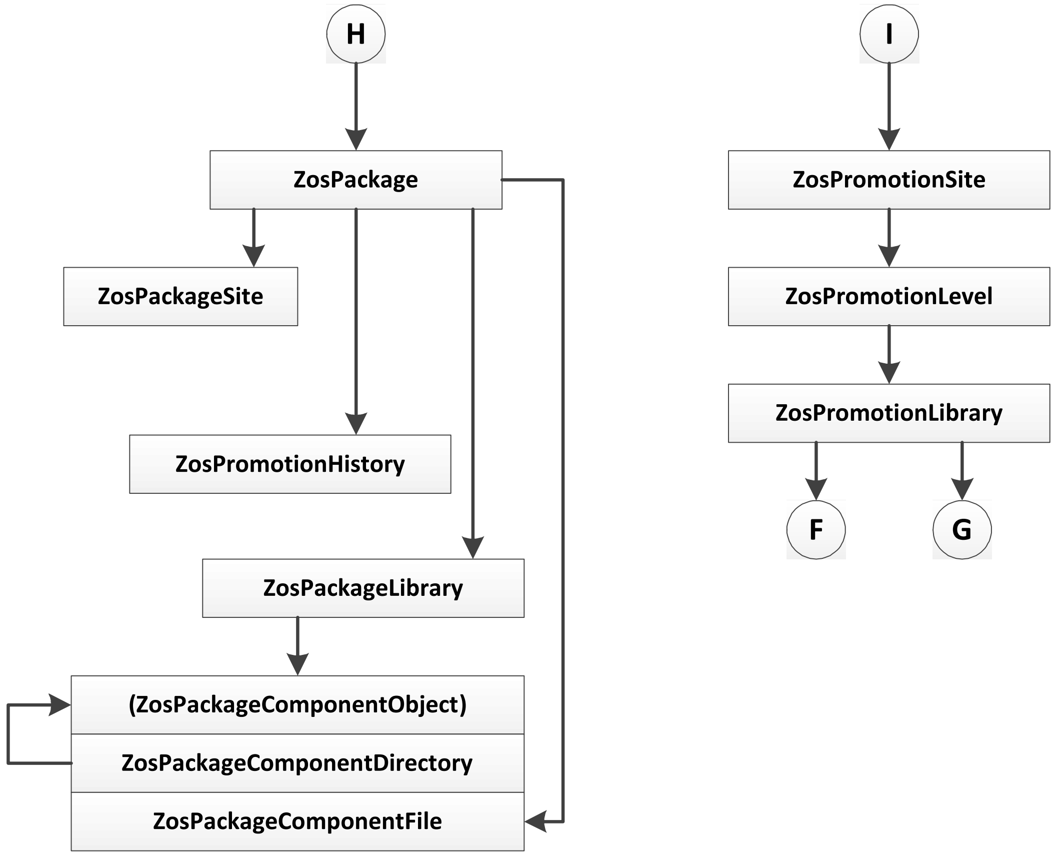


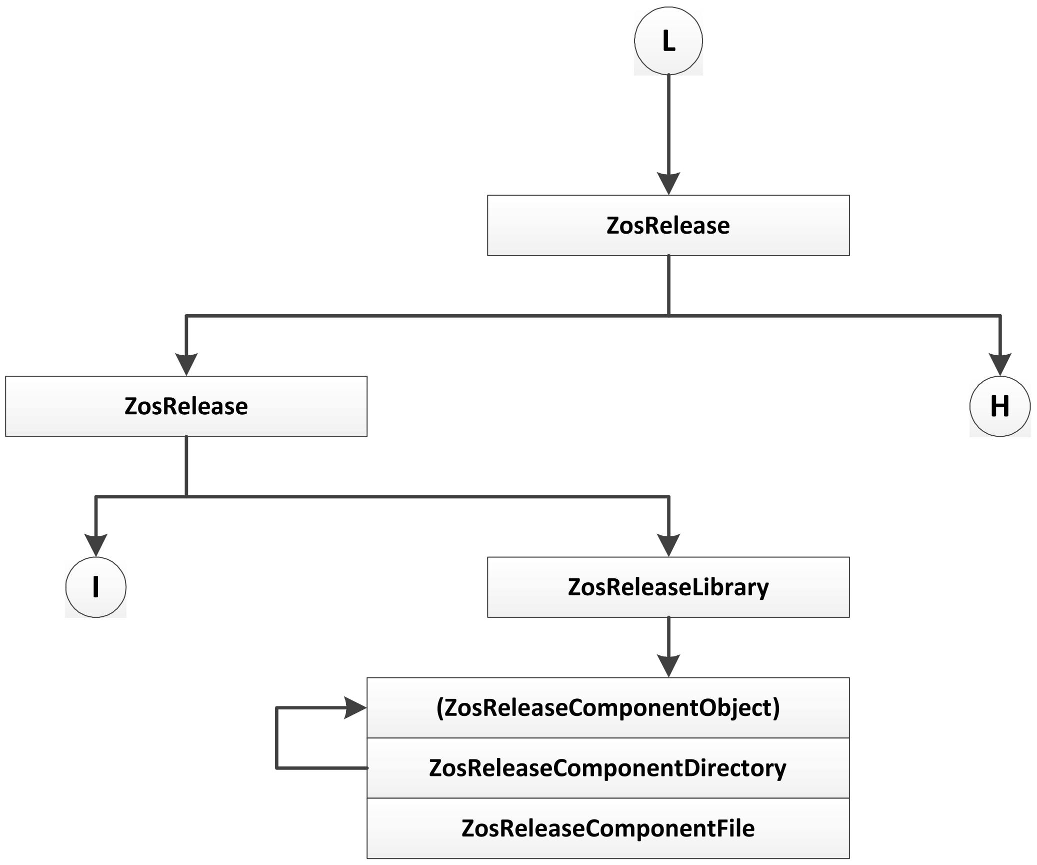



E
ZosUnixFolders
ZosUnixFolder
F
F
(ZosUnixObject)
ZosUnixDirectory
ZosUnixFile
ZosUnixLink
G
ZosPdsMember


B
ZosDataSetFolders
ZosDataSetFolder
C
ZosJobFolders
ZosJobFolder
D
ZosChangeManInstances
ZosChangeManInstance
J
K
L
ZosPrefixeMappings
ZosPrefixMapping
ZosNameFilters


K
ZosApplication
ZosBaselineLibrary
H
I
F
G


H
ZosPackage
ZosPackageSite
ZosPromotionHistory
ZosPackageLibrary
(ZosPackageComponentObject)
ZosPackageComponentDirectory
ZosPackageComponentFile
I
ZosPromotionSite
ZosPromotionLevel
ZosPromotionLibrary
F
G



Path Names

You may use the standard .NET classes in the System.IO namespace to access mainframe data using ChangeMan ZDD. You can traverse ChangeMan ZDD’s virtual folders, as well as read and write files using standard .NET classes.

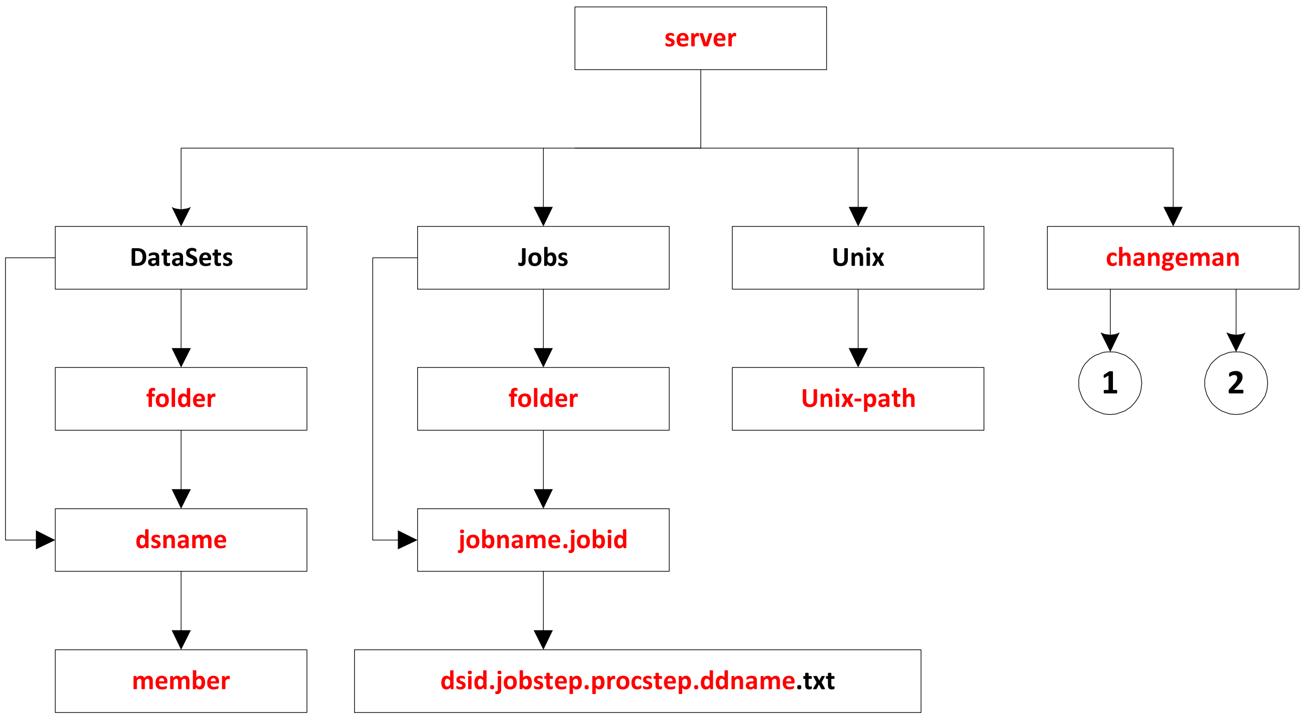
To access mainframe data, path names must be specified according to ChangeMan ZDD syntax rules. Path names are specified in UNC format, where the path name is preceded by a double backslash, followed by the server name. Each component in the path name is separated by a backslash.

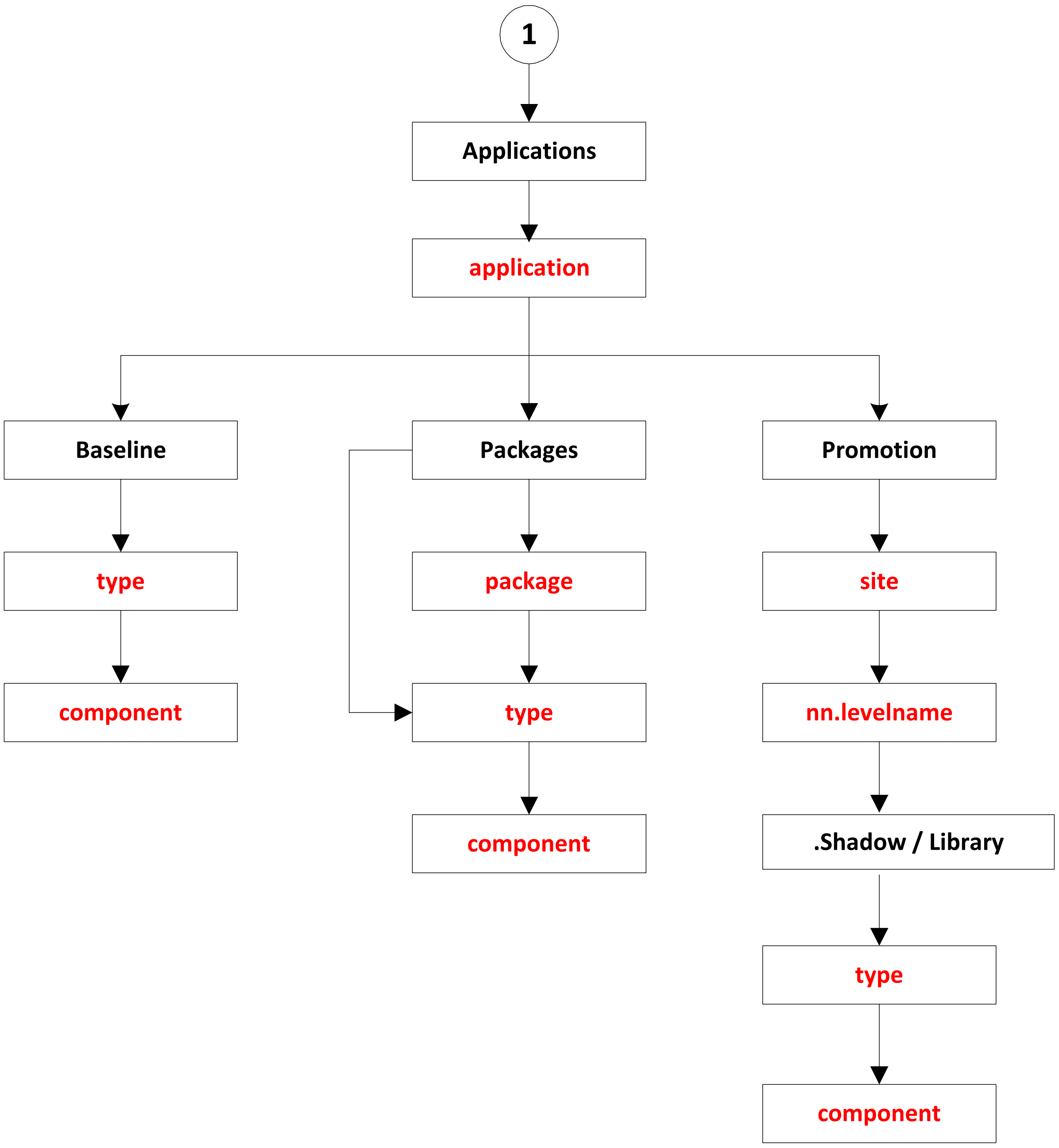
Following are some examples of path names:

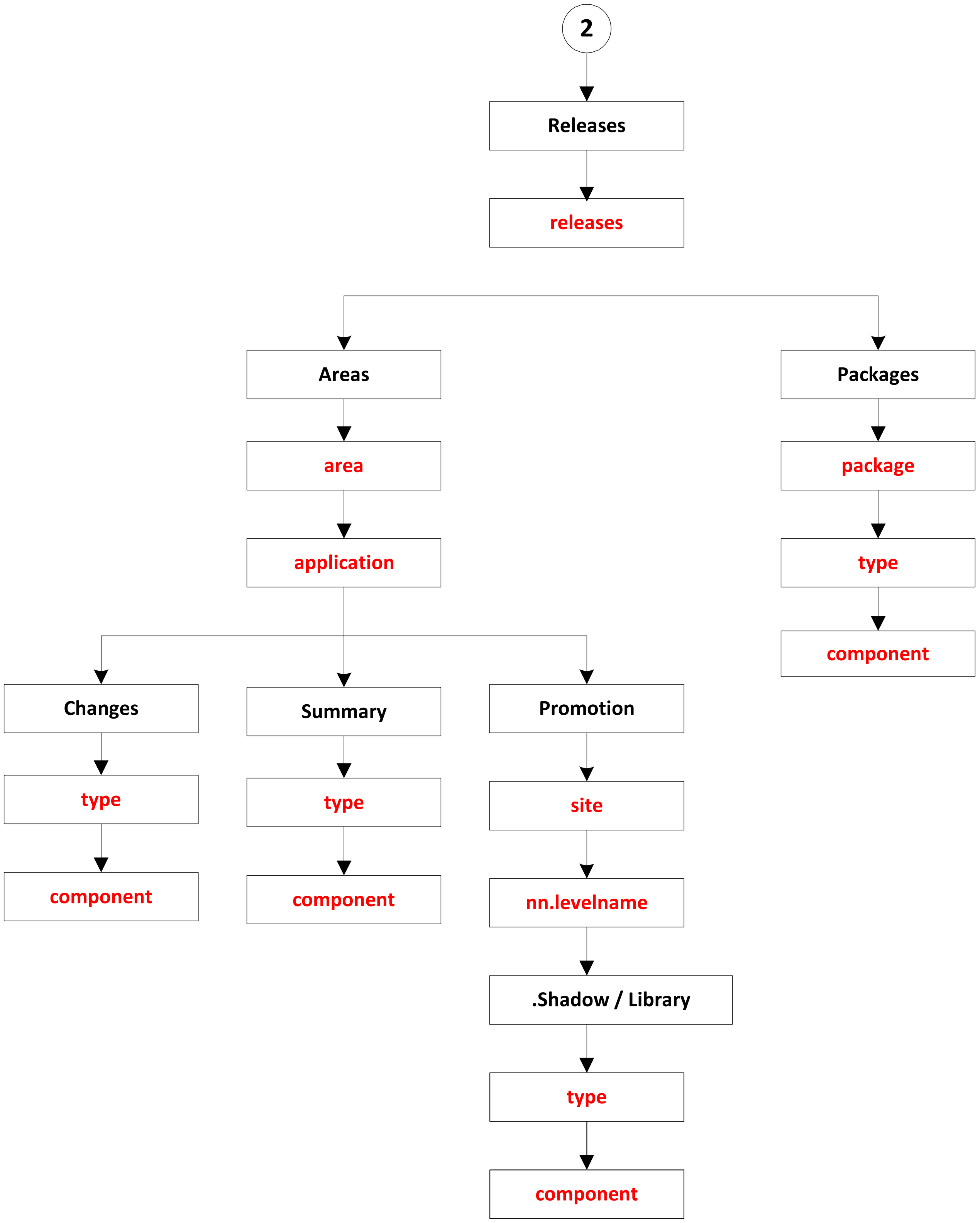
```
\\MyServer\DataSets\MY.TEST.SRC\PAYROLL.src
\\MyServer\Jobs\MYJOB.J0123456\D0000007.JOBSTEP1.PROCSTEP2.SYSPRINT.
    txt
\\MyServer\MyChangeMan\DEMO\Packages\DEMO000123\SRC\PAYROLL.src
```

The chart below illustrates the hierarchical directory structure in the ZDD path name syntax. Names that are colored red are variable, in which you substitute your own actual values for the names.

Names that are colored black are fixed and you specify them exactly as shown.




1
Applications
application
Baseline
type
component
Packages
package
type
component
Promotion
site
nn.levelname
.Shadow / Library
type
component


2
Releases
releases
Areas
area
application
Changes
type
component
Summary
type
component
Promotion
site
nn.levelname
.Shadow / Library
type
component
Packages
package
type
component

It is possible to have several user IDs logged on to a single server at the same time using alternate connections. Each server can have up to 255 alternate connections.

An alternate connection is specified by appending the connection ID (1 – 255) to the server name, separated by a colon (":"). It can also be specified by appending the connection ID enclosed in parentheses. The default connection ID is 0, and does not need to be specified.

```
\\MyServer:23\MyChangeMan\...
\\MyServer(23)\MyChangeMan\...
```

For more information, see the section entitled "Alternate Connections" on page 34.

Wild Characters

Standard Patterns

Some of the classes have methods with string arguments that allow wildcard characters. These strings are used as pattern-matching filters.

The following wildcard characters can be used in filter pattern strings:

Wildcard Character	Description
*	Matches zero or more characters
?	Matches a single character

For example, the pattern "A*" would match any string that starts with the letter A. The pattern "*Z" would match any string that ends in the letter Z. The pattern "A*Z" would match any string that starts with A and ends with Z. The pattern "A??D" would match a string that starts with A, followed by exactly two characters, and followed by D. Finally, the pattern "*" would match any string.

The examples below illustrate wildcard patterns.

Pattern	Match	No Match
A*	A AB ABCDEFGH	B BA
*Z	Z WXYZ	ZA AZA
A??DE	ABCDE AXYDE	ABCDEF AXYZDE
A*DE	ABCDE AXXXXXDE	ABCDEF
*	Z ABCDEFGH	

Data Set Name Patterns

As data set filtering functions are performed on the server, the IBM SMS syntax rules for wild characters in data set name patterns must be followed. The asterisk character works a little differently in patterns for data set names than it does in other patterns.

The following wildcard characters can be used in data set name pattern strings:

Wildcard Character	Description
*	Matches zero or more characters within a single data set name qualifier.
**	Matches zero or more data set name qualifiers.
?	Matches a single character.

NOTE A single "*" never includes multiple data set name qualifiers. A double "**" can represent any number of data set name qualifiers.

The examples below illustrate data set name patterns.

Pattern	Match	No Match
ABC.TEST???.D?TA	ABC.TEST001.DATA	ABC.TEST001.DAATA
ABC.T*.*.DATA	ABC.TEST.NEW.DATA	ABC.TEMP.VERY.OLD.DATA ABC.TEST.DATA ABC.PROD.NEW.DATA
ABC.*X*.DATA	ABC.X.DATA ABC.AX.DATA ABC.AAXBB.DATA ABC.XYZ.DATA	ABC.X.Y.DATA ABC.AABB.DATA
ABC.**.DATA	ABC.DATA ABC.TEMP.DATA ABC.VERY.OLD.DATA	ABC.TEMP.DATA.JUNK

Exceptions

When an error is encountered in the ChangeMan ZDD .NET programming interface, ChangeMan ZDD throws an exception. It is recommended that you enclose your code in a **try/catch** block to catch any exceptions and handle the errors appropriately. In the absence of a **try/catch** block, your program will abnormally terminate.

The examples below illustrate general exception handling in various languages.

C#

```
try
{
  (Some error-prone code here)
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
    Console.WriteLine(e.StackTrace);
}
```

C++

```
try
{
  (Some error-prone code here)
}
catch (Exception^ e)
{
    Console.WriteLine(e->Message);
    Console.WriteLine(e->StackTrace);
}
```

Visual Basic

```
Try
  (Some error-prone code here)

Catch e As Exception
    Console.WriteLine(e.Message);
    Console.WriteLine(e.StackTrace);
End Try
```

Jscript

```
try
{
  (Some error-prone code here)
}
catch (e : Exception)
{
    Console.WriteLine(e.Message);
    Console.WriteLine(e.StackTrace);
}
```

Collections

You can iterate through any of the collection objects using the following statements for the specific language:

Language	Statement
C#	`foreach`
C++	`for each`
Visual Basic	`For Each ... Next`
JScript	`for (var ... in ... )`

Examples:

C#

```
ZosServers servers;
foreach (ZosServer server in servers)
{
    ...
}
```

C++

```
ZosServers^ servers;
for each (ZosServer^ server in servers)
{
    ...
}
```

Visual Basic

```
Dim servers As ZosServers
Dim server As ZosServer
For Each server in servers
    ...
Next
```

Jscript

```
var servers : ZosServers;
for (var server in servers)
{
    ...
}
```

Alternate Connections

In a server application, there may be a requirement to have more than one user ID logged onto the same server at the same time. You can accomplish this by using alternate connections to the server. Each server can have alternate connections, with connection IDs numbered 1 – 255. The default connection has a connection ID of 0.

In path names, an alternate connection is specified by appending the connection ID (1 – 255) to the server name, separated by a colon (":"). It can also be specified by appending the connection ID enclosed in parentheses.

```
\\MyServer:23\...
\\MyServer(23)\...
```

The **ZosConnectionLock** class can be used to reserve a connection ID, and lock the connection ID so that it will not be used by other programs or threads. The default connection ID, 0, will never be locked.

With **ZosConnectionLock** you can either implicitly lock a connection ID via the constructor or you can explicitly lock a connection ID by calling the **Lock** method.

You must unlock the connection ID by calling either the **Unlock** or **Dispose** method of **ZosConnectionLock**. With C# and Visual Basic, you can have the connection automatically unlocked by using a **using** statement. With C++, you can have the connection automatically unlocked by declaring the **ZosConnectionLock** object as a stack variable.

If the connection is not automatically unlocked, then you should ensure that the connection gets unlocked, by explicitly unlocking it in the **finally** block of a **try** / **finally** construction.

The examples below, illustrate obtaining and using an alternate connection in various languages.

C# Automatic

```
using (ZosConnectionLock conlock =
        new ZosConnectionLock("MyServer", true))
{
     short conID = conLock.Connection;
     ZosServer server = network.Servers["MyServer", conID];

     (Do something here)
}
```

C# Explicit

```
ZosConnectionLock conlock = new ZosConnectionLock("MyServer");
try
{
    short conID = conLock.Lock();
    ZosServer server = network.Servers["MyServer", conID];

    (Do something here)
}
finally
{
    conLock.Unlock();
}
```

C++ Automatic

```
ZosConnectionLock conLock("MyServer", true);
short conID = conLock.Connection;
ZosServer^ server = network.Servers["MyServer", conID];

(Do something here)
```

C++ Explicit

```
ZosConnectionLock^ conlock = gcnew ZosConnectionLock("MyServer");
try
{
    short conID = conLock->Lock();
    ZosServer^ server = network.Servers["MyServer", conID];

    (Do something here)
}
finally
{
    conLock->Unlock();
}
```

Visual Basic Automatic

```
Using conLock AsNew ZosConnectionLock("MyServer", True)
    Dim conID As Int16 = conLock.Connection
    Dim server As ZosServer = network.Servers(serverName, conID)

    (Do something here)

End Using
```

Visual Basic Explicit

```
Dim conLock As ZosConnectionLock = new ZosConnectionLock("MyServer")
Try
    Dim conID As Int16 = conLock.Lock()
    Dim server As ZosServer = network.Servers(serverName, conID)

    (Do something here)

Finally
    conLock.Unlock()
End Try
```

Enumerations

The .NET programming interface includes a number of enumerated types that are used as properties and function arguments. The table below lists the enumerations that are described in detail in this section.

Enumeration	Description
ZosAuditPackageOptions	ChangeMan audit package options
ZosAuditReleaseAreaOptions	ChangeMan audit release area options
ZosBuildType	Build type (normal, recompile, relink)
ZosComponentHistoryStatus	ChangeMan component history status flags
ZosComponentHistoryType	ChangeMan component history type
ZosComponentLocation	ChangeMan component location
ZosComponentLockStatus	ChangeMan component lock status
ZosComponentPromotionStatus	Component promotion history status flags
ZosComponentStatus	ChangeMan component status
ZosComponentStatusFlags	ChangeMan component status flags for filtering
ZosDataSetEAttr	Data set extended attributes (NO, OPT)
ZosDataSetType	Data set type (organization)
ZosEnvironmentType	ChangeMan environment type
ZosFileFormat	File format for local files
ZosFileTypeClass	File type class
ZosFreezeType	ChangeMan refreeze / unfreeze type
ZosJobCompletionType	Job completion type
ZosJobHoldType	Job hold type
ZosJobPhase	Job phase (specific status)
ZosJobQueryType	Job filter type
ZosJobStatus	Job status (general status)
ZosJobType	Job type (started task, batch job, TSO, APPC)
ZosLibType	Library type
ZosLikeType	Like ChangeMan library type
ZosOutputQueue	JES output queue type
ZosPackageApprovalAction	Package approval action
ZosPackageLevel	ChangeMan package level
ZosPackageLevelFlags	ChangeMan package level flags for filtering
ZosPackagePromotionAction	Package promotion history action flags
ZosPackagePromotionStatus	Package promotion history status flags
ZosPackageStatus	ChangeMan package status

Enumeration	Description
ZosPackageStatusFlags	ChangeMan package status flags for filtering
ZosPackageType	ChangeMan package type
ZosPackageTypeFlags	ChangeMan package type flags for filtering
ZosProblemActionType	ChangeMan package problem action code
ZosPromotionOverlayStatus	ChangeMan promotion overlay status
ZosPromotionTarget	ChangeMan promotion target type
ZosRecordFormat	Record format
ZosReleaseApprovalAction	ChangeMan release approval action
ZosReleaseApprovalType	ChangeMan release approval type
ZosReleaseAreaStatus	ChangeMan release area status
ZosReleaseAreaType	ChangeMan release area type
ZosReleaseStatus	ChangeMan release status
ZosSchedulerType	ChangeMan scheduler type
ZosSpaceUnit	Space allocation units
ZosStagingVersionLocation	ChangeMan staging version location
ZosStagingVersSaveOption	ChangeMan staging version save option
ZosUnixAccess	Unix access perrmission flags
ZosUnixAccessCheck	Unix access checking flags
ZosUnixFileFormat	Unix file format
ZosUnixFileType	Unix file system object type

ZosAuditPackageOptions Enumeration (Flags)

Name	Value	Description
None	`0x0000`	no options
AutoResolve	`0x0001`	auto resolve out-of-syncs
History	`0x0002`	include history records
UpdateTargetRcOnly	`0x0004`	update only this package return code
FormatReport	`0x0008`	format report for printing
Trace	`0x0010`	enable trace
StagingLibOnly	`0x0020`	audit staging libraries only
PartAsASimple	`0x0040`	audit as simple package (participating only)
PartAsPrimary	`0x0080`	audit as primary package (participating only)
PartByDept	`0x0100`	audit by department number (participating only)
SuppressNotify	`0x0200`	suppress TSO notify message
CrossAppHeaderTop	`0x0400`	cross-application headers, top line only
CrossAppHeaderFull	`0x0800`	cross-application headers, full header
LockPackage	`0x1000`	lock package during audit

ZosAuditReleaseAreaOptions Enumeration (Flags)

Name	Value	Description
None	`0x0000`	no options
AutoResolveAll	`0x0001`	auto-resolve all: build all load modules
AutoResolveComposite	`0x0002`	auto-resolve composite: build composite (like-load)
AutoResolveSubroutine	`0x0003`	auto-resolve subroutine: build subroutines (like-NCAL)
AutoResolveMask	`0x0003`	auto-resolve flag mask
IncludeRelatedApps	`0x0004`	include related applications
IgnoreHigherAreas	`0x0010`	ignore higher areas
IgnoreHigherAreasCond	`0x0020`	ignore higher areas conditionally

ZosBuildType Enumeration

Name	Value	Description
Build	0	normal build
Recompile	1	recompile
Relink	2	re-link

ZosComponentHistoryStatus

Name	Value	Description
None	0x00	no status
CheckOut	0x01	checked out
BackOut	0x02	backed out
Promoted	0x04	promoted
Demoted	0x08	demoted
Deleted	0x10	deleted
Baseline	0x20	baseline
DelArch	0x40	deleted/archived

ZosComponentHistoryType Enumeration

Name	Value	Description
Full	0	full list
Short	1	short list
Concurrent	2	concurrent

ZosComponentLocation Enumeration

Name	Value	Description
None	0	unknown
Development	1	development
Staging	2	staging
Promotion	3	promotion
Baseline	4	baseline
Package	5	package
Backup	6	backup
Zdd	7	ChangeMan ZDD

ZosComponentLockStatus Enumeration

Name	Value	Description
Unlocked	0	unlocked
LockedUser	1	locked by current user
LockedOther	2	locked by other user
Frozen	3	generated
Generated	4	generated
Recompile	5	recompile
Relink	6	re-link

ZosComponentPromotionStatus

Name	Value	Description
None	0x00	no status
Restaged	0x01	component restaged
Overlaid	0x02	component overlaid
Deleted	0x04	component deleted

ZosComponentStatus Enumeration

Name	Value	Description
Active	0	active
Approved	1	approved
Checkout	2	checkout
Demoted	3	demoted
Frozen	4	frozen
Inactive	5	inactive
Incomplete	6	incomplete
Promoted	7	promoted
Refrozen	8	refrozen
Rejected	9	rejected
RemotePromo	10	remote promotion
Submitted	11	submitted
Unfrozen	12	unfrozen

ZosComponentStatusFlags Enumeration (Flags)

Name	Value	Description
Any	0x0000	any status
Active	0x0001	active
Approved	0x0002	approved
Checkout	0x0004	checkout
Demoted	0x0008	demoted
Frozen	0x0010	frozen
Inactive	0x0020	inactive
Incomplete	0x0040	incomplete
Promoted	0x0080	promoted
Refrozen	0x0100	refrozen
Rejected	0x0200	rejected
RemotePromo	0x0400	remote promotion
Submitted	0x0800	submitted
Unfrozen	0x1000	unfrozen

ZosDataSetEAttr Enumeration

Name	Value	Description
Default	0	unspecified
No	1	no
Opt	2	optional

ZosDataSetType Enumeration

Name	Value	Description
None	0x00	unknown
Dir	0x01	direct access
Vsam	0x02	VSAM
Hfs	0x03	HFS
Seq	0x10	sequential
SeqL	0x11	large sequential
SeqE	0x12	extended format
Pds	0x20	PDS

Name	Value	Description
Pdse	`0x21`	PDSE
Lib	`0x22`	Librarian
Pan	`0x23`	Panvalet
Mig	`0x80`	migrated

ZosEnvironmentType Enumeration

Name	Value	Description
None	0	unknown
All	1	all (no remote)
Development	2	development only site
DevProd	3	development and production site
Production	4	production only site

ZosFileFormat Enumeration

Name	Value	Description
None	0	unknown
AsciiText	1	ASCII text (CR/LF)
EbcdicText	2	EBCDIC text (CR/LF)
Binary	3	binary data
AsciiData	4	ASCII data
EbcdicData	5	EBCDIC data
UnicodeText	6	Unicode text (CR/LF)
Utf8Text	7	UTF-8 text (CR/LF)
UnicodeData	8	Unicode data
BinaryCRLF	9	Binary CR/LF

ZosFileTypeClass Enumeration

Name	Value	Description
None	0	unknown
DataSet	1	data set
Unix	2	Unix file
Jes	3	JES file
ChangeMan	4	ChangeMan ZMF

ZosFreezeType Enumeration

Name	Value	Description
None	0	no options
General	1	general information
NonSource	2	non-source components

Name	Value	Description
SourceLoad	3	source/load components
Utility	4	utility information
Sites	5	sites information
Forms	6	forms information

ZosImpactRelationship Enumeration

Name	Value	Description
None	0	unknown
Copybook	1	copybook to source
Subroutine	2	linked load to composite load
JclProc	3	cataloged procedure to execution JCL
JclPgm	4	program name to JCL/procedure
JclDsn	5	data set name to JCL/procedure

ZosJobCompletionType Enumeration

Name	Value	Description
None	0	no completion info
Normal	1	job ended normally
EndComp	2	job ended with CC
JclError	3	job had a JCL error
Cancel	4	job was canceled
Abend	5	job abended
ConvAbend	6	converter abended
SecError	7	security error
EomFail	8	job failed in EOM

ZosJobHoldType Enumeration

Name	Value	Description
Unknown	0	unknown
No	1	job is not held
Yes	2	job is held
Dup	3	job is held for duplicate job name

ZosJobPhase Enumeration

Name	Value	Description
None	0	unknown
NoSubchainExists	1	No sub-chain exists
ActiveInCiFss	2	Active in CI FSS
AwaitingPostscanBatch	3	Awaiting post-scan (batch)

Name	Value	Description
AwaitingPostscanDemSel	4	Awaiting post-scan (demand select)
AwaitingVolumeFetch	5	Awaiting volume fetch
AwaitingSetup	6	Awaiting setup
MdsSystemSelectProcessing	7	MDS system select processing
AwaitingResourceAllocation	8	Awaiting resource allocation
AwaitingUnavailableVolumes	9	Awaiting unavailable volumes
AwaitingVolumeMounts	10	Awaiting volume mounts
MdsSystemVerifyProcessing	11	MDS system verify processing
ErrorDuringMdsProcessing	12	Error during MDS processing
AwaitingExecution	13	Awaiting execution
ActivelyExecuting	14	Actively executing
ActiveInOutput	17	Active in output
AwaitingMdsRestartProcess	18	Awaiting MDS restart process
MainAndMdsProcessComplete	19	Main and MDS process complete
AwaitingOutputService	20	Awaiting output service
AwaitingOutputServiceWriter	21	Awaiting output service writer
AwaitingRsvdServices	22	Awaiting rsvd services
OutputServiceComplete	23	Output service complete
AwaitingSelectionOnMain	24	Awaiting selection on main
EndingFunctionWaiting	25	Ending function waiting
EndingFunctionNotProcessed	26	Ending function not processed
ActiveInInputProcessing	128	Active in input processing
AwaitingConversion	129	Awaiting conversion
ActiveInConversion	130	Active in conversion
ActiveInSetup	131	Active in setup
ActiveInSpin	132	Active in spin
AwaitingOutput	133	Awaiting output
AwaitingPurge	134	Awaiting purge
ActiveInPurge	135	Active in purge
ActiveOnNjeSysoutReceiver	136	Active on NJE SYSOUT receiver
AwaitingNjeTransmission	137	Awaiting NJE transmission
ActiveOnNjeJobTransmitter	138	Active on NJE Job transmitter

ZosJobQueryType Enumeration

Name	Value	Description
None	0	unknown
QueueJobname	1	all jobs: job name
QueueOwner	2	all jobs: owner
ActiveJobname	3	active jobs: job name
ActiveOwner	4	active jobs: owner
ActiveAll	5	active jobs: all

ZosJobStatus Enumeration

Name	Value	Description
None	0	unknown
Wait	1	waiting for execution
Hold	2	operator hold
ExecIn	3	executing: swapped in
ExecOut	4	executing: swapped out
ExecNswp	5	executing: non-swappable
NjeActive	6	active in NJE
Output	7	output queue

ZosJobType Enumeration

Name	Value	Description
None	0	unknown
Stc	1	started task
Tsu	2	TSO user
Job	3	batch job
Appc	4	APPC transaction

ZosLibType Enumeration

Name	Value	Description
Std	0	PDS/PDSE
Lib	1	Librarian
Pan	2	Panvalet

ZosLikeType Enumeration

Name	Value	Description
None	0	like none
Copy	1	like copy
Load	2	like load
Other	3	like other
Pds	4	like PDS
Source	5	like source

ZosOutputQueue Enumeration

Name	Value	Description
Unknown	0	unknown
Wtr	1	writer queue
Hold	2	hold queue
XWtr	3	external writer hold queue

ZosPackageApprovalAction Enumeration

Name	Value	Description
None	0	unknown
Approve	1	approve package
CheckOff	2	check off
Pending	3	decision pending
Reject	4	reject package
Review	5	under review
Final	6	final approval for linked packages

ZosPackageLevel Enumeration

Name	Value	Description
None	0	unknown
Simple	1	simple
Complex	2	complex
Super	3	super
Participating	4	participating
Other	5	other

ZosPackageLevelFlags Enumeration (Flags)

Name	Value	Description
Any	0x00	any level
Simple	0x02	simple
Complex	0x04	complex
Super	0x08	super
Participating	0x10	participating

ZosPackagePromotionAction

Name	Value	Description
Any	0x00	any action
FirstPromote	0x01	first promotion
FullPromote	0x02	full promotion
FullDemote	0x04	full demotion
SelPromote	0x08	selective promotion
SelDemote	0x10	selective demotion

ZosPackagePromotionStatus

Name	Value	Description
Any	0x00	any status
Built	0x01	job built
Submitted	0x02	job submitted
Completed	0x04	job completed
Failed	0x08	job failed

ZosPackageStatus Enumeration

Name	Value	Description
None	0	unknown
Approved	1	approved
BackOut	2	back out
Baseline	3	baseline
Closed	4	closed
Deleted	5	deleted
Development	6	development

Name	Value	Description
Distributed	7	distributed
Frozen	8	frozen
Installed	9	installed
Open	10	open
Rejected	11	rejected
Tcc	12	temporary change cycled
Other	13	other

ZosPackageStatusFlags Enumeration (Flags)

Name	Value	Description
Any	`0x0000`	any status
Approved	`0x0002`	approved
BackOut	`0x0004`	back out
Baseline	`0x0008`	baseline
Closed	`0x0010`	closed
Deleted	`0x0020`	deleted
Development	`0x0040`	development
Distributed	`0x0080`	distributed
Frozen	`0x0100`	frozen
Installed	`0x0200`	installed
Open	`0x0400`	open
Rejected	`0x0800`	rejected
Tcc	`0x1000`	temporary change cycled

ZosPackageType Enumeration

Name	Value	Description
None	`0`	unknown
PlannedPerm	`1`	planned permanent
PlannedTemp	`2`	planned temporary
UnplannedPerm	`3`	unplanned permanent
UnplannedTemp	`4`	unplanned temporary
Other	`5`	other

ZosPackageTypeFlags Enumeration (Flags)

Name	Value	Description
Any	`0x00`	any type
PlannedPerm	`0x02`	planned permanent
PlannedTemp	`0x04`	planned temporary
UnplannedPerm	`0x08`	unplanned permanent
UnplannedTemp	`0x10`	unplanned temporary

ZosProblemActionType Enumeration

Name	Value	Description
None	0	unknown
Hold	1	hold production
Backout	2	back out change
Other	3	other instructions

ZosPromotionOverlayStatus Enumeration

Name	Value	Description
None	0	no status
NoHistory	1	exists in promotion library, but no history
History	2	not in library, but exists in history
Common	3	common to both promotion library and history

ZosPromotionTarget Enumeration

Name	Value	Description
Shadow	0	shadow
Library1	1	library 1
Library2	2	library 2
Library3	3	library 3

ZosRecordFormat Enumeration

Name	Value	Description
None	0x00	unknown
F	0x80	fixed length
V	0x40	variable length
U	0xC0	undefined length
B	0x10	blocked
S	0x08	spanned
A	0x04	ANSI carriage control
M	0x02	machine carriage
FA	0x84	fixed, ANSI cc
VA	0x44	variable, ANSI cc

Name	Value	Description
FM	`0x82`	fixed, machine cc
VM	`0x42`	variable, machine cc
FB	`0x90`	fixed block
VB	`0x50`	variable block
FBA	`0x94`	fixed block, ANSI cc
VBA	`0x54`	variable block, ANSI cc
FBM	`0x92`	fixed block, machine cc
VBM	`0x52`	variable block, machine cc
FBS	`0x98`	fixed block spanned
VBS	`0x58`	variable block spanned
FBSA	`0x9C`	fixed block spanned, ANSI cc
VBSA	`0x5C`	variable block spanned, ANSI cc
FBSM	`0x9A`	fixed block spanned, machine cc
VBSM	`0x5A`	variable block spanned, machine cc

ZosReleaseApprovalAction Enumeration

Name	Value	Description
None	0	unknown
Approve	1	approve release
Reject	2	reject release

ZosReleaseApprovalType Enumeration

Name	Value	Description
Release	0	release
CheckIn	1	check in
CheckOff	2	check off

ZosReleaseAreaStatus Enumeration (Flags)

Name	Value	Description
None	0x0000	no status
Blocked	0x0001	area blocked
AprCheckIn	0x0002	area check in approved
AprCheckOff	0x0004	area check off approved
RejCheckIn	0x0008	area check in rejected
RejCheckOff	0x0010	area check off rejected
BlockPend	0x0020	area block pending
CheckInPend	0x0040	area check in pending
ApprovalPend	0x0080	area approval pending
AuditPend	0x0100	area audit pending

ZosReleaseAreaType Enumeration

Name	Value	Description
Subsystem	0	subsystem
System	1	system

ZosReleaseStatus Enumeration

Name	Value	Description
None	0	unknown
Approved	1	approved
Backout	2	back out
Baseline	3	baseline
Blocked	4	blocked
Deleted	5	deleted
Development	6	development
Distributed	7	distributed
Installed	8	installed
Rejected	9	rejected

ZosSchedulerType Enumeration

Name	Value	Description
None	0	unknown
ChangeMan	1	ChangeMan scheduler
Manual	2	manual scheduling
Other	3	other scheduler

ZosSpaceUnit Enumeration

Name	Value	Description
None	0	unknown
Cyl	1	cylinders
Trk	2	tracks
Blk	3	blocks

ZosStagingVersionLocation Enumeration

Name	Value	Description
None	0	unknown
Staging	2	staging
Baseline	4	baseline
Backup	6	backup

ZosStagingVersSaveOption Enumeration

Name	Value	Description
None	0	none
Prompt	1	prompt
Always	2	always

ZosUnixAccess Enumeration (Flags)

Name	Value	Description
None	0x00	no access
Execute	0x01	execute access permission
Write	0x02	write access permission
Read	0x04	read access permission

ZosUnixAccessCheck Enumeration (Flags)

Name	Value	Description
None	0x00	not specified
Execute	0x01	check for execute access
Write	0x02	check for write access
Read	0x04	check for read access
Exists	0x08	check for file existence

ZosUnixFileFormat Enumeration

Name	Value	Description
None	0	not specified
Bin	1	binary data
Nl	2	NL (new line)
Cr	3	CR (carriage return)
Lf	4	LF (line feed)
CrLf	5	CR & LF
LfCr	6	LF & CR
CrNl	7	CR & NL

ZosUnixFileType Enumeration

Name	Value	Description
None	0	unknown
Directory	1	directory
CharSpecial	2	character device (not used in ZDD Network)
File	3	regular file
Fifo	4	named pipe (not used in ZDD Network)
SymLink	5	symbolic link

Chapter 3

Class Reference

This chapter describes the properties and methods for each ChangeMan ZDD object. Examples are shown in C#, C++, Visual Basic, and JScript, although you may use any language that supports .NET.

ZosApplication

The **ZosApplication** object represents a ChangeMan ZMF application. This object can be obtained using either the **GetApplication** method or the **GetApplications** method of **ZosChangeManInstance**.

ZosApplication Properties

ZosApplication exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the application.
Path	String	R	Full path name of the application.
ChangeManInstance	ZosChangeManInstance	R	Parent ChangeMan instance for this application.
Description	String	R	Description of the application.
BaselineMemberFilters	ZosNameFilters	R	Collection of member name filters for baseline libraries.
PackageMemberFilters	ZosNameFilters	R	Collection of member name filters for package libraries.
PackageMemberFilters **PromotionMemberFilters**	ZosNameFilters4	R	Collection of member name filters for promotion libraries.

ZosApplication Methods

ZosApplication exposes the following methods:

Function	**Description**
`ZosBaselineLibrary GetBaselineLibrary( String libType )`	Gets a single baseline library by name.
`ZosBaselineLibrary[] GetBaselineLibraries()`	Gets an array containing the baseline libraries for the application.
`String[] GetSiteNames( Boolean ipOnly [optional] )`	Gets an array containing the site names defined for an application. ipOnly – Requests only sites that have an IP address defined
`ZosPromotionSite GetPromotionSite( String siteName )`	Gets a single promotion site by name.
`ZosPromotionSite[] GetPromotionSites()`	Gets an array containing the promotion sites for the application.
`ZosPromotionLevel GetPromotionLevel( String siteName, String promotionName )` `ZosPromotionLevel GetPromotionLevel, String siteName, Int16 promotionLevel )`	Gets a promotion level, given the site name and promotion name or level number.

```
ZosComponentHistory[]
    GetComponentHistory(
        String componentType,         [opt]
        String componentName,         [opt]
        ZosComponentHistoryType type, [opt]
        String package,               [opt]
        ZosComponentHistoryStatus flags,
        DateTime fromChangeDate,      [opt]
        DateTime toChangeDate,        [opt]
        DateTime fromBaselineDate,    [opt]
        DateTime toBaselineDat        [opt]
        )
```
	Gets a list of component history records for a given component.
	
	All arguments are optional.
	
	componentType: Component type filter
	componentName: Component name filter
	historyType: Indicates type of history list to be returned
	package: Package name filter
	statusFlags: History status flags filter
	fromChangeDate: Components changed after this date
	toChangeDate: Components changed before this date
	fromBaselineDate: Packages baselined after this date
	toBaselineDate: Packages baselined before this date

```
ZosPackage
    GetPackage(
        String packageName
        )
```
	Gets a single package by name.

`ZosPackage[]`
`GetPackages()`
 	
`ZosPackages[]`
`GetPackage(`
`ZosPackageLevelFlags levelFlags,`
`ZosPackageTypeFlags typeFlags,`
`ZosPackageStatusFlags statusFlags`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`String[] departments`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`DateTime minInstallDate,`
`DateTime maxInstallDate`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`Int32 minPackageNumber,`
`Int32 maxPackageNumber`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`ZosPackageLevelFlags levelFlags,`
`ZosPackageTypeFlags typeFlags,`
`ZosPackageStatusFlags statusFlags,`
`String[] departments`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`ZosPackageLevelFlags levelFlags,`
`ZosPackageTypeFlags typeFlags,`
`ZosPackageStatusFlags statusFlags,`
`String[] departments,`
`DateTime minInstallDate,`
`DateTime maxInstallDate`
`)`
 	
`ZosPackages[]`
`GetPackage(`
`ZosPackageLevelFlags levelFlags,`
`ZosPackageTypeFlags typeFlags,`
`ZosPackageStatusFlags statusFlags,`
`String[] departments,`
`DateTime minInstallDate,`
`DateTime maxInstallDate,`
`Int32 minPackageNumber,`
`Int32 maxPackageNumber`
`)`
	Gets an array of packages.
	
	The applications can optionally be filtered by package levels, package types, package status, department numbers, install date range, or package number range.
	
	If filtering by department number, the department numbers in the list can contain wild characters. See the section on wild characters for details.

ZosApplication Examples

Examples of using **ZosApplication** are shown below:

C#
```
ZosApplication app;
ZosPackage package = app.GetPackage("TEST000123");
ZosPackage[] packages = app.GetPackages();
```

C++
```
ZosApplication^ app;
ZosPackage package = app.GetPackage("TEST000123");
array<ZosPackage^>^ packages = app.GetPackages();
```

Visual Basic
```
Dim app as ZosApplication;
Dim package As ZosPackage = app.GetPackage("TEST000123")
Dim packages() As ZosPackage = app.GetPackages()
```

Jscript
```
var app : ZosApplication;
var package : ZosPackage app = app.GetPackage("TEST000123");
var packages : ZosPackage [] = app.GetPackages();
```

ZosBaselineLibrary

The **ZosBaselineLibrary** object represents a ChangeMan baseline library for an application. This object can be obtained using the **GetBaselineLibrary** or **GetBaselineLibraries** methods of **ZosApplication**.

ZosBaselineLibrary Properties

ZosBaselineLibrary exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Library type name
Path	String	R	Full file system path name for the library
Description	String	R	Library description
IsUnix	Boolean	R	Indicates whether a library is a PDS or Unix directory
DataSetName	String	R	Data set name for the library
TargetLibrary	String	R	Target build library
LikeType	ZosLikeType	R	Like library type option

StagingVersSaveOption	ZosStagingVersSaveOption	R	Staging version save option
DeferredAllocation	Boolean	R	Indicates whether allocations are deferred
DataSetType	ZosDataSetType	R	Data set type (organization)
RecordFormat	ZosRecordFormat	R	Record format
RecordLength	Int16	R	Record length
BlockSize	Int16	R	Block size
SpaceUnit	ZosSpaceUnit	R	Space unit type
PrimarySpace	Int32	R	Primary space quantity
SecondarySpace	Int32	R	Secondary space quantity
DirectoryBlocks	Int32	R	Number of directory blocks
UnitName	String	R	Unit name
Volume	String	R	Volume serial number

ZosBaselineLibrary Methods

ZosBaselineLibrary exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the library information.

`ZosPdsMember`
`    GetPdsComponent(`
`        String name`
`        )`
	Gets a single component of a baseline PDS library by name. Component name can be specified with or without an extension.

`ZosPdsMember[]`
`GetPdsComponents()`
 	
`ZosPdsMember[]`
`GetPdsComponents(`
`String nameFilter`
`)`
 	
`ZosPdsMember[]`
`GetPdsComponents(`
`DateTime changeTime`
`)`
 	
`ZosPdsMember[]`
`GetPdsComponents(`
`String nameFilter,`
`DateTime changeTime`
`)`
	Gets an array of components that belong to a baseline PDS library. The list can optionally be filtered by component name.
	
	nameFilter - Componenent name filter (pattern)
	
	changeTime - get components changed after the specified time

`ZosUnixObject[]`
`GetUnixComponent(`
`String name`
`)`
	Gets a single component of a baseline Unix library by file name.

`ZosUnixObject[]`
`GetUnixComponents()`
 	
`ZosUnixObject[]`
`GetUnixComponents(`
`DateTime changeTime`
`)`
 	
`ZosUnixObject[]`
`GetUnixComponents(`
`String dirName`
`)`
 	
`ZosUnixObject[]`
`GetUnixComponents(`
`String dirName,`
`String nameFilter`
`)`
 	
`ZosUnixObject[]`
`GetUnixComponents(`
`String dirName,`
`String nameFilter,`
`DateTime changeTime`
`)`
	Gets an array of components that belong to a baseline Unix library. The list can optionally be filtered by component name.
	
	For Unix libraries, components are retrieved hierarchically. This function only returns components in a specified subdirectory. The array returned contains both directory and file objects.
	
	dirName - Subdirectory name
	
	nameFilter - Componenent name filter (pattern)
	
	changeTime - get components changed after the specified time

ZosBuildInfo

The **ZosBuildInfo** object represents a set of build properties that can be used to build, recompile, or relink a component in a package.

An empty **ZosBuildInfo** object can be created using the default constructor. You can then set the desired **ZosBuildInfo** properties before using it to build a component.

You can clone the build information from designated compile procedures or component history using alternate forms of the constructor. This cloned **ZosBuildInfo** object can be used to build components after making any desired changes to its properties.

ZosBuildInfo is used as input to the **Build**, **Recompile**, and **Relink** methods of **ZosPackage**.

ZosBuildInfo Constructor

The default constructor can be used to create a new **ZosBuildInfo** object. Because the constructor has no arguments, you must initialize the object by setting its properties. The other constructor copies the properties from an existing component based upon designated compile procedures or history.

Constructor	Parameters
`ZosBuildInfo()`	

`ZosBuildInfo(`
`    ZosPackage package,`
`    String componentName,`
`    String componentType`
`    )`
	Package and component from which to copy properties.

`ZosBuildInfo(`
`    ZosPackageComponentFile`
`component`
`    )`
	Package component from which to copy properties.

ZosBuildInfo Properties

ZosBuildInfo exposes the following properties:

Property	Type	R/W	Description
Language	String	R/W	Component language.
BuildProc	String	R/W	Build procedure name.
CompileParm	String	R/W	Compile parameters.
LinkParm	String	R/W	Link parameters.
UseHistory	Boolean	R/W	Indicates that history should override options specified here.
Db2Precompile	Boolean	R/W	DB2 pre-compile indicator.
Db2Subsystem	String	R/W	DB2 subsystem name.
Db2LinkLib	String	R/W	DB2 link library data set name.

Db2Version	String	R/W	DB2 version ID.
UserOptions	ZosNameValue[]	R/W	User options. Each user option is a name/value pair. See table below.
UserVariables	ZosNameValue[]	R/W	User variables. Each user variable is a name/value pair. See table below.

UserOptions

User options are a set of name/value pairs. Each name must be one of the names in the following table:

Variable Name	Value Length
UserOption01 - UserOption20	1
UserOption0101 - UserOption0105	1
UserOption0201 - UserOption0203	2
UserOption0301 - UserOption0303	3
UserOption0401 - UserOption0403	4
UserOption0801 - UserOption0805	8
UserOption1001 - UserOption1002	10
UserOption1601 - UserOption1602	16
UserOption3401 - UserOption3402	34
UserOption4401 - UserOption4402	44
UserOption6401 - UserOption6405	64
UserOption7201 - UserOption7205	72

UserVariables

User variables are a set of name/value pairs. Each name must be one of the names in the following table:

Variable Name	Value Length
UserVariable01 - UserVariable05	8
UserVariable06 - UserVariable10	72

ZosChangeManInstance

The **ZosChangeManInstance** object represents a single ChangeMan instance on the server. This object can be obtained using the **ChangeManInstance** property of **ZosServer** or the **Item** property of **ZosChangeManInstances**.

ZosChangeManInstance Properties

ZosChangeManInstance exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the ChangeMan instance.
Path	String	R	Full path name of the ChangeMan instance.
Server	ZosServer	R	Parent server for this ChangeMan instance.
Port	Int32	R/W	I/P port number for ChangeMan instance.
UtcOffset	TimeSpan	R	Time zone offset from UTC. UTC + UtcOffset = Local
Today	DateTime	R	Current date on the server.
Description	String	R/W	ChangeMan description.
EnvironmentType	ZosEnvironmentType	R	ChangeMan environment type.
Filters	ZosNameFilters	R	Collection of all application name filters for folder. Default is all applications.
FileFormatMappings	ZosileFormatMappings	R	Collection of file format mappings for server.
EroLicensed	Boolean	R	Indicates whether ERO is licensed
EroSupported	Boolean	R	Indicates whether ERO functions are supported

ZosChangeManInstance Methods

ZosChangeManInstance exposes the following methods:

Function	Description

`void`
`    SubmitXml(`
`        String inputFileName,`
`        String outputFileName`
`        )`
	Submit XML request to ChangeMan.

`ZosApplication`
`    GetApplication(`
`        String appName`
`        )`
	Gets a single application by name.

`ZosApplication[]`
`    GetApplications(`
`       Boolean includeHidden [optional]`
`         )`
 	
`ZosApplication[]`
`    GetApplications(`
`       String nameFilters,   [optional]`
`       Boolean includeHidden [optional]`
`         )`
	Gets an array of applications. The applications can optionally be filtered by name. The filer string can contain '*' and '?' wild characters. Multiple filters can be specified, separated by spaces.
	
	Hidden applications can optionally be included.

`ZosRelease[]`
`    GetReleases(`
`       Boolean includeHidden [optional]`
`         )`
 	
`ZosRelease[]`
`    GetReleases(`
`       String nameFilters,   [optional]`
`       Boolean includeHidden [optional]`
`         )`
	Gets an array of releases. The releases can optionally be filtered by name. The filer string can contain '*' and '?' wild characters. Multiple filters can be specified, separated by spaces.
	
	Hidden releases can optionally be included.

```
ZosPackage[]
    GetPackages(
        String
            nameFilters,    [optional]
        ZosPackageLevelFlags
            levels,         [optional]
        ZosPackageTypeFlags
            types,          [optional]
        ZosPackageStatusFlags
            status,         [optional]
        DateTime
            minInstallDate, [optional]
        DateTime
            maxInstallDate, [optional]
        String
            department,     [optional]
        String
            release,        [optional]
        String
            promotionSite,  [optional]
        String
            promotionName   [optional]
        )
```
	Gets an array of packages that match search arguments...
	
	The packages can be filtered by package name, package levels, package types, package status, install date range, department, release, or promotion location.
	
	The package name filter contains one or more packages names, separated by semicolons. Each package name can be masked using the asterisk (*) wild character.
	
	Example: JOHN*;MARY*;JUDY*

```
ZosQueryImpactResult[]
    QueryImpact(
        ZosImpactRelationship rel,
        String topComponent,
        String topApp,      [optional]
        String topLibType   [optional]
        )

ZosQueryImpactResult[]
    QueryImpact(
        ZosImpactRelationship rel,
          String topComponent, [opt]
          String topApp,       [opt]
          String topLibType,   [opt]
          String bottomComponent,
          String bottomApp,    [opt]
          String bottomLibType [opt]
          )

ZosQueryImpactResult[]
    QueryImpact(
        ZosImpactRelationship rel,
        String topComponent,
        UInt32 topBun
        )

ZosQueryImpactResult[]
    QueryImpact(
        ZosImpactRelationship rel,
        String topComponent,
        String topApp,
        String topLibType,
        String bottomComponent,
        String bottomApp,
        String bottomLibType,
        UInt32 topBun
        )
```
	Performs an impact analysis query.

ZosChangeManInstance Examples

Examples of using **ZosChangeManInstance** are shown below:

C#

```
ZosChangeManInstance changeman;
changeman.SubmitXml(“C:\XML\Request.xml”, “C:\XML\Response.xml”);
ZosApplication app = changeman.GetApplication(“DEMO”);
ZosApplication[] apps = changeman.GetApplications();
ZosApplication[] apps = changeman.GetApplications(“A* B*”);
```

C++

```
ZosChangeManInstance^ changeman;
changeman->SubmitXml(“C:\XML\Request.xml”, “C:\XML\Response.xml”);
ZosApplication app = changeman.GetApplication(“DEMO”);
array<ZosApplication^>^ apps = changeman.GetApplications();
array<ZosApplication^>^ apps = changeman.GetApplications(“A* B*”);
```

Visual Basic
```
Dim changeman As ZosChangeManInstance;
changeman.SubmitXml(“C:\XML\Request.xml”, “C:\XML\Response.xml”)
Dim app As ZosApplication = changeman.GetApplication(“DEMO”)
Dim apps() As ZosApplication = changeman.GetApplications()
Dim apps() As ZosApplication = changeman.GetApplications(“A* B*”)
```

Jscript
```
var changeman : ZosChangeManInstance;
changeman.SubmitXml(“C:\XML\Request.xml”, “C:\XML\Response.xml”);
var app : ZosApplication app = changeman.GetApplication(“DEMO”);
var apps : ZosApplication[] = changeman.GetApplications();
var apps : ZosApplication[] = changeman.GetApplications(“A* B*”);
```

ZosChangeManInstances

The **ZosChangeManInstances** object is a collection of all ChangeMan instances on a server. This object is obtained using the **ChangeManInstances** property of the **ZosServer** object.

ZosChangeManInstances Properties

ZosChangeManInstances exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosChangeManInstance	R	Folder with specified name or index.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosChangeManInstances Methods

ZosChangeManInstances exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`Int32 Add( String name, Int32 port, String description [optional] )`	Adds a new ChangeMan instance. Returns index at which object has been added.

`Boolean`
`    Remove(`
`        String name`
`        )`
	Deletes a ChangeMan instance. Returns true if instance was removed or false if name is not found.

`Int32`
`    FindIndex(`
`        String name`
`        )`
	Searches for ChangeMan instance with specified name and returns zero-based index. Returns -1 if name is not found.

`ZosChangeManInstance`
`    Find(`
`        String name`
`        )`
	Searches for ChangeMan instance with specified name and returns reference to object. Returns null if name is not found.

ZosCheckInStatus

The **ZosCheckInStatus** object shows status information for a release check in operation for a particular component.

The check in status is returned by the **ReleaseCheckIn** method of **ZosPackage** and by the **CheckIn** method of **ZosReleaseArea**.

ZosCheckInStatus Properties

ZosCheckInStatus exposes the following properties:

Property	Type	R/W	Description
Release	String	R	Release name.
ReleaseArea	String	R	Release area name.
Package	String	R	Package name.
ComponentName	String	R	Target component name.
ComponentType	String	R	Component type (library type).
User	String	R	User ID who last updated the component.
CheckInTime	DateTime	R	Date and time component was checked in.
Status	String	R	Status description.

ZosComponentHistory

The **ZosComponentHistory** object is a ChangeMan component general history record.

The component history can be retrieved using the **GetComponentHistory** method of **ZosApplication**.

ZosComponentHistory Properties

ZosComponentHistory exposes the following properties:

Property	Type	R/W	Description
Package	String	R	Package name
ComponentType	String	R	Component type
ComponentName	String	R	Component name
Version	Int16	R	Version number
ModLevel	Int16	R	Modification level
User	String	R	User who last updated component
BuildProc	String	R	Build procedure
PromotionName	String	R	Promotion level name
PromotionLevel	Int16	R	Promotion level number
UpdateTime	DateTime	R	Date and time of update
Status	ZosComponentHistoryStatus	R	Component history status flags

ZosComponentPromotionHistory

The **ZosComponentPromotionHistory** object is a ChangeMan component promotion history record. The component promotion history records represent a component promotion event.

The component promotion history can be retrieved using the **GetComponentPromotionHistory** method of **ZosPackage**.

ZosComponentPromotionHistory Properties

ZosComponentPromotionHistory exposes the following properties:

Property	Type	R/W	Description
PromotionSite	String	R	Promotion site name
PromotionName	String	R	Promotion level name
PromotionLevel	Int16	R	Promotion level number
ComponentType	String	R	Component type
ComponentName	String	R	Component name
PromotionUser	String	R	Promoting user ID
PromotionTime	DateTime	R	Date and time of the promotion
Status	ZosComponentPromotionStatus	R	Promotion status flags

ZosComponentStagingVersion

The **ZosComponentStagingVersion** object represents a staging version of a ChangeMan package component.

The staging versions for a component can be retrieved using the **GetStagingVersions** method of **ZosPackageComponentFile**.

ZosComponentStagingVersions Properties

ZosComponentStagingVersions exposes the following properties:

Property	Type	R/W	Description
Path	String	R	Full file system path name for the component.
VersionIndex	UInt16	R	Version index (sequence number).
VersionStamp	UInt32	R	Version stamp (unique ID).
LineCount	UInt32	R	Number of lines after update.
User	String	R	User ID.
ChangeDescription	String	R	Change description.
ChangeTime	DateTime	R	Date and time component was updated.
Location	ZosStagingVersionLocation	R	Code for library where this version of the component resides.

ZosConnectionLock

In a server application, there may be a requirement to have more than one user ID logged onto the same server at the same time. You can accomplish this by using alternate connections to the server. Each server can have alternate connections, with connection IDs numbered 1 – 255. The default connection has a connection ID of 0.

The **ZosConnectionLock** class can be used to reserve a connection ID, and lock the connection ID so that it will not be used by other programs or threads. The default connection ID, 0, will never be locked.

With **ZosConnectionLock** you can either implicitly lock a connection ID via the constructor or you can explicitly lock a connection ID by calling the **Lock** method.

You must unlock the connection ID by calling either the **Unlock** or **Dispose** method of **ZosConnectionLock**. With C# and Visual Basic, you can have the connection automatically unlocked by using a **using** statement. With C++, you can have the connection automatically unlocked by declaring the **ZosConnectionLock** object as a stack variable.

If the connection is not automatically unlocked, then you should ensure that the connection gets unlocked, by explicitly unlocking it in the **finally** block of a **try** / **finally** construction.

For more information on the usage of **ZosConnectionLock**, see the section entitled "Alternate Connections" on page 34.

ZosConnectionLock Constructor

The following constructor can be used to initialize a new **ZosConnectionLock** object:

Constructor	Parameters

`ZosConnectionLock(`
`    String name,`
`    Boolean locked` [optional]
`    )`
	name: Server name for which a connection is to be locked
	locked: Indicates connection is to be initially locked

ZosConnectionLock Properties

ZosConnectionLock exposes the properties below. All properties are read only.

Property	Type	R/W	Description
Name	String	R	Name of the server for which a connection is to be locked.
Connection	Int16	R	Connection ID that has been locked or zero if no connection is locked.

ZosConnectionLock Methods

ZosConnectionLock exposes the following methods:

Function	Description

`Int16`
`    Lock()`
	Obtains and locks a connection ID. Returns the connection ID.

`void`
`    Unlock()`
	Releases the connection ID that was previously locked.

`void`
`    Dispose()`
	Releases the connection ID that was previously locked and destroys the lock. Object cannot be used again after calling Dispose().

ZosConnectionLock Examples

Examples of using **ZosConnectionLock** are shown in the section entitled "Alternate Connections" on page 34.

ZosDataSet

The **ZosDataSet** object represents a data set in the server. This object can be obtained using the **GetDataSet** method of **ZosServer** or the **GetDataSets** method of **ZosDataSetFolder**.

ZosDataSet Properties

ZosDataSet exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Data set name.
Path	String	R	Full path name of the data set.
DataSetType	ZosDataSetType	R	Data set type (organization)
RecordFormat	ZosRecordFormat	R	Record format
RecordLength	Int16	R	Logical record length
BlockSize	Int16	R	Block size
DataClass	String	R	Data class
StorageClass	String	R	Storage class
ManagementClass	String	R	Management class
ExtendedAttributes	ZosDataSetEAttr	R	Extended attributes: Default, No, Opt
UnitName	String	R	Unit name
Volume	String	R	Volume serial number
SpaceUnit	ZosSpaceUnit	R	Space unit type
PrimarySpace	Int32	R	Primary space allocation
SecondarySpace	Int32	R	Secondary space allocation
Extents	Int32	R	Number of extents allocated
AllocatedTracks	Int32	R	Number of tracks allocated
UsedTracks	Int32	R	Number of tracks used
UsedPercent	Int16	R	Percent of space used
DirectoryBlocks	Int32	R	Number of directory blocks
UsedDirectoryBlocks	Int32	R	Number of directory blocks used
Members	Int32	R	Number of members
CreationDate	DateTime	R	Creation date
LastAccessDate	DateTime	R	Last access date
PdseVersion	Int16	R	PDSE version number: 0 (default), 1, 2

MaxGens	Int32	R	Maximum number of PDSE member generations
Encrypted	Boolean	R	Indicates data set is encrypted
JobName	String	R	Job name used to create data set (extended attribute)
StepName	String	R	Step name used to create data set (extended attribute)

ZosDataSet Methods

ZosDataSet exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the data set information.
`ZosDataSetInfo GetInfo()`	Gets a data set information object containing the data set information. This object can be used to create a new data set modeled after this data set.
`ZosPdsMember GetMember( String name )`	Gets a single member of a partitioned data set by member name.
`ZosPdsMember[] GetMembers()` `ZosPdsMember[] GetMembers( String nameFilter )` `ZosPdsMember[] GetMembers( DateTime changeTime )` `ZosPdsMember[] GetMembers( String nameFilter, DateTime changeTime )`	Gets an array of members that belong to a partitioned data set. The list can optionally be filtered. nameFilter - Componenent name filter (pattern) changeTime - get members changed after the specified time
`void Delete()`	Deletes the data set.
`void Rename( String newName )`	Renames the data set.

`void`
`    Compress()`
	Compresses a partitioned data set.

`void`
`    Migrate()`
	HSM migrates a data set.

`void`
`    Recall()`
	HSM recalls a data set.

`void`
`    CopyTo(`
`        ZosDataSet dataset`
`        )`
 	
`void`
`    CopyTo(`
`        ZosDataSet dataset,`
`        String member`
`        )`
 	
`void`
`    CopyTo(`
`        ZosDataSet dataset,`
`        String[] members`
`        )`
	Copy this data set to another data set.
	
	For partitioned data sets, you can copy the full data set or selected members.
	
	Member names can be specified using wild characters.

`Static`
`ZosDataSet`
`    Create(`
`        ZosServer server,`
`        String dsname,`
`        ZosDataSetInfo info`
`        )`
	Create a new data set on the server. The data set attributes are specified using a ZosDataSetInfo object.
	
	Returns a data set object representing the new data set.

`static`
`ZosRecordFormat`
`    StringToRecordFormat(`
`        String^ text`
`        )`
	Converts a character string representation of a record format to a ZosRecordFormat enumeration.

`static`
`String^`
`    RecordFormatToString(`
`        ZosRecordFormat recfm`
`        )`
	Formats a ZosRecordFormat enumeration into a display string.

ZosDataSet Examples

Examples of using **ZosDataSet** are shown below:

C#

```
ZosDataSet dataset;
ZosDataSet dataset2;
ZosRecordFormat recfmt;
String text;
recfmt = ZodDataSet.StringToRecordFormat(“FB”);
text = ZosDataSet.RecordFormatToString(recfm);
ZosPdsMember member = dataset.GetMember(“JUNK”);
ZosPdsMembers members = dataset.GetMembers(“X*”);
ZosDataSetInfo info = dataset.GetInfo();
info.DataSetType = ZosDataSetType.Pdse;
dataset2 = ZosDataSet.Create(server, “NEW.DATA.SET”, info);
```

C++

```
ZosDataSet^ dataset;
ZosDataSet^ dataset2;
ZosRecordFormat recfmt;
String^ text;
recfmt = ZodDataSet::StringToRecordFormat(“FB”);
text = ZosDataSet::RecordFormatToString(recfm);
ZosPdsMember^ member = dataset.GetMember(“JUNK”);
array<ZosPdsMember^>^ members = dataset.GetMembers(“X*”);
ZosDataSetInfo^ info = dataset.GetInfo();
info.DataSetType = ZosDataSetType::Pdse;
dataset2 = ZosDataSet::Create(server, “NEW.DATA.SET”, info);
```

Visual Basic

```
Dim dataset As ZosDataSet
Dim dataset2 As ZosDataSet
Dim recmt As ZosRecordFormat
Dim text As String
recfmt = ZodDataSet.StringToRecordFormat(“FB”)
text = ZosDataSet.RecordFormatToString(recfm)
Dim member As ZosPdsMember = dataset.GetMember(“JUNK”)
Dim members() As ZosPdsMember = dataset.GetMembers(“X*”)
Dim info As ZosDataSetInfo = dataset.GetInfo()
info.DataSetType = ZosDataSetType.Pdse
dataset2 = ZosDataSet.Create(server, “NEW.DATA.SET”, info)
```

Jscript

```
var dataset : ZosDataSet;
var dataset2 : ZosDataSet;
var recmt : ZosRecordFormat;
var text : String;
recfmt = ZodDataSet.StringToRecordFormat(“FB”);
text = ZosDataSet.RecordFormatToString(recfm);
var member : ZosPdsMember = dataset.GetMember(“JUNK”);
var members : ZosPdsMembers[] = dataset.GetMembers(“X*”);
var info : ZosDatSetInfo = dataset.GetInfo();
info.DataSetType = ZosDataSetType.Pdse;
dataset2 = ZosDataSet.Create(server, “NEW.DATA.SET”, info);
```

ZosDataSetFolder

The **ZosDataSetFolder** object represents a single data set folder. This object can be obtained using the Item property of **ZosDataSetFolders**.

ZosDataSetFolder Properties

ZosDataSetFolder exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the folder.
Path	String	R	Full path name of the folder.
Subfolders	ZosDataSetFolders	R	Collection of all subfolders for this folder.
Filters	ZosNameFilters	R	Collection of all data set name filters for folder.
MemberFilters	ZosNameFilters	R	Collection of member name filters for libraries under folder.
PrefixMappings	ZosPrefixMappings	R	Collection of all data set name prefixes for folder.

ZosDataSetFolder Methods

ZosDataSetFolder exposes the following methods:

Function	Description
`ZosDataSet[] GetDataSets()`	Gets an array of data sets that match the filters for this folder.

ZosDataSetFolders

The **ZosDataSetFolders** object is a collection of all data set folders with the same parent folder. This object is obtained using the **DataSetFolders** property of **ZosServer** or the **Subfolders** property of the **ZosDataSetFolder** object.

ZosDataSetFolders Properties

ZosDataSetFolders exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosDataSetFolder	R	Folder with specified name or index.

Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosDataSetFolders Methods

ZosDataSetFolders exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`Int32 Add( String folderName )`	Adds a new folder. Returns index at which object has been added.
`Boolean Remove( String folderName )`	Deletes a folder. Returns true if folder was removed or false if folder is not found.
`Int32 FindIndex( String name )`	Searches for folder with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosDataSetFolder Find( String name )`	Searches for folder with specified name and returns reference to object. Returns null if name is not found.

ZosDataSetInfo

The **ZosDataSetInfo** object represents a set of data set properties that can be used to create a new data set.

An empty **ZosDataSetInfo** object can be created using the default constructor. You can then set the desired **ZosDataSetInfo** properties, before using it, to create a new data set.

You can clone the properties of another data set using the **GetInfo** method of **ZosDataSet** to copy the properties of an existing data set into a new **ZosDataInfo** object. There is also one form of the **ZosDataSetInfo** constructor that initializes the properties from an existing data set. This cloned **ZosDataSetInfo** object can be used to create a new data set after making any desired changes to its properties.

ZosDataSetInfo Constructor

The default constructor can be used to create a new **ZosDataSetInfo** object. Because that constructor has no arguments, you must initialize the object by setting its properties. The other constructor copies the properties from an existing data set.

Constructor	Parameters
`ZosDataSetInfo()`	
`ZosDataSetInfo( ZosDataSet dataset )`	Existing data set from which to copy properties.

ZosDataSetInfo Properties

ZosDataSetInfo exposes the properties below. All properties are read / write.

Property	Type	R/W	Description
DataSetType	ZosDataSetType	R/W	Data set type: Seq, Pds, Pdse
RecordFormat	ZosRecordFormat	R/W	Record format.
RecordLength	Int16	R/W	Record length.
BlockSize	Int16	R/W	Block size.
DataClass	String	R/W	SMS data class.
StorageClass	String	R/W	SMS storage class.
ManagementClass	String	R/W	SMS management class.
UnitName	String	R/W	Unit name.
Volume	String	R/W	Volume serial number.
SpaceUnit	ZosSpaceUnit	R/W	Space units (cylinders, tracks, blocks)
PrimarySpace	Int32	R/W	Primary space quantity.
SecondarySpace	Int32	R/W	Secondary space quantity.
DirectoryBlocks	Int32	R/W	PDS directory blocks.
ExtendedAttributes	ZosDataSetEAttr	R/W	Extended attributes: Default, No, Opt
PdseVersion	Int16	R/W	PDSE version number: 0 (default), 1, 2
MaxGens	Int32	R/W	Maximum number of PDSE member generations

ZosDataSetProfile

The **ZosDataSetProfile** object represents a single data set profile. This object can be obtained using the **Item** property of **ZosDataSetProfiles**. **ZosDataSetProfile** specifies the default attributes of a newly created data set based on data set name patterns.

ZosDataSetProfile Constructor

The following constructor can be used to initialize a new **ZosDataSetProfile** object:

Constructor	Parameters
`ZosFileFormatMapping(`	
`    String dsName,`	Data set name
`    ZosDataSetType dsType,`	Data set type
`    ZosRecordFormat recordFmt, [optional]`	Record format
`    Int16 recordLength,        [optional]`	Record length
`    Int16 blockSize,           [optional]`	Block size
`    String dataClass,          [optional]`	Data class
`    String storageClass,       [optional]`	Storage class
`    String managementClass,    [optional]`	Management class
`    ZosSpaceUnit spaceUnit,    [optional]`	Space units
`    Int32 primarySpace,        [optional]`	Primary
`    Int32 nSecondarySpace,     [optional]`	Secondary space
`    Int32 directoryBlocks,     [optional]`	Directory blocks
`    String unitName,           [optional]`	Unit name
`    String volume,             [optional]`	Volume serial number
`    ZosDataSetEAttr eattr,     [optional]`	Extended attributes
`    Int16 pdseVersion,         [optional]`	PDSE version number (0, 1, 2)
`    Int32 maxGens              [optional]`	Max number of generations
`    )`	

ZosDataSetProfile Properties

ZosDataSetProfile exposes the following properties:

Property	Type	R/W	Description
DataSetName	String	R	Data set name pattern.
DataSetType	ZosDataSetType	R	Data set type: Seq, Pds, Pdse
RecordFormat	ZosRecordFormat	R	Record format.
RecordLength	Int16	R	Record length.
BlockSize	Int16	R	Block size.
DataClass	String	R	SMS data class.
StorageClass	String	R	SMS storage class.
ManagementClass	String	R	SMS management class.
UnitName	String	R	Unit name.
Volume	String	R	Volume serial number.

SpaceUnit	ZosSpaceUnit	R	Space units (cylinders, tracks, blocks)
PrimarySpace	Int32	R	Primary space quantity.
SecondarySpace	Int32	R	Secondary space quantity.
DirectoryBlocks	Int32	R	PDS directory blocks.
ExtendedAttributes	ZosDataSetEAttr	R	Extended attributes (no, optional)
PdseVersion	Int16	R	PDSE version number: 0 (default), 1, 2
MaxGens	Int32	R	Maximum number of PDSE member generations

ZosDataSetProfiles

The **ZosDataSetProfiles** object is a collection of all data set profiles for a server. This object is obtained using the **DataSetProfiles** property of the **ZosServer** object.

ZosDataSetProfiles Properties

ZosDataSetProfiles exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosDataSetProfile	R	Data set profile with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosDataSetProfiles Methods

ZosDataSetProfiles exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`ZosDataSetProfile[] ToArray()`	Copies the entire collection to a one-dimensional array.
`void FromArray( ZosDataSetProfile[] array )`	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.

`Int32`
`    Add(`
`        Int32 index,`
`        ZosDataSetProfile profile`
`        )`
	Adds a new data set profile.
	Index indicates position for new item. Specify –1 to insert at end.
	Returns index at which object has been added.

`Int32`
`    Add(`
`        Int32 index,`
`        String dsName,`
`        ZosDataSetType dsType,`
`        ZosRecordFormat recordFmt, [optional]`
`        Int16 recordLength,        [optional]`
`        Int16 blockSize,           [optional]`
`        String dataClass,          [optional]`
`        String storageClass,       [optional]`
`        String managementClass,    [optional]`
`        ZosSpaceUnit spaceUnit,    [optional]`
`        Int32 primarySpace,        [optional]`
`        Int32 nSecondarySpace,     [optional]`
`        Int32 directoryBlocks,     [optional]`
`        String unitName,           [optional]`
`        String volume,             [optional]`
`        ZosDataAetEAttr eattr,     [optional]`
`        Int16 pdseVersion,         [optional]`
`        Int32 maxGens              [optional]`
`        )`
	Adds a new data set profile.
	Index indicates position for new item. Specify –1 to insert at end.
	Returns index at which object has been added.

`void`
`    RemoveAt(`
`        Int32 index`
`        )`
	Deletes a data set profile, specified by index.

`Boolean`
`    Remove(`
`        String name`
`        )`
	Deletes a data set profile, specified by data set name pattern. Returns true if item was removed or false if item is not found.

`Int32`
`    Move(`
`        Int32 indexTo,`
`        Int32 indexFrom`
`        )`
	Changes the order of data set profiles.

`Int32`
`    FindIndex(`
`        String name`
`        )`
	Searches for profile with specified name and returns zero-based index. Returns -1 if name is not found.

`ZosDataSetProfile`
`    Find(`
`        String name`
`        )`
	Searches for profile with specified name and returns reference to object. Returns null if name is not found.

ZosFileExtensionMapping

The **ZosFileExtensionMapping** object represents a single file extension mapping. This object can be obtained using the Item property of **ZosFileExtensionMappings**. **ZosFileExtensionMapping** maps a data set name pattern to a local file name extension.

ZosFileExtensionMapping Constructor

The following constructor can be used to initialize a new **ZosFileExtensionMapping** object:

Constructor	Parameters

`ZosFileExtensionMapping(`
`    String dsName,`
`    String fileExt`
`    )`
	Data set name pattern
	File extension

ZosFileExtensionMapping Properties

ZosFileExtensionMapping exposes the following properties:

Property	Type	R/W	Description
DataSetName	String	R	Data set name pattern.
FileExtension	String	R	File extension.

ZosFileExtensionMappings

The **ZosFileExtensionMappings** object is a collection of all file extension mappings for a server. This object is obtained using the **FileExtensions** property of the **ZosServer** object.

ZosFileExtensionMappings Properties

ZosFileExtensionMappings exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosFileExtensionMapping	R	File extension mapping with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosFileExtensionMappings Methods

ZosFileExtensionMappings exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`ZosFileExtensionMapping[] ToArray()`	Copies the entire collection to a one-dimensional array.
`void FromArray( ZosFileExtensionMapping[] array )`	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.
`Int32 Add( Int32 index, ZosFileExtensionMapping mapping )`	Adds a new file extension mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`Int32 Add( Int32 index, String dsName, String fileExtension )`	Adds a new file extension mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`void RemoveAt( Int32 index )`	Deletes a file extension, specified by index.
`Boolean Remove( String name )`	Deletes a file extension, specified by data set name pattern. Returns true if item was removed or false if item is not found.
`Int32 Move( Int32 indexTo, Int32 indexFrom )`	Changes the order of file extension mappings.
`Int32 FindIndex( String name )`	Searches for mapping with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosFileExtensionMapping Find( String name )`	Searches for mapping with specified name and returns reference to object. Returns null if name is not found.

ZosFileExtensionMappings Examples

Examples of using **ZosFileExtensionMappings** are shown below:

C#
```
ZosFileExtensionMappings mappings;
ZosFileExtensionMapping[] mappingArray = new ZosFileExtensionMapping[]
{
    new ZosFileExtensionMapping("**.CNTL", "jcl"),
    new ZosFileExtensionMapping("**.LIST", "txt")
};
mappings.FromArray(mappingArray);
mappings.Add(-1, "**.COBOL", "cbl");
mappings.Remove(“**.PARMLIB”);
mappings.Move(4,2);
```

C++
```
ZosFileExtensionMappings^ mappings;
array<ZosFileExtensionMapping>^ mappingArray =
{
    new ZosFileExtensionMapping("**.CNTL", "jcl"),
    new ZosFileExtensionMapping("**.LIST", "txt")
};
mappings.FromArray(mappingArray);
mappings->Add(-1, "**.COBOL", "cbl");
mappings->Remove(“**.PARMLIB”);
mappings->Move(4,2);
```

Visual Basic
```
Dim mappings As ZosFileExtensionMappings
Dim mappingArray() As ZosFileExtensionMapping = _
{ _
    New ZosFileExtensionMapping("**.CNTL", "jcl"), _
    New ZosFileExtensionMapping("**.LIST", "txt")  _
}
mappings.FromArray(mappingArray)
mappings.Add(-1, "**.COBOL", "cbl")
mappings.Remove(“**.PARMLIB”)
mappings.Move(4,2)
```

Jscript
```
var mappings : ZosFileExtensionMappings;
var mappingArray : ZosFileExtensionMapping[] =
[
    new ZosFileExtensionMapping("**.CNTL", "jcl"),
    new ZosFileExtensionMapping("**.LIST", "txt")
];
mappings.FromArray(mappingArray);
mappings.Add(-1, "**.COBOL", "cbl");
mappings.Remove(“**.PARMLIB”);
mappings.Move(4,2);
```

ZosFileFormatMapping

The **ZosFileFormatMapping** object represents a single file format mapping. This object can be obtained using the Item property of **ZosFileFormatMappings**. **ZosFileFormatMapping** maps a data set name pattern to a local file format.

ZosFileFormatMapping Constructor

The following constructor can be used to initialize a new **ZosFileFormatMapping** object:

Constructor	Parameters

`ZosFileFormatMapping(`
`    String name,`
`    ZosFileFormat format`
`    )`
	
	Name pattern
	File format

ZosFileFormatMapping Properties

ZosFileFormatMapping exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name pattern.
FileFormat	ZosFileFormat	R	File format

ZosFileFormatMapping Examples

Examples of using **ZosFileFormatMapping** are shown below:

C#
```
ZosFileFormatMapping format;
String name = format.Name;
ZosFileFormat type = format.FileFormat;
```

C++
```
ZosFileFormatMapping^ format;
String^ name = format->Name;
ZosFileFormat^ type = format->FileFormat;
```

Visual Basic
```
Dim format As ZosFileFormatMapping;
Dim name As String = format.Name
Dim type As ZosFileFormat = format.FileFormat
```

Jscript
```
var format : ZosFileFormatMapping;
var name : String = format.Name;
var type : ZosFileFormat = format.FileFormat;
```

ZosFileFormatMappings

The **ZosFileFormatMappings** object is a collection of file format mappings for a server or a ChangeMan instance. This object is obtained using the **DataSetFileFormats** property or the **UnixFileFormats** property of the **ZosServer** object. For ChangeMan instances, it can be obtained using the **FileFormats** property of the **ZosChangeManFolder** object.

ZosFileFormatMappings Properties

ZosFileFormatMappings exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosFileFormatMapping	R	File format mapping with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection

ZosFileFormatMappings Methods

ZosFileFormatMappings exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes collection.

`ZosFileFormatMapping[]`
`    ToArray()`
	Copies the entire collection to a one-dimensional array.

`void`
`    FromArray(`
`        ZosFileFormatMapping[]`
`            array`
`        )`
	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.

`Int32`
`    Add(`
`        Int32 index,`
`        ZosFileFormatMapping`
`            mapping`
`        )`
	Adds a new file format mapping.
	Index indicates position for new item. Specify –1 to insert at end.
	Returns index at which object has been added.

`Int32`
`    Add(`
`        Int32 index,`
`        String name,`
`        ZosFileFormat format`
`        )`
	Adds a new file format mapping.
	Index indicates position for new item. Specify –1 to insert at end.
	Returns index at which object has been added.

`void RemoveAt( Int32 index )`	Deletes a file format mapping, specified by index.
`Boolean Remove( String name )`	Deletes a file format mapping, specified by data set name pattern. Returns true if item was removed or false if item is not found.
`Int32 Move( Int32 indexTo, Int32 indexFrom )`	Changes the order of file format mappings.
`Int32 FindIndex( String name )`	Searches for mapping with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosFileFormatMapping Find( String name )`	Searches for mapping with specified name and returns reference to object. Returns null if name is not found.

ZosFileFormatMappings Examples

Examples of using **ZosFileFormatMappings** are shown below:

C#

```
ZosFileFormatMappings formats;
ZosFileFormatMapping[] formatArray =
   new ZosFileFormatMapping[]
{
   new ZosFileFormatMapping("**.ETEXT", ZosFileFormat.EbcdicText),
   new ZosFileFormatMapping("**.ATEXT", ZosFileFormat.AsciiText)
};
formats.FromArray(formatArray);
formats.Add(-1, “**.BINARY”, ZosFileFormat::Binary);
formats.Remove(“**.TRASH”);
formats.Move(4,2);
```

C++

```
ZosFileFormatMappings^ formats;
array<ZosFileFormatMapping>^ formatArray =
{
   new ZosFileFormatMapping("**.ETEXT", ZosFileFormat.EbcdicText),
   new ZosFileFormatMapping("**.ATEXT", ZosFileFormat.AsciiText)
};
formats.FromArray(formatArray);
formats->Add(-1, “**.BINARY”, ZosFileFormat::Binary);
formats->Remove(“**.TRASH”);
formats->Move(4,2);
```

Visual Basic

```
Dim formats As ZosFileFormatMappings
Dim formatArray() As ZosFileFormatMapping = _
{ _
 New ZosFileFormatMapping("**.ETEXT", ZosFileFormat.EbcdicText), _
 New ZosFileFormatMapping("**.ATEXT", ZosFileFormat.AsciiText) _
}file format mapping
formats.FromArray(formatArray)
formats.Add(-1, "**.BINARY", ZosFileFormat::Binary)
formats.Remove("**.TRASH")
formats.Move(4,2)
```

Jscript

```
var formats : ZosFileFormatMappings;
var formatArray : ZosFileFormatMapping[] =
[
   new ZosFileFormatMapping("**.ETEXT", ZosFileFormat.EbcdicText),
   new ZosFileFormatMapping("**.ATEXT", ZosFileFormat.AsciiText)
];
formats.FromArray(formatArray);
formats.Add(-1, "**.BINARY", ZosFileFormat::Binary);
formats.Remove("**.TRASH");
formats.Move(4,2);
```

ZosJesFile

The **ZosJesFile** object represents a JES spool file that is part of a job's output. This object can be obtained using the **GetFile** or **GetFiles** methods of **ZosJesJob**.

ZosJesFile Properties

ZosJesFile exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for spool file. Spool file names have the following format (.txt is the extension): *dsid.jobstep.procstep.ddname*.txt
Path	String	R	Full file system path name for the spool file
DsID	String	R	Spool data set ID (unique identifier).
JobStep	String	R	Job step name.
ProcStep	String	R	Proc step name.
DDName	String	R	DD name.

Owner	String	R	Owner user ID.
Dest	String	R	Print destination.
Forms	String	R	Form number.
ProcessMode	String	R	Process mode.
AppcJobName	String	R	APPC job name.
AppcJobID	String	R	APPC job ID.
Bytes	Int64	R	Number of bytes.
Lines	Int64	R	Number of lines.
Pages	Int64	R	Number of pages.
RecLen	Int16	R	Record length.
Copies	Int16	R	Number of copies.
Class	Char	R	Output class.
Queue	ZosOutputQueue	R	Output queue (writer, hold, or external writer).

ZosJesFile Methods

ZosJesFile exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the spool file information.
`void Delete()`	Deletes the JES spool file.
`void Requeue( Char newClass, String newDest [optional] )`	Re-queues the JES spool file to a new output class and destination.

ZosJesJob

The **ZosJesJob** object represents a JES job on the server. This object can be obtained using the **GetJesJob** method of **ZosServer** or the **GetJesJobs** method of **ZosJobFolder**.

ZosJesJob Properties

ZosJesJob exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File system name: *jobname.jobid*
Path	String	R	Full file system path name of the job.
JobName	String	R	Job name
JobID	String	R	Job ID
Owner	String	R	Owner user ID
Class	String	R	Job class
System	String	R	System on which job is active
OriginNode	String	R	Origin node from which job was submitted
ExecutionNode	String	R	Execution node on which job ran
JobStep	String	R	Currently executing job step
ProcStep	String	R	Currently executing proc step
CompletionType	ZosJobCompletionType	R	Job completion type
CompletionCode	Int32	R	Highest return code or last abend code
Completion	String	R	Formatted job completion code and type
Status	ZosJobStatus	R	Job status (general)
Phase	ZosJobPhase	R	Job phase (specific job status)
Type	ZosJobType	R	Job type (batch, started task, TSO, APPC)
HoldType	ZosJobHoldType	R	Job hold type
Priority	Int16	R	Job priority

ZosJesJob Methods

ZosJesJob exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the job information.

`void`
`    Cancel(`
`        Boolean purge  [optional]`
`        )`
	Cancel job and optionally purge output.

`void`
`    Delete()`
	Deletes spool output.

`void`
`    Requeue(`
`        Char newClass,`
`        String newDest  [optional]`
`        )`
	Re-queues spool output to a new output class and destination.

`ZosJesFile`
`    GetFile(`
`        String name`
`        )`
	Gets a specific JES spool file that belongs to a job. The file can be specified using either the full file name, or just the data set ID (DSID).

`ZosJesFile`
`    GetFile(`
`        String jobstep,`
`        String procstep,`
`        String ddname)`
	Gets a specific JES spool file that belongs to a job. The file is specified using the job step, proc step, and DD name combination.

`ZosJesFile[]`
`    GetFiles(`
`        String filter  [optional]`
`        )`
	Gets an array of JES spool files that belong to a job. The list can optionally be filtered by spool file name with wild characters.

ZosJobFolder

The **ZosJobFolder** object represents a single job folder. This object can be obtained using the **Item** property of **ZosJobFolders**.

ZosJobFolder Properties

ZosJobFolder exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the folder.
Path	String	R	Full path name of the folder.
Subfolders	ZosJobFolders	R	Collection of all subfolders for this folder.
QueryType	ZosJobQueryType	R/W	Type of job query.
QueryArgument	String	R/W	Query search argument is job name, prefix, or userid.

ZosJobFolder Methods

ZosJobFolder exposes the following methods:

Function	Description
`ZosJesJob[] GetJesJobs()`	Gets an array of JES jobs that match the filters for this folder.

ZosJobFolders

The **ZosJobFolders** object is a collection of all job folders with the same parent folder. This object is obtained using the **JobFolders** property of **ZosServer** or the **Subfolders** property of the **ZosJobFolder** object.

ZosJobFolders Properties

ZosJobFolders exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosJobFolder	R	Folder with specified name or index.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosJobFolders Methods

ZosJobFolders exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`Int32 Add( String folderName, ZosJobQueryType queryType, String queryArg [optional] )`	Adds a new folder. Search argument is job name, prefix, or userid. Returns index at which object has been added.
`Boolean Remove( String folderName )`	Deletes a folder. Returns true if folder was removed or false if folder is not found.

`Int32 FindIndex( String name )`	Searches for folder with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosJobFolder Find( String name )`	Searches for folder with specified name and returns reference to object. Returns null if name is not found.

ZosLibTypeMapping

The **ZosLibTypeMapping** object represents a single library type mapping. This object can be obtained using the Item property of **ZosLibTypeMappings**. **ZosLibTypeMapping** maps a data set name pattern to a library type.

ZosLibTypeMapping Constructor

The following constructor can be used to initialize a new **ZosLibTypeMapping** object:

Constructor	Parameters
`ZosLibTypeMapping( String dsName, ZosLibType libType )`	Data set name pattern Library type (Standard, Librarian, Panvalet)

ZosLibTypeMapping Properties

ZosLibTypeMapping exposes the following properties:

Property	Type	R/W	Description
DataSetName	String	R	Data set name pattern.
LibType	ZosLibType	R	Library type (Standard, Librarian, Panvalet)

ZosLibTypeMapping Examples

Examples of using **ZosLibTypeMapping** are shown below:

C#

```
ZosLibTypeMapping libType;
String dsName = libType.DataSetName;
ZosLibType type = libType.LibType;
```

C++

```
ZosLibTypeMapping^ libType;
String^ dsName = libType->DataSetName;
ZosLibType ^ type = libType->LibType;
```

Visual Basic

```
Dim libType As ZosLibTypeMapping
Dim dsName As String = libType.DataSetName
Dim type As ZosLibType = libType.LibType
```

Jscript

```
Var libType : ZosLibTypeMapping;
var dsName : String = libType.DataSetName;
var type : ZosLibType = libType.LibType;
```

ZosLibTypeMappings

The **ZosLibTypeMappings** object is a collection of all library type mappings for a server. This object is obtained using the **LibTypes** property of the **ZosServer** object. Library types only need to be defined if you are using Librarian or Panvalet libraries.

ZosLibTypeMappings Properties

ZosLibTypeMappings exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosLibTypeMapping	R	Library type mapping with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosLibTypeMappings Methods

ZosLibTypeMappings exposes the following methods:

Function	Description
`void Refresh()`	Refreshes collection.
`ZosLibTypeMapping[] ToArray()`	Copies the entire collection to a one-dimensional array.
`void FromArray( ZosLibTypeMapping[] array )`	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.
`Int32 Add( Int32 index, ZosLibTypeMapping mapping )`	Adds a new library type mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`Int32 Add( Int32 index, String dsName, ZosLibType libType )`	Adds a new library type mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`void RemoveAt( Int32 index )`	Deletes a library type mapping, specified by index.
`Boolean Remove( String name )`	Deletes a library type mapping, specified by data set name pattern. Returns true if item was removed or false if item is not found.
`Int32 Move( Int32 indexTo, Int32 indexFrom )`	Changes the order of library type mappings.
`Int32 FindIndex( String name )`	Searches for mapping with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosLibTypeMapping Find( String name )`	Searches for mapping with specified name and returns reference to object. Returns null if name is not found.

ZosLibTypeMappings Examples

Examples of using **ZosLibTypeMappings** are shown below:

C#

```
ZosLibTypeMappings libTypes;
ZosLibTypeMapping[] libTypeArray = new ZosLibTypeMapping[]
{
    new ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib),
    new ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)
};
libTypes.FromArray(libTypeArray);
libTypes.Add(-1, “**.PANVALET”, ZosLibType::Pan);
libTypes.Remove(“**.LIBRARY”);
libTypes.Move(4,2);
```

C++

```
ZosLibTypeMappings^ libTypes;
array<ZosLibTypeMapping>^ libTypeArray =
{
    new ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib),
    new ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)
};
libTypes.FromArray(libTypeArray);
libTypes->Add(-1, “**.PANVALET”, ZosLibType::Pan);
libTypes->Remove(“**.LIBRARY”);
libTypes->Move(4,2);
```

Visual Basic

```
Dim libTypes As ZosLibTypeMappings
Dim libTypeArray() As ZosLibTypeMapping = _
{ _
    New ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib), _
    New ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)  _
}
libTypes.FromArray(libTypeArray)
libTypes.Add(-1, “**.PANVALET”, ZosLibType::Pan)
libTypes.Remove(“**.LIBRARY”)
libTypes.Move(4,2)
```

Jscript

```
var libTypes : ZosLibTypeMappings;
var libTypeArray : ZosLibTypeMapping[] =
[
    new ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib),
    new ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)
];
libTypes.FromArray(libTypeArray);
libTypes.Add(-1, “**.PANVALET”, ZosLibType::Pan);
libTypes.Remove(“**.LIBRARY”);
libTypes.Move(4,2);
```

ZosNameFilters

The **ZosNameFilters** object is a collection of all name filters for a folder. This object is obtained using the Filters property of the **ZosDataSetFolder** object or the Filters property of the **ZosChangeManInstance** object.

ZosNameFilters Properties

ZosNameFilters exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	String	R	Filter with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosNameFilters Methods

ZosNameFilters exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes collection.

`String[]`
`    ToArray()`
	Copies the entire collection to a one-dimensional array.

`void`
`    FromArray(`
`        String[] array`
`        )`
	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.

`void`
`    Add(`
`        String name`
`        )`
	Adds a new name filter.

`void`
`    RemoveAt(`
`        Int32 index`
`        )`
	Deletes a filter, specified by index.

`Boolean`
`    Remove(`
`        String name`
`        )`
	Deletes a filter, specified by name. Returns true if item was removed or false if item is not found.

`Int32`
`    FindIndex(`
`        String name`
`        )`
	Searches for filter with specified name and returns zero-based index. Returns -1 if name is not found.

ZosNameFilters Examples

Examples of using **ZosNameFilters** are shown below:

C#

```
ZosNameFilters filters;
String[] filterArray = new String[]
{
    "**.ASM",
    "**.JAVA"
};
filters.FromArray(filterArray);
filters.Add("**.COBOL");
filters.Remove(“**.LIST”);
```

C++

```
ZosNameFilters^ filters;
array<String>^ filterArray =
{
    "**.ASM",
    "**.JAVA"
};
filters.FromArray(filterArray);
filters->Add("**.COBOL");
filters->Remove(“**.LIST”);
```

Visual Basic

```
Dim filters As ZosNameFilters
Dim filterArray() As String = _
{ _
    "**.ASM", _
    "**.JAVA" _
}
filters.FromArray(filterArray)
filters.Add("**.COBOL")
filters.Remove(“**.LIST”)
```

Jscript

```
var filters : ZosNameFilters;
var filterArray : String[] =
[
    "**.ASM",
    "**.JAVA"
];
filters.FromArray(filterArray);
filters.Add("**.COBOL");
filters.Remove(“**.LIST”);
```

ZosNameType

The **ZosNameType** object represents a name/type pair that is used to specify component names and types for functions such as promote or demote.

ZosNameType Constructor

The following constructor can be used to initialize a new **ZosNameType** object:

Constructor	Parameters

`ZosDataNameType(`
`    String name,`
`    String type`
`    )`
	
	Component name
	Component type

ZosNameType Properties

ZosNameType exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Component name.
Type	String	R	Component type.

ZosNameValue

The **ZosNameValue** object represents a name/value pair that is used to specify user variables for functions such as create package, promote, demote, or audit.

ZosNameValue Constructor

The following constructor can be used to initialize a new **ZosNameValue** object:

Constructor	Parameters

`ZosDataNameValue(`
`    String name,`
`    String value`
`    )`
	
	Name of the variable
	Value of the variable

ZosNameValue Properties

ZosNameValue exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the variable.
Value	String	R	Value of the variable.

ZosNetwork

The **ZosNetwork** object represents the overall ZDD Network. **ZosNetwork** is always the starting point for the ChangeMan ZDD programming interface. It is created as shown in the following section.

ZosNetwork Constructor

The following constructor can be used to initialize a new **ZosNetwork** object:

Constructor	Parameters
`ZosNetwork()`	(none)

See the ZosNetwork Examples section for an example of initializing the network for access.

ZosNetwork Properties

ZosNetwork exposes the following properties:

Property	Type	R/W	Description
Servers	ZosServers	R	Collection of all servers.
Servers[name]	ZosServer	R	Server with specified name.
Servers[name, id]	ZosServer	R	Server with specified name and connection ID.
LocalCodePage	Int32	R/W	Local ASCII code page.
CacheFolder	String	R/W	Name of folder used to store cached files.
CacheDays	Int32	R/W	Number of days to keep cached files.
NotifyPort	Int32	R/W	TCP/IP port number used to receive notification messages from the server. This port number should be unblocked on your local machine in the Windows firewall or other firewall software.
NotifyDelay	Int32	R/W	Time delay, in seconds, before a message box is displayed. The time delay allows messages to accumulate so that several messages can be displayed in a single message box.
NotifyMessageBox	Boolean	R/W	Display message box for notify messages.

TimeOut	Int32	R/W	Time, in minutes, to wait for a network operation to complete. Network operations are aborted if no response is received after this period of time. Must be in the range 3 - 30 minutes.
KeepAlive	Int32	R/W	TCP/IP keep alive time interval, in minutes. TCP/IP keep alive packets are sent after this many minutes of inactivity to detect lost connections.
MaxUploadSize	Int32	R/W	Maximum upload file size, in megabytes. Setting this value too high can exhaust virtual storage and cause S878 abends in the server. Must be in the range 16 - 256 megabytes.

ZosNetwork Examples

ZosNetwork is the root of the ChangeMan ZDD programming interface. The **ZosNetwork** object is created as shown in the following examples.

C#
```
ZosNetwork network = new ZosNetwork();
```

C++
```
ZosNetwork^ network = gcnew ZosNetwork();
```

Visual Basic
```
Dim network As New ZosNetwork()
```

Jscript
```
var network : ZosNetwork = new ZosNetwork();
```

Examples of getting or setting network properties are shown below.

C#

```
ZosNetwork network = new ZosNetwork();
ZosServers servers = network.Servers;
ZosServer server = network.Servers[“SYSA”];
network.CacheFolder = "C:\\Temp";
network.CacheDays   = 3;
network.NotifyPort  = 8000;
network.NotifyDelay = 60;
network.NotifyMessageBox = true;
```

C++

```
ZosNetwork^ network = gcnew ZosNetwork();
ZosServers^ servers = network->Servers;
ZosServer^ server = network->Servers[“SYSA”];
network->CacheFolder = "C:\\Temp";
network->CacheDays   = 3;
network->NotifyPort  = 8000;
network->NotifyDelay = 60;
network->NotifyMessageBox = true;
```

Visual Basic

```
Dim network As new ZosNetwork()
Dim servers As ZosServers = network.Servers
network.CacheFolder = "C:\Temp"
network.CacheDays   = 3
network.NotifyPort  = 8000
network.NotifyDelay = 60
network.NotifyMessageBox = True
```

Jscript

```
var network : ZosNetwork = new ZosNetwork();
var servers : ZosServers = network.Servers;
var server : ZosServer = network.Servers[“SYSA”];
network.CacheFolder = "C:\\Temp";
network.CacheDays   = 3;
network.NotifyPort  = 8000;
network.NotifyDelay = 60;
network.NotifyMessageBox = true;
```

ZosPackage

The **ZosPackage** object represents a ChangeMan ZMF package. This object can be obtained using either the **GetPackage** method or the **GetPackages** method of **ZosApplication**.

ZosPackage Properties

ZosPackage exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the package.
Path	String	R	Full path name of the package.
Application	ZosApplication	R	Parent application for this package.
Release	ZosRelease	R	Parent release for this package. Null if package is not attached to a release.
Title	String	R/W	Package title.
RequestorName	String	R/W	Requestor name.
RequestorPhone	String	R/W	Requestor telephone number.
WorkRequest	String	R/W	Work request number or name.
Department	String	R/W	Department number or name.
SuperPackage	String	R	Parent super or complex package.
CreatorUserID	String	R	Creator user ID.
Level	ZosPackageLevel	R/W	Package level.
Type	ZosPackageType	R	Package type.
Status	ZosPackageStatus	R	Package status.
TempDuration	Int32	R/W	Temporary change duration. This property is available for temporary packages only.
ReasonCode	Int32	R/W	Reason code.
AuditReturnCode	Int32	R	Audit return-code.
AuditPending	Boolean	R	Audit pending package lock

NearestInstallDate	DateTime	R	Nearest scheduled install date.
CreatedTime	DateTime	R	Date and time package was created.
InstalledTime	DateTime	R	Date and time package was installed.
FrozenTime	DateTime	R	Date and time package was frozen.
ApprovedTime	DateTime	R	Date and time package was approved.
BaselinedTime	DateTime	R	Date and time package was baselined.
BackedOutTime	DateTime	R	Date and time package was backed out.
RevertedTime	DateTime	R	Date and time package was reverted.
Description	String	R/W	Package description. The description is a single string, but can contain multiple lines, delimited by newline characters. When setting the description if a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.
SchedulerType	ZosSchedulerType	R/W	Scheduler type. This property is available for simple and participating packages only.
ProblemActionType	ZosProblemActionType	R/W	Problem action code. This property is available for simple and participating packages only.

OtherProblemAction	String	R/W	Other problem action. This property is available for simple and participating packages only.
ImplementationInstructions	String	R/W	Implementation instructions. he implementation instructions consist of a single string, but can contain multiple lines, delimited by newline characters. When setting the implementation instructions, if a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines. This property is available for simple and participating packages only.
ParticipatingPackages	String[]	R/W	Participating packages. The value is an array of strings, each containing a package name. This property is available for complex and super packages only.
AffectedApplications	String[]	R/W	Affected applications. The value is an array of strings, each containing an application name. This property is available for participating packages only.
PredecessorJobs	String[]	R/W	Predecessor jobs. The value is an array of strings, each containing a job name. This property is available for simple and participating packages only.

SuccessorJobs	String[]	R/W	Successor jobs. The value is an array of strings, each containing a job name. This property is available for simple and participating packages only.
Sites	ZosPackageSite[]	R/W	Package site information. The value is an array of package site objects. This property is available for simple and participating packages only.
Site	ZosPackageSite	R/W	Package site information. The value is a single package site object for packages with single sites. This property is available for simple and participating packages with a single site only.
Release	String	R	Name of ERO release with which package is associated.
ReleaseArea	String	R	Name of starting release area for release package check in.
ReleaseJoinedDate	DateTime	R	Date and time that package joined the release.
UserVariables	ZosNameValue[]	R/W	User variables (multiple). Allows getting or setting multiple user variables as an array. Each user variable in the array is a name/ value pair. See the chart below for list of valid variable names.

User variables are a set of name/value pairs. Each name must be one of the names in the chart below.

Variable Name	Value Length
UserVarLen101 - UserVarLen199	1
UserVarLen201 - UserVarLen211	2
UserVarLen301 - UserVarLen310	3
UserVarLen401 - UserVarLen410	4
UserVarLen801 - UserVarLen810	8
UserVarLen1601 - UserVarLen1605	16
UserVarLen4401 - UserVarLen4405	44
UserVarLen7201 - UserVarLen7205	72

ZosPackage Methods

ZosPackage exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the package information.

`ZosPackageInfo`
`    GetInfo()`
	Gets a package information object containing the package information. This object can be used to create a new package modeled after this package.

`ZosPackageSite`
`    GetSite(`
`        String siteName`
`        )`
	Get package site information by site name.

`void`
`    AddSite(`
`        ZosPackageSite site,`
`        )`
	Adds a new site to the package. If the site name already exists, the existing site information is replaced.

`void`
`    AddSite(`
`        String siteName,`
`        String primaryContactName,`
`        String primaryContactPhone,`
`        String alternateContactName,`
`        String alternateContactPhone,`
`        DateTime installStartTime,`
`        DateTime installEndTime`
`        )`
	Adds a new site to the package. If the site name already exists, the existing site information is replaced.

`Boolean`
`    RemoveSite(`
`        String siteName`
`        )`
	Removes a site from the package.

`void`
`    SetContact(`
`        String primaryContactName,`
`        String primaryContactPhone,`
`        String alternateContactName,`
`        String alternateContactPhone`
`        )`
 	
`void`
`    SetContact(`
`        String siteName,`
`        String primaryContactName,`
`        String primaryContactPhone,`
`        String alternateContactName,`
`        String alternateContactPhone`
`        )`
	Updates contact information. If no site name is specified in a DP environment, then all sites are updated with this contact information.

`void`
`    SetInstallTime(`
`        DateTime installStartTime,`
`        DateTime installEndTime)`
 	
`void`
`    SetInstallTime(`
`        String siteName,`
`        DateTime installStartTime,`
`        DateTime installEndTime)`
	Updates package install time. If no site name is specified in a DP environment, then all sites are updated with this same install time.

`ZosPackageLibrary`
`    GetLibrary(`
`        String libType`
`        )`
	Gets a single package library by name.

`ZosPackageLibrary[]`
`    GetLibraries()`
	Gets an array containing the staging libraries for a package.

`ZosPackageComponentFile`
`    GetComponent(`
`        String fileName`
`        )`
 	
`ZosPackageComponentFile`
`    GetComponent(`
`        String componentName,`
`        String libraryType`
`        )`
	Gets a single component by name and library type.
	For PDS member components, the name may be specified as "component.lib" or as separate component and library type names.
	For Unix libraries, componetName is the path name relative to the package library root.

```
ZosPackageComponentFile[]
    GetComponents()

ZosPackageComponentFile[]
    GetComponents(
        bool includeGenerated
        )

ZosPackageComponentFile[]
    GetComponents(
        String nameFilter
        )

ZosPackageComponentFile[]
    GetComponents(
        ZosComponentStatusFlags flags
        )

ZosPackageComponentFile[]
    GetComponents(
        DateTime changeTime
        )

ZosPackageComponentFile[]
    GetComponents(
        String nameFilter,
        bool includeGenerated
        )

ZosPackageComponentFile[]
    GetComponents(
        String nameFilter,
        ZosComponentStatusFlags flags
        )

ZosPackageComponentFile[]
    GetComponents(
        String nameFilter,
        bool includeGenerated,
        ZosComponentStatusFlags flags)

ZosPackageComponentFile[]
    GetComponents(
        String nameFilter,
        bool includeGenerated,
        ZosComponentStatusFlags flags,
        DateTime changeTime
        )
```

Gets an array of components that belong to a package.

The list can optionally be filtered by component name and component status.

The includeGenerated flag allows you to specify whether or not to include generated component types (LST, LOD, etc.).

Unix components are retrieved recursively, and the array returned contains components from all subdirectory levels.

The array returned contains component files only and does not include any directory objects.

nameFilter - Name filter

includeGenerated - Include generated components (LST, LOD, etc

statusFlags - Status filter flags

changeTime - get componentschanged after the specified time

`void`
`    CheckIn(`
`        String sourcePath,`
`        String componentName,`
`        String libtype,`
`        Boolean lock, [optional]`
`        String description  [optional]`
`        ZosFileFormat format [optional]`
`        )`
 	
`void`
`    CheckIn(`
`        String sourcePath,`
`        String[] componentNames,`
`        String libtype,`
`        Boolean lock,  [optional]`
`        String description  [optional]`
`        ZosFileFormat format [optional]`
`        )`
 	
`void`
`    CheckIn(`
`        String sourcePath,`
`        String componentName,`
`        String libtype,`
`        String targetSubdir,`
`        Boolean lock, [optional]`
`        String description  [optional]`
`        ZosFileFormat format [optional]`
`        )`
 	
`void`
`    CheckIn(`
`        String sourcePath,`
`        String[] componentNames,`
`        String libtype,`
`        String targetSubdir,`
`        Boolean lock,  [optional]`
`        String description  [optional]`
`        ZosFileFormat format [optional]`
`        )`
	Checks components in to a package. CheckIn does not build the comoponents; the Build function must be performed separately.
	
	The source path can refer to a directory on the local file system, a partitioned data set on the server, or a Unix directory on the server.
	
	If multiple components are specified, all must come from the same directory tree or data set.
	If checking in multiple components, all components must belong to the same library type.
	
	When checking in from a data set, specify the path as follows:
	
	`\\server\DataSets\dsname`
	
	where `server` is the server name and `dsname` is the name of the partitioned data set.
	
	When checking in from a Unix directory, specify the source directory path as follows:
	
	`\\`**`server`**`\Unix\`**`dirname`**
	
	where `server` is the server name and `dirname` is the path name of parent Unix directory.
	
	For Unix, the component names specify relative paths. Component names are relative to the the source path and relative to the target subdirectory.
	If no target subdirectory is specified, the target subdirectory is the root directory for the library type.

`ZosCheckInStatus[]`
`    ReleaseCheckIn(`
`       ZosNameType[] componentNames,`
`       Boolean replace,  [optional]`
`       Boolean eligibleOnly,  [optional]`
`       String changeDescription [optional]`
`        )`
	Checks package components in to a release. Component names are specified as name and type pairs.

`void`
`CheckOut(`
`String componentName,`
`String libtype,`
`Boolean lock,  [optional]`
`Boolean savePriorVers, [optional]`
`Int16 version,  [optional]`
`String jobCard  [optional]`
`)`
 	
`void`
`CheckOut(`
`String[] componentNames,`
`String libtype,`
`Boolean lock,  [optional]`
`Boolean savePriorVers, [optional]`
`Int16 version,  [optional]`
`String jobCard  [optional]`
`)`
	Checks components out to a package from a baseline library.
	
	When checking out previous (non-zero) baseline versions, the operation is performed in batch, and a job card must be supplied. If the job card contains multiple lines, they should be separated by a newline character.
	
	If checking out multiple components, all components must belong to the same library type.

`void`
`CheckOut(`
`String componentName,`
`String libtype,`
`Boolean lock,`
`Boolean savePriorVersion,`
`ZosPromotionLevel promoLevel`
`)`
 	
`void`
`CheckOut(`
`String[] componentNames,`
`String libtype,`
`Boolean lock,`
`Boolean savePriorVersion,`
`ZosPromotionLevel promoLevel`
`)`
	Checks components out to a package from a promotion library.
	
	Component names must be specified with an extension.
	
	If checking out multiple components, all components must belong to the same library type.

`void`
`CheckOut(`
`String componentName,`
`String libtype,`
`Boolean lock,`
`Boolean savePriorVersion,`
`ZosReleaseArea area`
`)`
 	
`void`
`CheckOut(`
`String[] componentNames,`
`String libtype,`
`Boolean lock,`
`Boolean savePriorVersion,`
`ZosReleaseArea area`
`)`
	Checks components out to a package from a release area.
	
	Component names must be specified with an extension.
	
	If checking out multiple components, all components must belong to the same library type.

`void Build( String componentName, String libtype, ZosBuildInfo info, String jobCard )` `void Build( String[] componentNames, String libtype, ZosBuildInfo info, String jobCard )`	Builds a component in a package. If building multiple components, all components must belong to the same library type. If the job card contains multiple lines, they should be separated by a newline character.
`void Recompile( String componentName, String libtype, ZosBuildInfo info, String jobCard, ZosPromotionLevel level [optional] )` `void Recompile( String[] componentNames, String libtype, ZosBuildInfo info, String jobCard ZosPromotionLevel level [optional] )`	Recompiles a component in a package. If promotion level is specified, components are recompiled from promotion libraries. If recompiling multiple components, all components must belong to the same library type. If the job card contains multiple lines, they should be separated by a newline character.
`void Recompile( String componentName, String libtype, ZosBuildInfo info, String jobCard, ZosReleaseArea area [optional] )` `void Recompile( String[] componentNames, String libtype, ZosBuildInfo info, String jobCard ZZosReleaseArea area [optional] )`	Recompiles a component in a package. If release area is specified, components are recompiled from the release area. If recompiling multiple components, all components must belong to the same library type. If the job card contains multiple lines, they should be separated by a newline character.

`void`
    `Relink(`
        `String componentName,`
        `String libtype,`
        `ZosBuildInfo info,`
        `String jobCard,`
        `String targetLoadLib`
        `)`
 	
`void`
    `Relink(`
        `String[] componentNames,`
        `String libtype,`
        `ZosBuildInfo info,`
        `String jobCard,`
        `String targetLoadLib`
    `)`
	Re-links a component in a package.
	
	If re-linking multiple components, all components must belong to the same library type.
	
	If the job card contains multiple lines, they should be separated by a newline character.

`void`
    `Promote(`
        `ZosPromotionLevel level,`
        `String jobCard,`
        `ZosNameValue[] userVars,[optional]`
        `DateTime scheduleTime   [optional]`
        `)`
 	
`void`
    `Promote(`
        `ZosPromotionLevel level,`
        `String[] componentNames,`
        `String jobCard,`
        `ZosNameValue[] userVars,[optional]`
        `DateTime scheduleTime   [optional]`
        `)`
	Promotes either a full package or selected components in a package.
	Components are specified as name/type pairs.
	
	If the job card contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)
	
	scheduleTime can be used to schedule promotion for a future date and time.

`void`
`    Demote(`
`        ZosPromotionLevel level,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
 	
`void`
`    Demote(`
`        ZosPromotionLevel level,`
`        ZosNameType[] components,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
	Demotes a either a full package or selected componens in a package. Components are specified as name/type pairs.
	
	If the job card, contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    ReleasePromote(`
`        ZosPromotionLevel level,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
 	
`void`
`    ReleasePromote(`
`        ZosPromotionLevel level,`
`        String[] componentNames,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
	Promotes a either a full package or selected components from the starting release area.
	
	Components are specified as name/type pairs.
	
	If the job card contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    ReleaseDemote(`
`        ZosPromotionLevel level,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
 	
`void`
`    ReleaseDemote(`
`        ZosPromotionLevel level,`
`        ZosNameType[] components,`
`        String jobCard,`
`        ZosNameValue[] userVars [optional]`
`        )`
	Demotes a either a full package or selected components from a release.
	
	Components are specified as name/type pairs.
	
	If the job card, contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    Remove(`
`        String componentName,`
`        String libtype`
`        )`
 	
`void`
`    Remove(`
`        String[] componentNames,`
`        String libtype`
`        )`
	Removes a component from a package.
	
	If removing multiple components, all components must belong to the same library type.

`void`
`    Lock(`
`        String componentName,`
`        String libtype`
`        )`
 	
`void`
`    Lock(`
`        String[] componentNames,`
`        String libtype`
`        )`
	Locks a package component.
	
	If locking multiple components, all components must belong to the same library type.

`void`
`    Unlock(`
`        String componentName,`
`        String libtype`
`        )`
 	
`void`
`    Unlock(`
`        String[] componentNames,`
`        String libtype`
`        )`
	Unlocks a package component.
	
	If unlocking multiple components, all components must belong to the same library type.

`void`
`    Audit(`
`        ZosPackageAuditOptions options,`
`        String jobCard,`
`        String[] scopeApps,  [optional]`
`        ZosNameValue[] userVars [optional]`
`        )`
	Audits a package.
	
	If the job card, contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    ResetAuditLock()`
	Resets audit pending lock for a package.

`void`
`    Freeze(`
`        ZosNameValue[] userVars [optional]`
`        )`
	Freezes a package.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    Revert(`
`        String reason`
`        )`
	Reverts a frozen package to development status.
	
	The reason is a single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundariesintomultiplelines.

`void`
`    Unfreeze(`
`        ZosFreezeType type`
`        )`
 	
`void`
`    Unfreeze(`
`        ZosFreezeType type,`
`        String componentName,`
`        String libtype`
`        )`
 	
`void`
`    Unfreeze(`
`        ZosFreezeType type,`
`        ZosNameType[] componentNames`
`        )`
	Unfreezes selective parts of a package. The type argument specifies which type of package data is to be unfrozen.
	
	If the type specifies NonSource or SourceLoad, then non-source or source/load components are unfrozen respectively. With these two types, you can selectively unfreeze components by specifying the component names. If no component names are provided, all components of the specified type are unfrozen.
	
	If both NonSource and SourceLoad components are to be unfrozen, they must be unfrozen separately.
	
	Components are specified as name/type pairs. Component names can be specified only with types NonSource and SourceLoad.

`void`
`    Refreeze(`
`        ZosFreezeType type`
`        )`
 	
`void`
`    Refreeze(`
`        ZosFreezeType type,`
`        String componentName,`
`        String libtype`
`        )`
 	
`void`
`    Refreeze(`
`        ZosFreezeType type,`
`        ZosNameType[] componentNames`
`        )`
	Refreezes selective parts of a package. The type argument specifies which type of package data is to be refrozen.
	
	If the type specifies NonSource or SourceLoad, then non-source or source/load components are refrozen respectively. With these two types, you can selectively refreeze components by specifying the component names. If no component names are provided, all components of the specified type are refrozen.
	
	If both NonSource and SourceLoad components are to be refrozen, they must be refrozen separately.
	
	Components are specified as name/type pairs. Component names can be specified only with types NonSource and SourceLoad.

`ZosScratchRenameInfo[]`
`    GetScratchList(`
`        String libtype  [optional]`
`        )`
	Gets a list of component scratch requests in the package. The list can optionally be filtered by library type.

`ZosScratchRenameInfo[]`
`GetRenameList(`
`String libtype [optional]`
`)`
	Gets a list of component rename requests in the package. The list can optionally be filtered by library type.

`ZosScratchRenameInfo[]`
`GetScratchRenameList(`
`String libtype [optional]`
`)`
	Gets a list of component scratch and rename requests in the package. The list can optionally be filtered by library type.

`void`
`Scratch(`
`String componentName,`
`String libtype`
`)`
	Adds a component scratch request to a package. This is a request to delete the component from the baseline library.

`void`
`Rename(`
`String componentName,`
`String newComponentName,`
`String libtype`
`)`
	Adds a component rename request to a package. This is a request to rename the component in the baseline library.

`void`
`CancelScratchRename(`
`String componentName,`
`String libtype`
`)`
	Cancels a pending component scratch or rename request from a package.

`void`
`Attach(`
`String release,`
`String releaseArea,`
`)`
	Attaches a package to an ERO release.

`void`
`Approve(`
`String entity`
`)`
	Approve a package.

`void`
`Review(`
`String entity`
`)`
	Mark a package as being under review for approval.

`void`
`CheckOff(`
`String entity,`
`String comments`
`)`
	Add a list of approval check-off comments to a package.
	The comments are single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.

`void`
`Reject(`
`String entity,`
`String reason`
`)`
	Reject a package approval.
	The reason is single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.

`void`
`    Backout(`
`        String reason`
`        )`
 	
`void`
`    Backout(`
`        String reason,`
`        String site,`
`        String jobCard`
`        )`
	Back out an installed package.
	
	The reason is single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.
	
	The job card is used only when a remote site is specified. If the job card contains multiple lines, they should be separated by a newline character.

`void`
`    Detach()`
	Detaches a package from an ERO release.

`void`
`    Delete()`
	Memo-deletes a package.

`void`
`    Undelete()`
	Restores a memo-deleted package.

`String`
`    GetUserVariable(`
`        String name`
`        )`
	Gets value of a named user variable. See UserVariables property description for a list of valid user variable names.

`void`
`    SetUserVariable(`
`        String name,`
`        String value`
`        )`
	Sets value of a named user variable. See UserVariables property description for a list of valid user variable names.

`ZosPackagePromotionHistory[]`
`    GetPackagePromotionHistory(`
`        String promotionSite,  [optional]`
`        String promotionName,  [optional]`
`        Boolean siteOnly,      [optional]`
`        ZosPackagePromotionAction`
`            actionFilter,      [optional]`
`        ZosPackagePromotionStatus`
`            statusFilter       [optional]`
`        )`
	Gets a list of package promotion history records for the package.

`ZosComponentPromotionHistory[]`
`    GetComponentPromotionHistory(`
`        String promotionSite,  [optional]`
`        String promotionName,  [optional]`
`        String componsntType,  [optional]`
`        String componentName,  [optional]`
`        ZosComponentPromotionStatus`
`            statusExclude      [optional]`
`        )`
	Gets a list of component promotion history records for the package.

`ZosPromotionOverlay[]`
`    CheckPromotionOverlay(`
`        ZosPromotionLevel level,`
`        ZosNameType[]`
`            componentNames [optional]`
`        )`
	Gets a list of components that would be overwritten by a promote operation. You can, optionally, specify a list of component names to be checked. If component names are not specified, then all package components are checked.

`static`
`ZosPackage`
`    Create(`
`        ZosApplication^ application,`
`        ZosPackageInfo^ info`
`        )`
	Create a new package for an application. The package information is specified using a ZosPackageInfo object.

ZosPackage Examples

Examples of using **ZosPackage** are shown below:

C#

```
ZosPackage package;
ZosPackageSite site = package.GetSite("NEWYORK");
ZosPackageSites[] sites = package.Sites;
package.Level = ZosPackageLevel.Simple;
ZosPackageInfo info = new ZosPackageInfo(package);
info.Title = "Second package";
ZosPackage package2 = Package.Create(application, info);
```

C++

```
ZosPackage^ package;
ZosPackage site = package.GetSite("NEWYORK");
array<ZosPackage^>^ sites = package.Sites;
package->Level = ZosPackageLevel::Simple;
ZosPackageInfo^ info = gcnew ZosPackageInfo(package);
info->Title = "Second package";
ZosPackage^ package2 = Package::Create(application, info);
```

Visual Basic

```
Dim package as ZosPackage;
Dim site As ZosPackage = package.GetSite(“NEWYORK”)
Dim sites () As ZosPackage = package.Sites
package.Level = ZosPackageLevel.Simple
Dim info As New ZosPackageInfo(package);
info.Title = “Second package”
Dim package2 As ZosPackage = Package.Create(app, info);
```

Jscript

```
var package : ZosPackage;
var site : ZosPackage app = package.GetSite(“NEWYORK”);
var sites : ZosPackage [] = package.Sites;
package.Level = ZosPackageLevel.Simple;
var info : ZosPackageInfo = new ZosPackageInfo(package);
info.Title = “Second package”;
var package2 : ZosPackage = Package.Create(app, info);
```

ZosPackageApprover

The **ZosPackageApprover** object contains information describing a package approver. This object can be obtained using the **Approvers** property of **ZosPackage**.

ZosPackageApprover Properties

ZosPackageApprover exposes the following properties:

Property	Type	R/W	Description
Entity	String	R	Security system entity name.
Approver	String	R	Approver user ID.
Description	String	R	Description of approver level or function.
ApprovalOrder	Int16	R	Approver level or sequence for hierarchical approvals.
ApprovalAction	ZosPackageApprovalAction	R	Most recent approval action.
ApprovedTime	DateTime	R	Date and time approval action taken.
CheckOffList	String	R	Check off comments. The comments are a single string, but can contain multiple lines, delimited by newline characters.
RejectReasons	String	R	Reasons for package rejection. The reasons are a single string, but can contain multiple lines, delimited by newline characters.

ZosPackageComponentDirectory

The **ZosPackageComponentDirectory** object represents a Unix subdirectory within a package library. This object can be obtained using the **GetComponents** method of either **ZosPackageLibrary** or **ZosPackageComponentDirectory**.

ZosPackageComponentDirectory Properties

ZosPackageComponentDirectory exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension (inherited from ZosPackageComponentObject).
Path	String	R	Full file system path name for the component (inherited from ZosPackageComponentObject).

ZosPackageComponentDirectory Methods

ZosPackageComponentDirectory exposes the following methods:

Function	Description

ZosPackageComponentFile
    GetComponent(
        String fileName
)
	Gets a single component by file name. The file name must reside in this subdirectory level.

ZosPackageComponentObject[]
    GetComponents()
 	
ZosPackageComponentObject[]
    GetComponents(
        String nameFilter
)
 	
ZosPackageComponentObject[]
    GetComponents(
        DateTime changeTime
)
 	
ZosPackageComponentObject[]
    GetComponents(
        ZosComponentStatusFlags flags
)
 	
ZosPackageComponentObject[]
    GetComponents(
        String nameFilter,
        ZosComponentStatusFlags flags
)
 	
ZosPackageComponentObject[]
    GetComponents(
        String nameFilter,
        ZosComponentStatusFlags flags,
        DateTime changeTime
)
	Gets an array of components that belong to a package library.
	
	The list can optionally be filtered by component name and component status.
	
	This function only returns components in this subdirectory level and the array returned contains both directory and file objects.
	
	To retrieve components in lower level subdirectories, use the GetComponents method of the parent ZosPackageComponentDirectory object.
	
	nameFilter - Name filter
	
	statusFlags - Status filter flags
	
	changeTime - get components changed after the specified time

ZosPackageComponentFile

The **ZosPackageComponentFile** object represents a component in a package, and can be either a PDS member or a Unix file. This object can be obtained using the **GetComponent** or **GetComponents** methods of either **ZosPackage** or **ZosPackageLibrary**.

ZosPackageComponentFile Properties

ZosPackageComponentFile exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension. Inherited from ZosPackageComponentObject
Path	String	R	Full file system path name for the component. Inherited from ZosPackageComponentObject
IsUnix	Boolean	R	Indicates whether component is a PDS member or Unix file.
OriginalName	String	R	Original name (from development).
ComponentName	String	R	Component name.
ComponentType	String	R	Component type (library type).
Description	String	R/W	Component description
LastWriteTime	DateTime	R	Date and time component was last updated.
Version	Int16	R	Version number.
ModLevel	Int16	R	Modification level.
User	String	R	User ID who last updated the component.
BuildProc	String	R	Build procedure.
Status	ZosComponentStatus	R	Component status.
LockStatus	ZosComponentLockStatus	R	Lock status.
BuildType	ZosBuildType	R	Build type (normal, recompile, re-link).

ZosPackageComponentFile Methods

ZosPackageComponentFile exposes the following methods:

Function	Description
`ZosComponentStagingVersion[] GetStagingVersions()`	Retrieves an array of the staging versions for this component.
`void Refresh()`	Refreshes the component information.

ZosPackageComponentObject

ZosPackageComponentObject represents a file system object in a package library. This object can be a PDS member, Unix directory, or Unix file.

ZosPackageComponentObject is the base class for the **ZosPackageComponentFile** and the **ZosPackageComponentDirectory** classes. The **IsDirectory** property indicates whether the **ZosPackageComponentObject** is actually a **ZosPackageComponentDirectory** or a **ZosPackageComponentFile** object.

This object can be obtained using the **GetComponents** method of **ZosPackageLibrary**.

ZosPackageComponentObject Properties

ZosPackageComponentObject exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension.
Path	String	R	Full file system path name for the component.
IsDirectory	Boolean	R	Indicates whether the object is a directory.
IsUnix	Boolean	R	Indicates whether the object is a Unix file system object or PDS member.

ZosPackageInfo

The **ZosPackageInfo** object represents a set of package properties that can be used to create a new package.

An empty **ZosPackageInfo** object can be created using the default constructor. You can then set the desired **ZosPackageInfo** properties before using it to create a new package.

You can clone the properties of another package using the **GetInfo** method of **ZosPackage** to copy the properties of an existing package set into a new **ZosPackageInfo** object. There is also one form of the **ZosPackageInfo** constructor that initializes the properties from an existing package. This cloned **ZosPackageInfo** object can be used to create a new package after making any desired changes to its properties.

ZosPackageInfo Constructor

The default constructor can be used to create a new **ZosPackageInfo** object. Because the constructor has no arguments, you must initialize the object by setting its properties. The other constructor copies the properties from an existing package.

Constructor	Parameters
`ZosPackageInfo()`	

`ZosPackageInfo(`
`    ZosPackage package`
`    )`
	Existing package from which to copy properties.

ZosPackageInfo Properties

ZosPackageInfo exposes the following properties:

Property	Type	R/W	Description
Title	String	R/W	Package title.
RequestorName	String	R/W	Requestor name.
RequestorPhone	String	R/W	Requestor telephone number.
WorkRequest	String	R/W	Work request number or name.
Department	String	R/W	Department number or name.
SuperPackage	String	R/W	Parent complex or super package.
Release	String	R/W	Name of ERO release with which package is associated.
ReleaseArea	String	R/W	Name of starting release area for release package check in.
Level	ZosPackageLevel	R/W	Package level.
Type	ZosPackageType	R/W	Package type.
TempDuration	Int32	R/W	Temporary change duration. This property is available for temporary packages only.
ReasonCode	Int32	R/W	Reason code.

Description	String	R/W	Package description. The description is a single string, but can contain multiple lines, delimited by newline characters. When setting the description if a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.
SchedulerType	ZosSchedulerType	R/W	Scheduler type. This property is available for simple and participating packages only.
ProblemActionType	ZosProblemActionType	R/W	Problem action code. This property is available for simple and participating packages only.
OtherProblemAction	String	R/W	Other problem action. This property is available for simple and participating packages only.
ImplementationInstructions	String	R/W	Implementation instructions. The implementation instructions consist of a single string, but can contain multiple lines, delimited by newline characters. When setting the implementation instructions, if a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines. This property is available for simple and participating packages only.

ParticipatingPackages	String[]	R/W	Participating packages. The value is an array of strings, each containing a package name. This property is available for complex and super packages only.
AffectedApplications	String[]	R/W	Affected applications. The value is an array of strings, each containing an application name. This property is available for participating packages only.
PredecessorJobs	String[]	R/W	Predecessor jobs. The value is an array of strings, each containing a job name. This property is available for simple and participating packages only.
SuccessorJobs	String[]	R/W	Successor jobs. The value is an array of strings, each containing a job name. This property is available for simple and participating packages only.
UserVariables	ZosNameValue[]	R/W	User variables. Each user variable is a name/value pair. See the **UserVariables** table below for a list of valid variable names.

Sites	ZosPackageSite[]	R/W	Package site information. The value is an array of package site objects. This property is available for simple and participating packages only.
Site	ZosPackageSite	R/W	Package site information. The value is a single package site object for packages with single sites. This property is available for simple and participating packages with a single site only.

UserVariables Table

User variables are a set of name/value pairs. Each name must be one of the names in the table below.

Variable Name	Value Length
UserVarLen101 - UserVarLen199	1
UserVarLen201 - UserVarLen211	2
UserVarLen301 - UserVarLen310	3
UserVarLen401 - UserVarLen410	4
UserVarLen801 - UserVarLen810	8
UserVarLen1601 - UserVarLen1605	16
UserVarLen4401 - UserVarLen4405	44
UserVarLen7201 - UserVarLen7205	72

ZosPackageLibrary

The **ZosPackageLibrary** object represents a ChangeMan package staging library. This object can be obtained using the **GetLibrary** or **GetLibraries** methods of **ZosPackage**.

ZosPackageLibrary Properties

ZosPackageLibrary exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Library type name
Path	String	R	Full file system path name for the library
Description	String	R	Library description
DataSetName	String	R	Data set name for the library
TargetLibrary	String	R	Target build library
LikeType	ZosLikeType	R	Like library type option
StagingVersSaveOption	ZosStagingVersSaveOption	R	Staging version save option
DeferredAllocation	Boolean	R	Indicates whether allocations are deferred
DataSetType	ZosDataSetType	R	Data set type (organization)
RecordFormat	ZosRecordFormat	R	Record format
RecordLength	Int16	R	Record length
BlockSize	Int16	R	Block size
SpaceUnit	ZosSpaceUnit	R	Space unit type
PrimarySpace	Int32	R	Primary space quantity
SecondarySpace	Int32	R	Secondary space quantity
DirectoryBlocks	Int32	R	Number of directory blocks
UnitName	String	R	Unit name
Volume	String	R	Volume serial number

ZosPackageLibrary Methods

ZosPackageLibrary exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the library information.
`ZosPackageComponentFile GetComponent( String componentName )`	Gets a single component by name. For Unix libraries, componentName is the path name relative to the package library root.
`ZosPackageComponentObject[] GetComponents()` `ZosPackageComponentObject[] GetComponents( String nameFilter )` `ZosPackageComponentObject[] GetComponents( DateTime changeTime )` `ZosPackageComponentObject[] GetComponents( ZosComponentStatusFlags flags )` `ZosPackageComponentObject[] GetComponents( String nameFilter, ZosComponentStatusFlags flags )` `ZosPackageComponentObject[] GetComponents( String nameFilter, ZosComponentStatusFlags flags, DateTime changeTime )`	Gets an array of components that belong to a package library. The list can optionally be filtered by component name and component status. For Unix libraries, components are retrieved hierarchically. This function only returns components in the top level subdirectory, and the array returned contains both directory and file objects. To retrieve components in lower level subdirectories, use the GetComponents method of the parent ZosPackageComponentDirectory object. nameFilter - Name filter statusFlags - Status filter flags changeTime - get components changed after the specified time

ZosPackagePromotionHistory

The **ZosPackagePromotionHistory** object is a ChangeMan package promotion history record. The package promotion history records represent a package promotion event.

The package promotion history records can be retrieved using the **GetPackagePromotionHistory** method of **ZosPackage**.

ZosPackagePromotionHistory Properties

ZosPackagePromotionHistory exposes the following properties:

Property	Type	R/W	Description
PromotionSite	String	R	Promotion site name
PromotionName	String	R	Promotion level name
PromotionLevel	Int16	R	Promotion level number
PromotionUser	String	R	Promoting user ID
PromotionTime	DateTime	R	Date and time of the promotion
ComponentCount	Int32	R	Number of components promoted
Action	ZosPackagePromotionAction	R	Type of promotion action
Status	ZosPackagePromotionStatus	R	Promotion status flags

ZosPackageSite

The **ZosPackageSite** object represents package site information for a single site.

This object can be obtained using the Site property or Sites property of **ZosPackage**. It can also be obtained from the **GetSite** method of **ZosPackage**. A **ZosPackageSite** object can be created using the constructor and then used to update package site information.

ZosPackageSite Constructor

The following constructor can be used to initialize a new **ZosPackageSite** object:

Constructor	Parameters
`ZosPackageSite( String siteName, String primaryContactName, String primaryContactPhone, String alternateContactName, String alternateContactPhone, DateTime installStartTime, DateTime installEndTime )`	Site name Primary contact name Primary contact phone Alternate contact name Alternate contact phone Install start date and time Install end date and time
`ZosPackageSite( String primaryContactName, String primaryContactPhone, String alternateContactName, String alternateContactPhone, DateTime installStartTime, DateTime installEndTime )`	Primary contact name Primary contact phone Alternate contact name Alternate contact phone Install start date and time Install end date and time

ZosPackageSite Properties

ZosPackageSite exposes the properties below. All properties are read-only. You must construct a new object in order to change any properties.

Property	Type	R/W	Description
SiteName	String	R	Site name
PrimaryContactName	String	R	Primary contact name
PrimaryContactPhone	String	R	Primary contact phone
AlternateContactName	String	R	Alternate contact name
AlternateContactPhone	String	R	Alternate contact phone
InstallStartTime	DateTime	R	Install start date and time
InstallEndTime	DateTime	R	Install end date and time

ZosPdsMember

The **ZosPdsMember** object represents a member of a partitioned data set (PDS or PDSE). This object can be obtained using the **GetMember** or **GetMembers** methods of **ZosDataSet**. It can also be obtained using the **GetComponent** or **GetComponents** methods of **ZosBaselineLibrary**.

ZosPdsMember Properties

ZosPdsMember exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for member, including file extension
Path	String	R	Full file system path name of the member.
MemberName	String	R	Member name
CreationDate	DateTime	R	Date member was created
LastWriteTime	DateTime	R	Date and time member was last updated
CurrentLines	Int32	R	Current number of lines.
InitialLines	Int32	R	Initial number of lines.
Version	Int16	R	Version number.
ModLevel	Int16	R	Modification level.
User	String	R	User ID who last updated the member.

ZosPdsMember Methods

ZosPdsMember exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the member information.
`void Delete()`	Deletes the member.
`void Rename( String newName )`	Renames the member.

ZosPrefixMapping

The **ZosPrefixMapping** object represents a single prefix mapping. This object can be obtained using the Item property of **ZosPrefixMappings**. **ZosPrefixMapping** specifies a default data set name prefix based on data set name pattern.

ZosPrefixMapping Constructor

The following constructor can be used to initialize a new **ZosPrefixMapping** object:

Constructor	Parameters

`ZosPrefixMapping(`
`    String dsName,`
`    String prefix`
`    )`
	
	Data set name pattern
	Data set name prefix

ZosPrefixMapping Properties

ZosPrefixMapping exposes the following properties:

Property	Type	R/W	Description
DataSetName	String	R	Data set name pattern.
Prefix	String	R	Data set name prefix.

ZosPrefixMappings

The **ZosPrefixMappings** object is a collection of all data set name prefix mappings for a folder. This object is obtained using the Prefixes property of the **ZosDataSetFolder** object.

ZosPrefixMappings Properties

ZosPrefixMappings exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosPrefixMapping	R	Prefix mapping with specified index or data set name pattern.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosPrefixMappings Methods

ZosPrefixMappings exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes collection.

`ZosPrefixMapping[]`
`    ToArray()`
	Copies the entire collection to a one-dimensional array.

`void FromArray( ZosPrefixMapping[] array )`	Copies the contents of a one-dimensional array into the collection. The existing contents of the collection are completely replaced.
`Int32 Add( Int32 index, ZosPrefixMapping mapping )`	Adds a new prefix mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`Int32 Add( Int32 index, String dsName, String prefix )`	Adds a new prefix mapping. Index indicates position for new item. Specify –1 to insert at end. Returns index at which object has been added.
`void RemoveAt( Int32 index )`	Deletes a prefix mapping, specified by index.
`Boolean Remove( String name )`	Deletes a prefix mapping, specified by data set name pattern. Returns true if item was removed or false if item is not found.
`Int32 Move( Int32 indexTo, Int32 indexFrom )`	Changes the order of prefix mappings.
`Int32 FindIndex( String name )`	Searches for mapping with specified name and returns zero-based index. Returns -1 if name is not found.
`ZosPrefixMapping Find( String name )`	Searches for mapping with specified name and returns reference to object. Returns null if name is not found.

ZosPromotionLevel

The **ZosPromotionLevel** object represents a ChangeMan promotion level for a promotion site. This object can be obtained using the **GetPromotionLevel** or **GetPromotionLevels** methods of **ZosPromotionSite**.

ZosPromotionLevel Properties

ZosPromotionLevel exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File system name (level.name) for promotion level
Path	String	R	Full file system path name for the promotion level
PromotionName	String	R	Promotion level name
PromotionLevel	Int16	R	Promotion level number
SiteName	String	R	Promotion site name
SecurityEntity	String	R	Security profile (entity)
PromotionProc	String	R	Promotion procedure

ZosPromotionLevel Methods

ZosPromotionLevel exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the promotion level information.
`ZosPromotionLibrary GetLibrary( String libType )` `ZosPromotionLibrary GetLibrary( String libType ZosPromotionTarget target )`	Gets a specific promotion library by name.
`ZosPromotionLibrary[] GetLibraries()` `ZosPromotionLibrary[] GetLibraries( ZosPromotionTarget target )`	Gets an array containing the libraries for a promotion level.

ZosPromotionLibrary

The **ZosPromotionLibrary** object represents a ChangeMan library for a promotion level. This object can be obtained using the **GetLibrary** or **GetLibraries** methods of **ZosPromotionLevel**.

ZosPromotionLibrary Properties

ZosPromotionLibrary exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Library type name.
Path	String	R	Full file system path name for the library
SiteName	String	R	Promotion site name
PromotionName	String	R	Promotion level name
PromotionLevel	Int16	R	Promotion level number
IsUnix	Boolean	R	Indicates whether the library is a PDS or Unix directory
DataSetName	String	R	Data set name or Unix path name

ZosPromotionLibrary Methods

ZosPromotionLibrary exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the library information.
`ZosPdsMember GetPdsComponent( String name )`	Gets a single component of a promotion PDS library by name. Component name can be specified with or without an extension.

`ZosPdsMember[] GetPdsComponents()`
 	
`ZosPdsMember[] GetPdsComponents( String nameFilter )`
 	
`ZosPdsMember[] GetPdsComponents( DateTime changeTime )`
 	
`ZosPdsMember[] GetPdsComponents( String nameFilter, DateTime changeTime )`
	Gets an array of components that belong to a promotion PDS library. The list can optionally be filtered by component name.
	
	nameFilter - Componenent name filter (pattern)
	
	changeTime - get components changed after the specified time

`ZosUnixObject[]`
`    GetUnixComponent(`
`        String name`
`        )`
	Gets a single component of a promotion Unix library by file name.

`ZosUnixObject[]`
`    GetUnixComponents()`
 	
`ZosUnixObject[]`
`   GetUnixComponents(`
`       DateTime changeTime`
`       )`
 	
`ZosUnixObject[]`
`    GetUnixComponents(`
`        String dirName`
`        )`
 	
`ZosUnixObject[]`
`    GetUnixComponents(`
`        String dirName,`
`        String nameFilter`
`        )`
 	
`ZosUnixObject[]`
`    GetUnixComponents(`
`        String dirName,`
`        String nameFilter,`
`        DateTime changeTime`
`        )`
	Gets an array of components that belong to a promotion Unix library. The list can optionally be filtered by component name.
	
	For Unix libraries, components are retrieved hierarchically. This function only returns components in a specified subdirectory. The array returned contains both directory and file objects.
	
	dirName - Subdirectory name
	
	nameFilter - Componenent name filter (pattern)
	
	changeTime - get components changed after the specified time

ZosPromotionOverlay

The **ZosPromotionOverlay** object contains information about a ChangeMan component that would be overwritten by a promote operation. The promotion overlay entries can be retrieved using the **CheckPromotionOverlay** method of **ZosPackage**.

ZosPromotionOverlay Properties

ZosPromotionOverlay exposes the following properties:

Property	Type	R/W	Description
ComponentType	String	R	Component type
ComponentName	String	R	Component name
Package	String	R	Package name
Release	String	R	Release name for package (ERO)
PromotionUser	String	R	Promoting user ID
PromotionTime	DateTime	R	Date and time of the promotion
OverlayStatus	ZosPromotionOverlayStatus	R	Overlay status
PackageStatus	ZosPackageStatus	R	Package status
IsRestaged	Boolean	R	Indicates whether component has been restaged

ZosPromotionSite

The **ZosPromotionSite** object represents a ChangeMan promotion site for an application. This object can be obtained using the **GetPromotionSite** or **GetPromotionSites** methods of **ZosApplication**.

ZosPromotionSite Properties

ZosPromotionSite exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Promotion site name
Path	String	R	Full file system path name for the promotion site
LocalReaderClass	Char	R	Local internal reader class

RemoteReaderClass	Char	R	Remote (site) internal reader class
ForcePriorSiteDemote	Boolean	R	Indicates whether to force demotion of prior sites

ZosPromotionSite Methods

ZosPromotionSite exposes the following methods:

Function	Description
void Refresh()	Refreshes the promotion site information.
ZosPromotionLevel GetPromotionLevel(String name) ZosPromotionLevel GetPromotionLevel(Int16 level)	Gets a specific promotion level by either name or level number.
ZosPromotionLevels[] GetPromotionLevels(String filter [optional])	Gets an array containing the promotion sites for the application. Results can be optionally filtered by the folder name (level.name) using wild characters.

ZosQueryImpactResult

The **ZosQueryImpactResult** object shows a result from a Query Impact operation. This is the same functionality as the ChangeMan ZMF “Impact Analysis” and “Bill of Materials” options.

The query impact results are returned by the **QueryImpact** method of **ZosChangeManInstance**.

ZosQueryImpactResult Properties

ZosQueryImpactResult exposes the following properties:

Property	Type	R/W	Description
ComponentName	String	R	Component name.

ApplicationComponentTypes	String	R	Application and component types string. String is in the following format: "app1:typ1 app2:typ2 ... appN:typN".
Bun	UInt32	R	Baseline ID. A number the uniquely identifies a baseline library.

ZosRelease

The **ZosRelease** object represents a ChangeMan ZMF release (ERO). This object can be obtained using either the **GetRelease** method or the **GetReleases** method of **ZosChangeManInstance**.

ZosRelease Properties

ZosRelease exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the release.
Path	String	R	Full path name of the release.
ChangeManInstance	ZosChangeManInstance	R	ChangeMan instance.
CreatorUserID	String	R	Creator user ID.
Description	String	R/W	Release description.
RequestorName	String	R/W	Requestor name.
RequestorPhone	String	R/W	Requestor telephone number.
WorkRequest	String	R/W	Work request number or name.
Department	String	R/W	Department number or name.
ImplementationInstructions	String	R/W	Parent super or complex package.
OtherProblemAction	String	R/W	Other problem action description.
ProblemActionType	ZosProblemActionType	R/W	Problem action type.
Status	ZosReleaseStatus	R	Release status.
AuditReturnCode	Int32	R	Audit return code.
FromInstallTime	DateTime	R	Starting install date and time.

ToInstallTime	DateTime	R	Ending install date and time.
CreatedTime	DateTime	R	Date and time release was created.
BlockedTime	DateTime	R	Date and time release was blocked.
ApprovedTime	DateTime	R	Date and time release was approved.
RejectedTime	DateTime	R	Date and time release was rejected.
RevertedTime	DateTime	R	Date and time release was reverted.
InstalledTime	DateTime	R	Date and time release was installed.
BackedOutTime	DateTime	R	Date and time release was backed out.
BaselinedTime	DateTime	R	Date and time release was baselined.
MemoDeletedTime	DateTime	R	Date and time release was memo deleted.
Approvers	ZosReleaseApprover[]	R	Approvers for the release.

ZosRelease Methods

ZosRelease exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the release information.
`String[] GetApplicationNames()`	Gets an array of application names that are joined to this release.
`ZosReleaseArea GetReleaseArea( String name )`	Get a release area by area name.
`ZosReleaseArea[] GetReleaseAreas()`	Gets array of all the release areas.
`ZosPackage GetPackage( String name )`	Get a package name.
`ZosPackage[] GetPackages()`	Gets array of all packages attached to the release.
`void SetInstallTime( DateTime fromTime, DateTime toTime )`	Updates start and end install times for the release.

`void`
`    Block()`
	Blocks the release.

`void`
`    Unblock()`
	Unblocks the release.

`void`
`    Revert(`
`        String reason`
`        )`
 	
`void`
`    Revert(`
`        String reason,`
`        String site,`
`        String jobCard`
`        )`
	Reverts a release to development status.
	
	The reason is a single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.
	
	The job card is used only when a remote site is specified. If the job card, contains multiple lines, they should be separated by a newline character.

`void`
`    Approve(`
`        String entity`
`        )`
	Approve a release.

`void`
`    Reject(`
`        String entity,`
`        String reason`
`        )`
	Reject a release approval.
	
	The reason is single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundaries into multiple lines.

`ZosTestReleaseResult[]`
`   Test()`
 	
`ZosTestReleaseResult[]`
`    Test(`
`       Boolean cleanupEmptyPackage,`
`       Boolean cleanupCmpFromDiffPkg,`
`       Boolean cleanupNotCheckedInCmp,`
`       String  jobCard)`
	Test release for errors, and optionally, perform automatic cleanup of packages and components.
	
	cleanupEmptyPackage - Detach empty packages from the release
	
	cleanupCmpFromDiffPkg - If different versions of a component exist in different packages, the version not checked into the release will be deleted from the package
	
	cleanupNotCheckedInCmp - Delete components from the package, if they were not checked into the release
	
	jobCard - Jobcards to use for automatic cleanup

ZosRelease Examples

Examples of using **ZosRelease** are shown below.

C#
```
ZosRelease release;
ZosReleaseArea area = release.GetArea(“QA”);
ZosReleasePackage[] packages = release.GetPackages();
release.RequestorName = “Mickey Mouse”;
release.Revert(“Terrible design”);
```

C++
```
ZosRelease^ release;
ZosReleaseArea area = release->GetArea(“QA”);
array<ZosReleasePackage^>^ packages = release->GetPackages();
release->RequestorName = “Mickey Mouse”;
release->Revert(“Terrible design”);
```

Visual Basic
```
Dim release as ZosRelease
Dim area As ZosReleaseArea = release.GetSite(“QA”)
Dim packages () As ZosPackage = release.GetPackages()
release.RequestorName = “Mickey Mouse”
release.Revert(“Terrible design”)
```

Jscript
```
var release : ZosRelease;
var area: ZosReleaseArea = release.GetSite(“QA”);
var packages : ZosPackage[] = release.GetPackages();
release.RequestorName = “Mickey Mouse”;
release.Revert(“Terrible design”);
```

ZosReleaseApprover

The **ZosReleaseApprover** object contains information describing a release approver. This object can be obtained using the **Approvers** property of **ZosRelease** or the **Approvers** property of **ZosReleaseArea**.

ZosReleaseApprover Properties

ZosReleaseApprover exposes the following properties:

Property	Type	R/W	Description
Entity	String	R	Security system entity name.
Approver	String	R	Approver user ID.

Description	String	R	Description of approver level or function.
ApprovalOrder	Int16	R	Approver level or sequence for hierarchical approvals.
ApprovalAction	ZosPackageApproval Action	R	Most recent approval action.
ApprovedTime	DateTime	R	Date and time approval action taken.
RejectReasons	String	R	Reasons for rejection. The reasons are a single string, but can contain multiple lines, delimited by newline characters.
IsAreaCheckInApprover	Boolean	R	Indicates whether check in approver.
IsAreaCheckOffApprover	Boolean	R	Indicates whether check off approver.
IsApproverNotified	Boolean	R	Indicates whether approver has been notified.

ZosReleaseArea

The **ZosReleaseArea** object represents a ChangeMan ZMF release area (ERO). This object can be obtained using either the **GetReleaseArea** method or the **GetReleaseAreas** method of **ZosRelease**.

ZosReleaseArea Properties

ZosReleaseArea exposes the following properties:

Property	Type	R/W	Description
Release	ZosRelease	R	Parent release object.
Name	String	R	Release area name.
Path	String	R	Release area full path name
AreaType	ZosReleaseAreaType	R	Release area type.
Status	ZosReleaseAreaStatus	R	Release area status.
NextArea	String	R	Next release area.
PriorArea	String	R	Prior release area.
Description	String	R	Release area description.

StepNumber	Int16	R	Release area sequence number.
AuditReturnCode	Int32	R	Audit return code.
Approvers	ZosReleaseApprover[]	R	Approvers for the release area.

ZosReleaseArea Methods

ZosReleaseArea exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the release area information.

`ZosReleaseLibrary`
`    GetLibrary(`
`        String appName,`
`        String libType`
`        )`
	Gets a release library by name.

`ZosReleaseLibrary`
`    GetLibraries(`
`        String appName`
`        )`
	Gets an array containing the release libraries for the application.

`ZosPromotionSite`
`    GetPromotionSite(`
`        String appName,`
`        String siteName`
`        )`
	Gets a promotion site by name.

`ZosPromotionSite`
`    GetPromotionSites(`
`        String appName`
`        )`
	Gets an array containing the promotion sites for the application.

`ZosPromotionLevel`
`    GetPromotionLevel(`
`        String appName,`
`        String siteName,`
`        String promotionName`
`        )`
 	
`ZosPromotionLevel`
`    GetPromotionLevel(`
`        String appName,`
`        String siteName,`
`        Int16 promotionLevel`
`        )`
	Gets a single promotion level given the site name and promotion level number or name.

`ZosCheckInStatus[]`
`    CheckIn(`
`        String appName,`
`        ZosNameType[] componentNames,`
`        Boolean rep,      [optional]`
`        Boolean eligOnly  [optional]`
`        )`
	Checks in components from on release area to the next release area. Component names are specified as name and type pairs.

`ZosRetrieveStatus[]`
`    Retrieve(`
`        String appName,`
`        String package`
`        )`
 	
`ZosRetrieveStatus[]`
`    Retrieve(`
`        String appName,`
`        ZosNameType[] componentNames`
`        )`
 	
`ZosRetrieveStatus[]`
`    Retrieve(`
`        String appName,`
`        String package,`
`        ZosNameType[] componentNames`
`        )`
	Retrieve removes components from a release area. Component names are specified as name and type pairs.

`void`
`    Promote(`
`        String appName,`
`        ZosPromotionLevel level,`
`        String jobCard,`
`       ZosNameValue[] uVars [optional]`
`        )`
 	
`void`
`    Promote(`
`        String appName,`
`        ZosPromotionLevel level,`
`        ZosNameType[] componentNames,`
`        String jobCard,`
`       ZosNameValue[] uVars [optional]`
`        )`
	Promotes a either a full package or selected components from the starting release area. Component names are specified as name and type pairs.
	
	If the job card, contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    Demote(`
`        String appName,`
`        ZosPromotionLevel level,`
`        String jobCard,`
`       ZosNameValue[] uVars [optional]`
`        )`
 	
`void`
`    Demote(`
`        String appName,`
`        ZosPromotionLevel level,`
`        ZosNameType[] componentNames,`
`        String jobCard,`
`       ZosNameValue[] uVars [optional]`
`        )`
	Demotes a either a full package or selected components from a release area. Component names are specified as name and type pairs.
	
	If the job card, contains multiple lines, they should be separated by a newline character.
	
	If user variables are specified, each is a name/value pair. Each name must be one of the following:
	
	UserVariable01 - UserVariable05 (length 8)
	UserVariable05 - UserVariable10 (length 72)

`void`
`    Audit(`
`       ZosAuditReleaseAreaOptions opt,`
`        String jobCard`
`        )`
	Audits a release area.
	
	If the job card, contains multiple lines, they should be separated by a newline character.

`void`
`    Block()`
	Blocks the release area.

`void`
`    Unblock()`
	Unblocks the release area.

`void`
`    NotifyCheckIn()`
	Notifies approvers to begin the approval process for check in.

`void`
`    NotifyCheckOff()`
	Notifies approvers to begin the approval process for check off.

`void`
`    ResetApprovals()`
	Resets the check-in approvals.

`void`
`    ApproveCheckIn(`
`        String entity`
`        )`
	Approves a release area for check in.

`void`
`    ApproveCheckOff(`
`        String entity`
`        )`
	Approves a release area for check off.

`void`
`    RejectCheckIn(`
`        String entity,`
`        String reason`
`        )`
	Reject a release area for check in.
	
	The reason is single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundariesintomultiplelines.

`void`
`    RejectCheckOff(`
`        String entity,`
`        String reason`
`        )`
	Reject a release area for check off.
	
	The reason is single string, but can contain multiple lines, delimited by newline characters. If a line exceeds 72 characters, the text will automatically be split on word boundariesintomultiplelines.

`ZosTestReleaseResult[]`
`    Test()`
 	
`ZosTestReleaseResult[]`
`    Test(`
`        String package`
`        )`
 	
`ZosTestReleaseResult[]`
`    Test(`
`        String package,`
`        Boolean excludePackage`
`        )`
	Test release area for errors.
	
	package - Package name pattern to include or exclude (optional)
	
	excludePackage - Exclude packages that match the package name pattern

ZosReleaseArea Examples

Examples of using **ZosReleaseArea** are shown below. ComponentDirectory

C#
```
ZosReleaseArea area;
ZosReleaseLibrary lib = area.GetLibrary(“APPX”, “SRC”);
ZosReleaseLibrary[] libs = area.GetLibraries(“APPX”);
area.Description = “Unit test”;
area.Retrieve(“APPX”, “APPX000123”);
```

C++
```
ZosReleaseArea^ area;
ZosReleaseLibrary lib = area->GetLibrary(“APPX”, “SRC”);
array<ZosReleaseLibrary^>^ libs = area->GetLibraries(“SRC”);
area->Description = “Unit test”;
area->Retrieve(“APPX”, “APPX000123”);
```

Visual Basic
```
Dim area as ZosReleaseArea
Dim lib As ZosReleaseLibrary = area.GetLibrary(“APPX”, “SRC”)
Dim libs () As ZosReleaseLibrary = area.GetLibraries(“SRC”)
area.Description = “Unit test”
area.Retrieve(“APPX”, “APPX000123”)
```

Jscript
```
var area : ZosReleaseArea;
var lib: ZosReleaseLibrary = area.GetLibrary(“APPX”, “SRC”);
var libs : ZosReleaseLibrary [] = area.GetLibraries(“SRC”);
area.Description = “Unit test”;
area.Retrieve(“APPX”, “APPX000123”);
```

ZosReleaseComponentDirectory

The **ZosReleaseComponentDirectory** object represents a Unix subdirectory within a release library. This object can be obtained using the **GetComponents** method of either **ZosReleaseLibrary** or **ZosReleaseComponentDirectory**.

ZosReleaseComponentDirectory Properties

ZosReleaseComponentDirectory exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension (inherited from ZosReleaseComponentObject).
Path	String	R	Full file system path name for the component (inherited from ZosReleaseComponentObject).

ZosReleaseComponentDirectory Methods

ZosReleaseComponentDirectory exposes the following methods:

Function	Description

`ZosReleaseComponentFile`
`    GetComponent(`
`        String fileName`
`        )`
	Gets a single component by file name. The file name must reside in this subdirectory level.

`ZosReleaseComponentObject[]`
`    GetComponents()`
 	
`ZosReleaseComponentObject[]`
`    GetComponents(`
`        String nameFilter`
`        )`
	Gets an array of components that belong to a release library.
	
	The list can optionally be filtered by component name.
	
	This function only returns components in this subdirectory level and the array returned contains both directory and file objects.
	
	To retrieve components in lower level subdirectories, use the GetComponents method of the parent ZosReleaseComponentDirectory object.

ZosReleaseComponentFile

The **ZosReleaseComponentFile** object represents a component in a release area, and can be either a PDS member or a Unix file. This object can be obtained using the **GetComponent** or **GetComponents** methods of either **ZosReleaseArea** or **ZosReleaseLibrary**.

ZosReleaseComponentFile Properties

ZosReleaseComponentFile exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension. Inherited from ZosReleaseComponentObject
Path	String	R	Full file system path name for the component. Inherited from ZosReleaseComponentObject
IsUnix	Boolean	R	Indicates whether component is a PDS member or Unix file.
ComponentName	String	R	Component name.
ComponentType	String	R	Component type (library type).
LikeType	ZosLikeType	R	Like type
DataSetName	String	R	Data set name for release area library
Package	String	R	Package from which component originated.
CheckInTime	DateTime	R	Date and time component was last checked in to release area.

ZosReleaseComponentFile Methods

ZosReleaseComponentFile exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the component information.

ZosReleaseComponentObject

ZosReleaseComponentObject represents a file system object in a release area library. This object can be a PDS member, Unix directory, or Unix file.

ZosReleaseComponentObject is the base class for the **ZosReleaseComponentFile** and the **ZosReleaseComponentDirectory** classes. The **IsDirectory** property indicates whether the **ZosReleaseComponentObject** is actually a **ZosReleaseComponentDirectory** or a **ZosReleaseComponentFile** object.

This object can be obtained using the **GetComponents** method of **ZosReleaseLibrary**.

ZosReleaseComponentObject Properties

ZosReleaseComponentObject exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension.
Path	String	R	Full file system path name for the component.
IsDirectory	Boolean	R	Indicates whether the object is a directory.
IsUnix	Boolean	R	Indicates whether the object is a Unix file system object or PDS member.

ZosReleaseLibrary

The **ZosReleaseLibrary** object represents a ChangeMan release area library. This object can be obtained using the **GetLibrary** or **GetLibraries** methods of **ZosReleaseArea**.

ZosReleaseLibrary Properties

ZosReleaseLibrary exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Library type name
Path	String	R	Full file system path name for the library
Description	String	R	Library description
DataSetName	String	R	Data set name for the library
TargetLibrary	String	R	Target build library
LikeType	ZosLikeType	R	Like library type option
StagingVersSaveOption	ZosStagingVersSaveOption	R	Staging version save option
DeferredAllocation	Boolean	R	Indicates whether allocations are deferred
DataSetType	ZosDataSetType	R	Data set type (organization)
RecordFormat	ZosRecordFormat	R	Record format
RecordLength	Int16	R	Record length
BlockSize	Int16	R	Block size
SpaceUnit	ZosSpaceUnit	R	Space unit type
PrimarySpace	Int32	R	Primary space quantity
SecondarySpace	Int32	R	Secondary space quantity
DirectoryBlocks	Int32	R	Number of directory blocks
UnitName	String	R	Unit name
Volume	String	R	Volume serial number

ZosReleaseLibrary Methods

ZosReleaseLibrary exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes the library information.

`ZosReleaseComponentFile`
`    GetComponent(`
`        String componentName`
`        )`
	Gets a single component by name.
	For Unix libraries, componentName is the path name relative to the release library root.

`ZosReleaseComponentObject[]`
`    GetComponents()`
 	
`ZosReleaseComponentObject[]`
`    GetComponents(`
`        String nameFilter`
`        )`
	Gets an array of components that belong to a release library.
	The list can optionally be filtered by component name.
	For Unix libraries, components are retrieved hierarchically. This function only returns components in the top level subdirectory, and the array returned contains both directory and file objects.
	To retrieve components in lower level subdirectories, use the GetComponents method of the parent ZosReleaseComponentDirectory object.

ZosRetrieveStatus

The **ZosRetrieveStatus** object shows status information for a release area retrieve operation for a particular component.

The retrieve status is returned by the **Retrieve** method of **ZosReleaseArea**.

ZosRetrieveStatus Properties

ZosRetrieveStatus exposes the following properties:

Property	Type	R/W	Description
Release	String	R	Release name.
ReleaseArea	String	R	Release area name.
Application	String	R	Application name.
ComponentName	String	R	Target component name.

ComponentType	String	R	Component type (library type).
User	String	R	User ID who last updated the component.
CheckInTime	DateTime	R	Date and time component was checked in.
Status	String	R	Status description.

ZosScratchRenameInfo

The **ZosScratchRenamInfo** object contains information about a component scratch or rename request in a package. You can obtain an array of **ZosScratchRenamInfo** objects using the **GetScratchList**, **GetRenameList**, or **GetScratchRenameList** methods of **ZosPackage**.

A **ZosScratchRenamInfo** object can represent either a scratch request or a rename request. You can determine the type of request by inspecting the **NewComponentName** property. The **NewComponentName** property is a null string for a scratch request; otherwise, the request is for a rename operation.

ZosScratchRename Properties

ZosScratchRenameInfo exposes the following properties:

Property	Type	R/W	Description
Name	String	R	File name for component, including file extension.
ComponentName	String	R	Component name.
NewComponentName	String	R	New component name for a rename operation. If NewComponentName is a null string, then this represents a scratch operation.
ComponentType	String	R	Component type (library type).
LastWriteTime	DateTime	R	Date and time component was last updated.
User	String	R	Updater user ID
ComponentStatus	ZosComponentStatus	R	Component status.

ZosServer

The **ZosServer** object represents a single connection to a server. This object can be obtained using the Servers property of **ZosNetwork** or the Item property of **ZosServers**.

Normally, there is only one user ID logged on to the each server from the desktop at a time. However, in a server application, there may be a requirement to have more than one user ID logged onto the same server at the same time. You can accomplish this by using alternate connections to the server. Each server can have alternate connections, with connection IDs numbered 1 – 255. The default connection has a connection ID of 0.

The **Connection** property of the **ZosServer** object contains the connection ID. The **Connection** property is read-only and you must create a new **ZosServer** object to change the connection ID.

There are two ways to create a **ZosServer** object with an alternate connection ID:

- Specify a connection ID when using the **Item** property of **ZosServers**.
- Call the **NewConnection** method of **ZosServer** to create a new server object with a different connection ID.

For more information on alternate connections, see the section entitled "Alternate Connections" on page 34.

ZossServer Properties

ZosServer exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the server.
Connection	Int16	R	Connection ID (default connection is 0)
Path	String	R	File system path name for server.
Description	String	R/W	Server description (volume label).
Address	String	R/W	I/P address or DNS name.
Port	Int32	R/W	I/P port number.
Secure	Boolean?	R/W	Enables TLS security for all ports on this server, including ChangeMan ports.
HostCodePage	Int32	R/W	Server EBCDIC code page

PasswordPhrase	Boolean	R/W	Indicates whether password phrases (long passwords) are allowed.
DisablePort	Boolean	R/W	Disables XCH port. If the port is disabled, the “DataSets”, “Jobs”, and “Unix” folders are hidden and unavailable.
MaxSessions	Int32	R/W	Maximum number of concurrent sessions.
UtcOffset	TimeSpan	R	Time zone offset from UTC. UTC + UtcOffset = Local
Today	DateTime	R	Current date on the server.
User	String	R	User ID of currently logged on user.
Groups	String[]	R	Array of security group names to which the currently logged on user is connected. *(requires SerNet 7.1.3+)*
DataSetFolders	ZosDataSetFolders	R	Collection of top-level data set folders for server.
JobFolders	ZosJobFolders	R	Collection of top-level job folders for server.
UnixFolders	ZosUnixFolders	R	Collection of Unix folders for server.
UnixRootDirectory	ZosUnixDirectory	R	Root directory for the Unix file system.
ChangeManInstances	ZosChangeManInstances	R	Collection of all ChangeMan instances for server.
DataSetFileFormatMappings	ZosFileFormatMappings	R	Collection of data set file format mappings for server.

UnixFileFormatMappings	ZosFileFormatMappings	R	Collection of Unix file format mappings for server. *This property is available for version 7.1+ servers only.*
LibTypeMappings	ZosLibTypeMappings	R	Collection of all library type mappings for server.
FileExtensionMappings	ZosFileExtensionMappings	R	Collection of all file-extension mappings for server.
DataSetProfiles	ZosDataSetProfiles	R	Collection of all data set profiles for server.

ZosServer Methods

ZosServer exposes the following methods:

Function	Description
`ZosServer NewConnection( Int16 connection )`	Create a new server object for the same server, but with a different connection ID. The connection ID must be 0 – 255.
`void Logon( String userID, [optional] String password, [optional] String newPassword [optional] )`	Logon to server.
`void Logoff()`	Logoff from server.
`void NotifyChange( String dsName, String memberName [optional] )`	Notifies file system driver that a data set or member has been created or deleted by an external process.

`ZosJesJob[]`
`    SubmitJcl(`
`        String fileName,`
`        Boolean notify     [optional]`
`        )`
	Submits JCL to the server. This routine returns an array of ZosJesJob objects representing the submitted jobs. A single JCL file can contain multiple jobs. The job IDs can be obtained from the JobID property of the ZosJesJob object.
	
	The notify argument is optional. Specify **true** to add a notify job step to the submitted JCL. A notify job step will send you a message indicating the highest return code for the job.

`void`
`    SubmitXml(`
`        String inputFileName,`
`        String outputFileName`
`        )`
	Submit XML request to server. Must be a SerNet XML service, rather than a ChangeMan XML service.

`ZosUnixObject`
`    GetUnixObject(`
`        String path`
`        )`
	Gets a Unix directory, file, or symbolic link object, given the Unix path name.

`ZosDataSet`
`    GetDataSet(`
`        String dsName`
`        )`
	Gets a data set by name.

`ZosJesJob`
`    GetJesJob(`
`        String fullname`
`        )`
	Gets a JES job by its fully qualified name in the following form: ***jobname.jobid***

`ZosJesJob`
`    GetJesJob(`
`        String jobname,`
`        String jobid`
`        )`
	Gets a JES job by job name and job ID.

ZosServer Examples

Examples of using **ZosServer** are shown below.

C#
```
ZosServer server;
String userID = server.User;
server.Address = “192.11.23.66”;
server.Logon(“USR001”, “password”);
server.NotifyChange(“USR001.NEW.DATA”, “MEMBER1”);
server.SubmitJcl(“C:\\JCL\\Print.jcl”);
```

C++
```
ZosServer^ server;
String^ userID = server->User;
Server->Address = “192.11.23.66”;
Server->Logon(“USR001”, “password”);
Server->NotifyChange(“USR001.NEW.DATA”, “MEMBER1”);
Server->SubmitJcl(“C:\\JCL\\Print.jcl”);
```

Visual Basic
```
Dim server As ZosServer
Dim userID As String server.User
server.Address = “192.11.23.66”
server.Logon(“USR001”, “password”)
server.NotifyChange(“USR001.NEW.DATA”, “MEMBER1”)
server.SubmitJcl(“C:\JCL\Print.jcl”)
```

Jscript
```
var server : ZosServer;
var userID : String = server.User;
server.Address = “192.11.23.66”;
server.Logon(“USR001”, “password”);
server.NotifyChange(“USR001.NEW.DATA”, “MEMBER1”);
server.SubmitJcl(“C:\\JCL\\Print.jcl”);
```

ZosServers

The **ZosServers** object is a collection of all servers in the ZDD Network. This object is obtained using the Servers property of the **ZosNetwork** object.

ZosServers Properties

ZosServers exposes the following properties:

Property	Type	R/W	Description
[index] **[name]** **[name, connID]**	ZosServer	R	Server with specified name or index. You can, optionally specify an alternate connection ID.

Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosServers Methods

ZosServers exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes collection.

`Int32`
`    Add(`
`        String name,`
`        String address,`
`        Int32 port,`
`        Int32 codePage,        [optional]`
`        String description,    [optional]`
`        Int32 maxSessions,     [optional]`
`        Boolean passPhrase,    [optional]`
`        Boolean secure         [optional]`
`        )`
	Adds a new server. Returns index at which object has been added.
	
	If you specify port number 0, the port will be disabled. The "DataSets", "Jobs", and "Unix" folders are not available if the port is disabled.
	
	If you specify secure, TLS security will be enabled for all ports, including ChangeMan ports.

`Boolean`
`    Remove(`
`        String name`
`        )`
	Deletes a server. Returns true if server was removed or false if server is not found.

`Int32`
`    FindIndex(`
`        String name`
`        )`
	Searches for server with specified name and returns zero-based index. Returns -1 if name is not found.

`ZosServer`
`    Find(`
`        String name`
`        )`
	Searches for server with specified name and returns reference to object. Returns null if name is not found.

ZosServers Examples

Examples of using **ZosServers** are shown below.

C#
```
ZosServers servers = network.Servers;
int count = servers.Count;
ZosServer server = network.Servers[“SYSA”];
servers.Add("Server1", "172.20.20.1", 5000, 1140, "Test");
servers.Remove(“Server1”);
```

C++
```
ZosServers^ servers = network->Servers;
int count = servers->Count;
ZosServer^ server = network.Servers[“SYSA”];
Servers->Add("Server1", "172.20.20.1", 5000, 1140, "Test");
Servers->Remove(“Server1”);
```

Visual Basic
```
Dim servers As ZosServers = network.Servers
Dim count As Integer count = servers.Count
Dim server as ZosServer = servers(“SYSA”)
servers.Add("Server1", "172.20.20.1", 5000, 1140, "Test")
servers.Remove(“Server1”)
```

Jscript
```
var servers : ZosServers = network.Servers;
var count : int = servers.count;
var server : ZosServer = servers[“SYSA”];
servers.Add("Server1", "172.20.20.1", 5000, 1140, "Test");
servers.Remove(“Server1”);
```

ZosTestReleaseResult

The **ZosTestReleaseResult** object shows a result from a Test Release or Test Area operation.

The test results are returned by the **Test** method of **ZosRelease** and by the **Test** method of **ZosReleaseArea**.

ZosTestReleaseResult Properties

ZosCheckInStatus exposes the following properties:

Property	Type	R/W	Description
Release	String	R	Release name.
ReleaseArea	String	R	Release area name.

ComponentType	String	R	Failing component type (library type).
ComponentName	String	R	Failing component name.
Package	String	R	Failing package name.
User	String	R	Failing component user ID.
OriginPackage	String	R	Originating package name.
OriginUser	String	R	Originating component user ID.
SourceComponentType	String	R	Source component type (library type).
SourceComponentName	String	R	Source component name.
Reason	String	R	Reason for failure description.
ReasonCode	Char	R	Failure reason code.
PackageStatus	ZosPackageStatus	R	Package status.

ZosUnixDirectory

The **ZosUnixDirectory** object represents a Unix directory. The **ZosUnixDirectory** class is derived from **ZosUnixObject**. The root directory for the Unix file system can be obtained from the **UnixRootDirectory** property of **ZosServer**. The directory represented by a Unix folder can be obtained from the **TargetDirectory** property of **ZosUnixFolder**.

ZosUnixDirectory Properties

ZosUnixDirectory exposes the following properties, all of which are inherited from **ZosUnixObject**.

Property	Type	R/W	Description
Name	String	R	Directory name. *(Inherited from ZosUnixObject)*.
Path	String	R	Full path name of the directory (for example `\\server\Unix\u\judy`). *(Inherited from ZosUnixObject)*.
UnixPath	String	R	Unix file system path name (for example `/u/judy`). *(Inherited from ZosUnixObject)*.
FileType	ZosUnixFileType	R	Type of Unix file system object, which is always Directory for this type of object. *(Inherited from ZosUnixObject)*.
CreationTime	DateTime	R	Creation time. *(Inherited from ZosUnixObject)*.
LastWriteTime	DateTime	R	Last update time. *(Inherited from ZosUnixObject)*.
LastAccessTime	DateTime	R	Last access time. *(Inherited from ZosUnixObject).*
User	String	R	User owner. *(Inherited from ZosUnixObject).*
Group	String	R	Group owner. *(Inherited from ZosUnixObject)*.
UserAccess	ZosUnixAccess	R/W	User access permissions. *(Inherited from ZosUnixObject)*.
GroupAccess	ZosUnixAccess	R/W	Group access permissions. *(Inherited from ZosUnixObject)*.
OtherAccess	ZosUnixAccess	R/W	Other access permissions. *(Inherited from ZosUnixObject)*.

ZosUnixDirectory Methods

ZosUnixDirectory exposes the following methods, some of which are inherited from **ZosUnixObject**.

Function	Description

`void`
`    Refresh()`
	Refreshes the Unix file system information.
	(Inherited from ZosUnixObject).

`Boolean`
`    CheckAccess(`
`        ZosUnixAccessCheck flags`
`        )`
	Checks whether or not the user has the specified access permissions for the Unix directory.
	(Inherited from ZosUnixObject).

`ZosUnixObject`
`    GetObject(`
`        String name`
`        )`
	Gets a single file system object from the directory. The object can be a directory, a file, or a symbolic link.

`ZosUnixObject[]`
`    GetObjects()`
 	
`ZosUnixObject[]`
`    GetObjects(`
`        String nameFilter`
`        )`
 	
`ZosUnixObject[]`
`    GetObjects(`
`        DateTime changeTime`
`        )`
 	
`ZosUnixObject[]`
`    GetObjects(`
`        String nameFilter,`
`        DateTime changeTime`
`        )`
	Gets an array of file system objects belonging to a Unix directory. The list can optionally be filtered.
	
	nameFilter - Name filter
	
	changeTime - get files c0hanged after the specified time

`void`
`    Remove(`
`        Boolean deleteContents`
`        )`
	Removes the directory and, optionally, deletes the directory contents.

`void`
    `Rename(`
        `String newName`
        `)`
	Renames the directory.
	(Inherited from ZosUnixObject).

`static`
`ZosUnixDirectory`
    `Create(`
        `ZosUnixDirectory parent,`
        `String name,`
        `ZosUnixAccess userAccess,`
        `ZosUnixAccess groupAccess,`
        `ZosUnixAccess otherAccess`
        `)`
	Creates a new directory.

ZosUnixDirectory Examples

Examples of using **ZosUnixDirectory** are shown below.

C#

```
ZosUnixDirectory dir;
ZosUnixObject file = dir.GetObject(“WarAndPeace.txt”);
ZosUnixObject[] files = dir.GetObjects(“X*”);
dir.Rename(“Garbage”);
dir.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write;
```

C++

```
ZosUnixDirectory^ dir;
ZosUnixObject^ file = dir.GetObject(“WarAndPeace.txt”);
array<ZosUnixObject^>^ files = dir.GetObjects(“X*”);
dir.Rename(“Garbage”);
dir.OtherAccess = ZosUnxAccess::Read | ZosUnixAccess::Write;
```

Visual Basic

```
Dim dir As ZosUnixDirectory
Dim file As ZosUnixObject = dir.GetObject(“War.txt”)
Dim files() As ZosUnixObject[] = dir.GetObjects(“X*”)
dir.Rename(“Garbage”)
dir.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write
```

Jscript

```
var dir: ZosUnixDirectory;
var file : ZosUnixObject = dir.GetObject(“WarAndPeace.txt”);
var files : ZosUnixObject[] = dir.GetObjects(“X*”);
dir.Rename(“Garbage”);
dir.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write
```

ZosUnixFile

The **ZosUnixFile** object represents a Unix file. The **ZosUnixFile** class is derived from **ZosUnixObject**.

ZosUnixFile Properties

ZosUnixFile exposes the following properties, most of which are inherited from **ZosUnixObject**.

Property	Type	R/W	Description
Name	String	R	Directory name *(Inherited from ZosUnixObject)*.
Path	String	R	Full path name of the directory (for example `\\server\Unix\u\judy\abc.txt`). *(Inherited from ZosUnixObject)*.
UnixPath	String	R	Unix file system path name (for example `/u/judy/abc.txt`). *(Inherited from ZosUnixObject)*.
FileType	ZosUnixFileType	R	Type of Unix file system object, which is always File for this type of object *(Inherited from ZosUnixObject)*.
FileSize	Int64	R	File size (bytes)
FileFormat	ZosUnixFileFormat	R	File format
CreationTime	DateTime	R	Creation time *(Inherited from ZosUnixObject)*.
LastWriteTime	DateTime	R	last update time *(Inherited from ZosUnixObject)*.
LastAccessTime	DateTime	R	Last access time *(Inherited from ZosUnixObject)*.
User	String	R	User owner *(Inherited from ZosUnixObject)*.
Group	String	R	Group owner *(Inherited from ZosUnixObject)*.
UserAccess	ZosUnixAccess	R/W	User access permissions *(Inherited from ZosUnixObject)*.
GroupAccess	ZosUnixAccess	R/W	Group access permissions *(Inherited from ZosUnixObject)*.
OtherAccess	ZosUnixAccess	R/W	Other access permissions *(Inherited from ZosUnixObject)*.

ZosUnixFile Methods

ZosUnixFile exposes the following methods, some of which are inherited from **ZosUnixObject**.

Function	Description

`void`
`    Refresh()`
	Refreshes the Unix file system information.
	(Inherited from ZosUnixObject).

`Boolean`
`    CheckAccess(`
`        ZosUnixAccessCheck flags`
`        )`
	Checks whether or not the user has the specified access permissions for the Unix file.
	(Inherited from ZosUnixObject).

`void`
`    Delete()`
	Deletes the file.
	(Inherited from ZosUnixObject).

`void`
`    Rename(`
`        String newName,`
`        Boolean replaceExisting`
`        )`
	Renames the file. You can optionally replace an existing file with the same name.
	(Inherited from ZosUnixObject).

`static`
`ZosUnixFile`
`    Create(`
`        ZosUnixDirectory parent,`
`        String name,`
`        ZosUnixAccess userAccess,`
`        ZosUnixAccess groupAccess,`
`        ZosUnixAccess otherAccess`
`        )`
	Creates a new empty file.

`void`
`    CopyTo(`
`        String path,`
`        Boolean replaceExisting`
`        )`
	Copies the file to another Unix file. You can optionally replace an existing file with the same name.

`void`
`    Export(`
`        String dsname,`
`        String member  [optional]`
`        )`
	Copies the file to a data set or PDS member.

`Void`
`    Import(`
`        String dsname,`
`        String member  [optional]`
`        )`
	Copies the file from a data set or PDS member.

ZosUnixFile Examples

Examples of using **ZosUnixFile** are shown below.

C#

```
ZosUnixFile file;
file.CopyTo("/u/MarthStewart/Recipes/GritsAndJowls.txt");
file.Import("MY.GARBAGE.DATA");
file.Rename("HumptyDumpty.txt", true);
file.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write;
```

C++

```
ZosUnixFile^ file;
file.CopyTo("/u/MarthStewart/Recipes/GritsAndJowls.txt");
file.Import("MY.GARBAGE.DATA");
file.Rename("HumptyDumpty.txt", true);
file.OtherAccess = ZosUnxAccess::Read | ZosUnixAccess::Write;
```

Visual Basic

```
Dim file As ZosUnixFile
file.CopyTo("/u/MarthStewart/Recipes/GritsAndJowls.txt")
file.Import("MY.GARBAGE.DATA")
file.Rename("HumptyDumpty.txt", true)
file.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write
```

Jscript

```
var file: ZosUnixFile;
file.CopyTo("/u/MarthStewart/Recipes/GritsAndJowls.txt");
file.Import("MY.GARBAGE.DATA");
file.Rename("HumptyDumpty.txt", true);
file.OtherAccess = ZosUnxAccess.Read | ZosUnixAccess.Write;
```

ZosUnixFolder

The **ZosUnixFolder** object represents a single user-defined Unix folder. A Unix folder is a local Windows alias for a Unix directory. This is conceptually similar to a Unix symbolic link, but the Unix folder is user-specific and is known only on the user's Windows machine. Unix folders can only be created at the root directory level, but they can refer to directories at any level.

Unix folder names must begin with "!" to distinguish them from Unix directories, files, or symbolic links. Therefore, Unix directories or symbolic links in the root directory cannot begin with "!". This naming restriction applies only to the root directory level.

This object can be obtained using the **UnixFolder** property of **ZosServer** or the Item property of **ZosUnixFolders**.

ZosUnixFolder Properties

ZosUnixFolder exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Name of the folder.
Path	String	R	Full path name of the folder.
TargetPath	String	R/W	Unix path name for the directory represented by this folder.
TargetDirectory	ZosUnixDirectory	R	Unix directory object for the directory represented by this object

ZosUnixFolder Methods

ZosUnixFolder exposes the following methods:

Function	Description

`void`
`    Delete()`
	Deletes the folder.

`void`
`    Rename(`
`        String newName`
`        )`
	Renames the folder.

ZosUnixFolders

The **ZosUnixFolders** object is a collection of all Unix folders for the same server. This object is obtained using the **UnixFolders** property of the **ZosServer** object.

ZosUnixFolders Properties

ZosUnixFolders exposes the following properties:

Property	Type	R/W	Description
[index] **[name]**	ZosUnixFolder	R	Folder with specified name or index.
Count	Int32	R	Number of objects in collection.
Path	String	R	File system path name for collection.

ZosUnixFolders Methods

ZosUnixFolders exposes the following methods:

Function	Description

`void`
`    Refresh()`
	Refreshes collection.

`Int32`
`    Add(`
`        String folderName,`
`        String targetPath`
`        )`
	Adds a new folder.
	Target path is Unix path name.
	Returns index at which object has been added.

`Int32`
`    Add(`
`        String folderName,`
`       ZosUnixDirectory targetDirectory`
`        )`
	Adds a new folder.
	Returns index at which object has been added.

`Boolean`
`    Remove(`
`        String folderName`
`        )`
	Deletes a folder. Returns true if folder was removed or false if folder is not found.

`Int32`
`    FindIndex(`
`        String name`
`        )`
	Searches for folder with specified name and returns zero-based index. Returns -1 if name is not found.

`ZosUnixFolder`
`    Find(`
`        String name`
`        )`
	Searches for folder with specified name and returns reference to object. Returns null if name is not found.

ZosUnixLink

The **ZosUnixLink** object represents a Unix symbolic link. The **ZosUnixLink** class is derived from **ZosUnixObject**. The symbolic link can be to either a directory or file.

ZosUnixLink Properties

ZosUnixLink exposes the following properties, most of which are inherited from **ZosUnixObject**.

Property	Type	R/W	Description
Name	String	R	Symbolic link name. *(Inherited from ZosUnixObject)*.

Path	String	R	Full path name of the symbolic link (for example `\\server\Unix\u\judy\symlink`). *(Inherited from ZosUnixObject)*.
UnixPath	String	R	Unix file system path name (for example `/u/judy/symlink`). *(Inherited from ZosUnixObject)*.
FileType	ZosUnixFileType	R	Type of Unix file system object, which is always SymLink for this type of object. *(Inherited from ZosUnixObject)*.
TargetPath	String	R	Path name for target of link.
TargetObject	ZosUnixObject	R	The ZosUnixDirectory or ZosUnixFile object that represents the target of the link.
CreationTime	DateTime	R	Creation time. *(Inherited from ZosUnixObject)*.
LastWriteTime	DateTime	R	Last update time. *(Inherited from ZosUnixObject)*.
LastAccessTime	DateTime	R	Last access time. *(Inherited from ZosUnixObject)*.
User	String	R	User owner. *(Inherited from ZosUnixObject)*.
Group	String	R	Group owner. *(Inherited from ZosUnixObject)*.
UserAccess	ZosUnixAccess	R/W	User access permissions. (*(Inherited from ZosUnixObject)*.
GroupAccess	ZosUnixAccess	R/W	Group access permissions. *(Inherited from ZosUnixObject)*.
OtherAccess	ZosUnixAccess	R/W	Other access permissions. *(Inherited from ZosUnixObject)*.

ZosUnixLink Methods

ZosUnixLink exposes the following methods, most of which are inherited from **ZosUnixObject**.

Function	**Description**

`void`
`    Refresh()`
	Refreshes the Unix file system information. *(Inherited from ZosUnixObject)*.

`Boolean`
`    CheckAccess(`
`        ZosUnixAccessCheck flags`
`        )`
	Checks whether or not the user has the specified access permissions for the Unix file. *(Inherited from ZosUnixObject)*.

`void Delete()`	Deletes the symbolic link. *(Inherited from ZosUnixObject)*.
`void Rename( String newName, Boolean replaceExisting )`	Renames the file. You can optionally replace an existing file with the same name. *(Inherited from ZosUnixObject)*.
`static ZosUnixLink Create( ZosUnixDirectory parent, String name, String targetPath )`	Creates a new symbolic link.

ZosUnixLink Examples

Examples of using **ZosUnixLink** are shown below.

C#
```
ZosUnixLink link;
ZosUnixDir parent;
ZosUnixObject target = link.TargetObject;
link = ZosUnixLink.Create(parent, “MyStuff”, “/u/Judy/Stuff”);
```

C++
```
ZosUnixLink^ link;
ZosUnixDir^ parent;
ZosUnixObject target = link.TargetObject;
link = ZosUnixLink::Create(parent, “MyStuff”, “/u/Judy/Stuff”);
```

Visual Basic
```
Dim link As ZosUnixLink
Dim parent As ZosUnixDir
Dim target As ZosUnixObject = link.TargetObject;
link = ZosUnixLink.Create(parent, “MyStuff”, “/u/Judy/Stuff”)
```

Jscript
```
var link : ZosUnixLink;
var parent : ZosUnixDir;
var target : ZosUnixObject = link.TargetObject;
link = ZosUnixLink.Create(parent, “MyStuff”, “/u/Judy/Stuff”);
```

ZosUnixObject

The **ZosUnixObject** object represents a Unix file system object, which can be a Unix directory, a Unix file, or a Unix symbolic link. **ZosUnixObject** is the base class for the **ZosUnixDirectory**, **ZosUnixFile**, and **ZosUnixLink** classes. The **FileType** property indicates whether the **ZosUnixObject** is actually a **ZosUnixDirectory**, a **ZosUnixFile**, or a **ZosUnixLink** object.

This object can be obtained using any of the following:

- **GetUnixObject** method of **ZosServer**
- **UnixRootDirectory** property of **ZosServer**
- **TargetDirectory** property of **ZosUnixFolder**
- **TargetObject** of **ZosUnixLink**
- **GetObject** method of **ZosDirectory**
- **GetObjects** member of **ZosDirectory**

ZosUnixObject Properties

ZosUnixObject exposes the following properties:

Property	Type	R/W	Description
Name	String	R	Directory, file, or symbolic link name.
Path	String	R	Full path name of the Unix file system object (for example `\\server\Unix\u\judy`).
UnixPath	String	R	Unix file system path name (for example `/u/judy`).
FileType	ZosUnixFileType	R	Type of Unix file system object (directory, file, or symbolic link).
CreationTime	DateTime	R	Creation time.
LastWriteTime	DateTime	R	Last update time.
LastAccessTime	DateTime	R	Last access time.
User	String	R	User owner.
Group	String	R	Group owner.
UserAccess	ZosUnixAccess	R/W	User access permissions.
GroupAccess	ZosUnixAccess	R/W	Group access permissions.
OtherAccess	ZosUnixAccess	R/W	Other access permissions.

ZosUnixObject Methods

ZosUnixObject exposes the following methods:

Function	Description
`void Refresh()`	Refreshes the Unix file system information.

`Boolean`
    `CheckAccess(`
        `ZosUnixAccessCheck flags`
        `)`
	Checks whether or not the user has the specified access permissions for the Unix directory or file.

`void`
    `Delete()`
	Deletes the file, directory, or link. For directories, contents are also deleted.

`void`
    `Rename(`
        `String newName,`
        `Boolean replaceExisting` [optional]
        `)`
	Renames the file. You can optionally replace an existing file with the same name. This option is not valid for directories.

Chapter 4

Examples

Several sample scripts are provided with ChangeMan ZDD that illustrate how to use the programming interface to perform some common ChangeMan ZDD operations. You can find these samples in the Samples\.NET folder, in the directory where ChangeMan ZDD is installed. There are samples for C#, Visual Basic, and JScript.

Logging on to a Server

You can use the ***Logon*** method (function) from your program or script to log on to a z/OS server. An example of when you would use ***Logon*** is when your program or script is accessing a data set on a z/OS server.

The following scripts illustrate how to log on to a z/OS server.

C# Example

```
/**********************************************************************
* File Name:   Logon.cs
*
* Description: Logon to server.  If userid and password not specified,
*              user will be prompted.
*
* Usage:       Logon <server> [<userid>] [<password>] [<newpassword>]
*
* Copyright ©2007, Serena Software. Licensed material. All rights reserved.
**********************************************************************/

using System;
using System.Collections.Generic;
using System.Text;
using ZosApi;

namespace Logon
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                //////////////////////////////
                // Get command line arguments
                //////////////////////////////

                if (args.Length < 1)
                {
                    Console.WriteLine("Usage: Logon <server> [<userid>]
                      [<password>] [<newpassword>]");
                    Environment.Exit(0);
                }

                String serverName;
                String userID;
                String password;
                String newPassword;

                serverName = args[0];

                if (args.Length > 1)
                {
                    userID = args[1];
                }
                else
                {
                    userID = "";
                }
                if (args.Length > 2)
                {
                    password = args[2];
                }
                else
                {
                    password = "";
                }

                if (args.Length > 3)
                {
                    newPassword = args[3];
                }
                else
                {
                    newPassword = "";
                }

                ///////////////////
                // Logon to server
                ///////////////////

                ZosNetwork network = new ZosNetwork();
                ZosServer server  = network.Servers[serverName];

                if (server == null)
                {
                    Console.WriteLine("Server {0} not found", serverName);
                    Environment.Exit(1);
                }

                server.Logon(userID, password, newPassword);

               Console.WriteLine("User {0} logged onto {1}", userID, serverName);
            }

            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine(e.TargetSite);
            }
        }
    }
}
```

Visual Basic Example

```
'***********************************************************************
' File Name:   Logon.vb
'
' Description: Logon to server.  If userid and password not specified,
'              user will be prompted.
'
' Usage:       Logon <server> [<userid>] [<password>] [<newpassword>]
'
' Copyright ©2007, Serena Software. Licensed material. All rights reserved.
'***********************************************************************

Imports System
Imports ZosApi
```

```
Module Logon

    Sub Main()

        Try

            '----------------------------
            ' Get command line arguments
            '----------------------------

            Dim args As String() = Environment.GetCommandLineArgs()

            If args.Length < 2

                Console.WriteLine("Usage: Logon <server> (<userid>) (<password>) _
                 (<newpassword>)")
                Environment.Exit(0)

            End If

            Dim serverName As String
            Dim userID As String
            Dim password As String
            Dim newPassword As String

            serverName = args(1)

            If args.Length > 2
                userID = args(2)
            Else
                userID = ""
            End If

            If args.Length > 3
                password = args(3)
            Else
                password = ""
            End If

            If args.Length > 4
                newPassword = args(4)
            Else
                newPassword = ""
            End If

            '-----------------
            ' Logon to server
            '-----------------

            Dim network As ZosNetwork = new ZosNetwork()
            Dim server As ZosServer  = network.Servers(serverName)

            If server Is Nothing Then

                Console.WriteLine("Server {0} not found", serverName)
                Environment.Exit(1)

            End If

            server.Logon(userID, password, newPassword)

            Console.WriteLine("User {0} logged onto {1}", userID, serverName)
        Catch e As Exception

            Console.WriteLine(e.Message)
            Console.WriteLine(e.TargetSite)

        End Try

    End Sub

End Module
```

JScript Example

```
/**********************************************************************
* File Name:   Logon.js
*
* Description: Logon to server.  If userid and password not specified,
*              user will be prompted.
*
* Usage:       Logon <server> [<userid>] [<password>] [<newpassword>]
*
* Copyright ©2007, Serena Software. Licensed material. All rights reserved.
**********************************************************************/

import System;
import ZosApi;

try
{
    //////////////////////////////
    // Get command line arguments
    //////////////////////////////

    var args : String[] = Environment.GetCommandLineArgs();

    if (args.Length < 2)
    {
        Console.WriteLine("Usage: Logon <server> [<userid>] [<password>]
          [<newpassword>]");
        Environment.Exit(0);
    }

    var serverName  : String;
    var userID      : String;
    var password    : String;
    var newPassword : String;

    serverName = args[1];

    if (args.Length > 2)
    {
        userID = args[2];
    }
    else
    {
        userID = "";
    }

    if (args.Length > 3)
    {
        password = args[3];
    }
```

```
        else
        {
            password = "";
        }

        if (args.Length > 4)
        {
            newPassword = args[4];
        }
        else
        {
            newPassword = "";
        }

        ///////////////////
        // Logon to server
        ///////////////////

        var network : ZosNetwork = new ZosNetwork();
        var server  : ZosServer  = network.Servers[serverName];

        if (server == null)
        {
            Console.WriteLine("Server {0} not found", serverName);
            Environment.Exit(1);
        }

        server.Logon(userID, password, newPassword);

        Console.WriteLine("User {0} logged onto {1}", userID, serverName);
    }

    catch (e : Exception)
    {
        Console.WriteLine(e.Message);
        Console.WriteLine(e.TargetSite);
    }
```

Submitting JCL to a Server

You can use the ***SubmitJCL*** method (function) from your program or script to submit JCL to a z/OS server. A situation where you might use ***SubmitJCL*** is when a program or script, that runs from Windows Task Scheduler, needs to submit a nightly batch job to a z/OS server.

The following scripts illustrate how to submit JCL to a z/OS server.

C# Example

```
/**********************************************************************
* File Name:   SubmitJcl.cs
*
* Description: Submit job to server.
*
* Usage:       SubmitJcl <server> <file.name>
*
* Copyright ©2007, Serena Software. Licensed material. All rights reserved.
**********************************************************************/
```

```
using System;
using System.Collections.Generic;
using System.Text;
using ZosApi;

namespace SubmitJcl
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                ///////////////////////////////
                // Get command line arguments
                ///////////////////////////////

                if (args.Length < 2)
                {
                    Console.WriteLine("Usage: SubmitJcl <server> <file.name>");
                    Environment.Exit(1);
                }

                String serverName = args[0];
                String fileName = args[1];

                //////////////
                // Submit JCL
                //////////////

                ZosNetwork network = new ZosNetwork();
                ZosServer server  = network.Servers[serverName];

                ZosJesJob[] jobs = server.SubmitJcl(fileName);

                foreach (ZosJesJob job in jobs)
                {
                    Console.WriteLine("Job submitted: JobName={0} JobID={1}",
                                      job.JobName,
                                      job.JobID);
                }
            }

            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine(e.TargetSite);
            }
        }
    }
}
```

Visual Basic Example

```
'***********************************************************************
' File Name:   SubmitJcl.vb
'
' Description: Submit job to server.
'
' Usage:       SubmitJcl <server> <file.name>
'
' Copyright ©2007, Serena Software. Licensed material. All rights reserved.
'***********************************************************************

Imports System
Imports ZosApi

Module SubmitJcl

    Sub Main()

        Try

            '----------------------------
            ' Get command line arguments
            '----------------------------

            Dim args As String() = Environment.GetCommandLineArgs()

            If args.Length < 3

                Console.WriteLine("Usage: SubmitJcl <server> <file.name>")
                Environment.Exit(1)

            End If

            Dim serverName As String = args(1)
            Dim fileName As String = args(2)

            '------------
            ' Submit JCL
            '------------

            Dim network As ZosNetwork = new ZosNetwork()
            Dim server As ZosServer  = network.Servers(serverName)

            Dim suppressMessage As Boolean = false

            Dim jobs() as ZosJesJob = server.SubmitJcl(fileName)

            Dim job As ZosJesJob

            For Each job In jobs

                Console.WriteLine("Job submitted: JobName={0} JobID={1}", _
                                  job.JobName, _
                                  job.JobID)
            Next

        Catch e As Exception

            Console.WriteLine(e.Message)
            Console.WriteLine(e.TargetSite)

        End Try
    End Sub

End Module
```

JScript Example

```
/*********************************************************************
* File Name:   SubmitJcl.js
*
* Description: Submit job to server.
*
* Usage:       SubmitJcl <server> <file.name>
*
* Copyright ©2007, Serena Software. Licensed material. All rights reserved.
*********************************************************************/

import System;
import ZosApi;

try
{
    ///////////////////////////////
    // Get command line arguments
    ///////////////////////////////

    var args : String[] = Environment.GetCommandLineArgs();

    if (args.Length < 3)
    {
        Console.WriteLine("Usage: SubmitJcl <server> <file.name>");
        Environment.Exit(1);
    }

    var serverName : String = args[1];
    var fileName   : String = args[2];

    //////////////
    // Submit JCL
    //////////////

    var network : ZosNetwork = new ZosNetwork();
    var server  : ZosServer  = network.Servers[serverName];

    var jobs: ZosJesJob[] = server.SubmitJcl(fileName);

    for (var job in jobs)
    {
        Console.WriteLine("Job submitted: JobName={0} JobID={1}",
                          job.JobName,
                          job.JobID);
    }
}

catch (e : Exception)
{
    Console.WriteLine(e.Message);
    Console.WriteLine(e.TargetSite);
}
```

Configuring ChangeMan ZDD for a New User

To simplify the setup of ChangeMan ZDD for multiple desktops, you can write a script to automate many of the configuration tasks. Then, a new user can configure ChangeMan ZDD for their own desktop simply by executing the script.

The following scripts illustrate how the configuration tasks can be performed.

C# Example

```
/**********************************************************************
* File Name:   NewConfig.cs
*
* Description: Sample for creating a new configuration.
*
* Usage:       NewConfig <userid>
*
* Copyright ©2003-2011, Serena Software. Licensed material. All rights reserved.
**********************************************************************/

using System;
using System.Collections.Generic;
using System.Text;
using ZosApi;

namespace NewConfig
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                ///////////////////////////////
                // Get command line arguments
                ///////////////////////////////

                if (args.Length < 1)
                {
                    Console.WriteLine("Usage: NewConfig <userid>");
                    Environment.Exit(1);
                }

                String userID = args[0];

                //////////////////////////////
                // Update network properties
                //////////////////////////////

                ZosNetwork network = new ZosNetwork();

                network.CacheFolder = "C:\\Temp";
                network.CacheDays   = 3;
                network.NotifyPort  = 8000;
                network.NotifyJobStep = true;
                network.NotifyMessageBox = true;

                ///////////////////////
                // Add the new servers
                ///////////////////////
```

```
           ZosServers servers = network.Servers;

           servers.Add("Server1", "172.20.20.1", 5000, 1140, "Description1");
           servers.Add("Server2", "172.20.20.2", 5000, 1140, "Description2");
           servers.Add("Server3", "172.20.20.3", 5000, 1140, "Description3");

           ////////////////////////////////////////
           // Update the properties for each server
           ////////////////////////////////////////

           foreach (ZosServer server in servers)
           {
               int index;

               ////////////////////////////////////////
               // Add the data set file format entries
               ////////////////////////////////////////

               ZosFileFormatMappings dsFileFormats =
server.DataSetFileFormatMappings;

              dsFileFormats.Add(-1, "**.ASCII.TEXT",   ZosFileFormat.AsciiText);
              dsFileFormats.Add(-1, "**.ASCII.DATA",   ZosFileFormat.AsciiData);
               dsFileFormats.Add(-1, "**.UNICODE.TEXT",
ZosFileFormat.UnicodeText);
               dsFileFormats.Add(-1, "**.EBCDIC.TEXT",
ZosFileFormat.EbcdicText);
               dsFileFormats.Add(-1, "**.EBCDIC.DATA",
ZosFileFormat.EbcdicData);
               dsFileFormats.Add(-1, "**.BINARY",
ZosFileFormat.BinaryCRLF);

               ///////////////////////////////////////////////////////////////
               // The following illustrates a faster way to do the same thing
               ///////////////////////////////////////////////////////////////

               ZosFileFormatMapping[] fileFormatArray = new
ZosFileFormatMapping[]
               {
                   new ZosFileFormatMapping("**.ASCII.TEXT",
ZosFileFormat.AsciiText),
                   new ZosFileFormatMapping("**.ASCII.DATA",
ZosFileFormat.AsciiData),
                   new ZosFileFormatMapping("**.UNICODE.TEXT",
ZosFileFormat.UnicodeText),
                   new ZosFileFormatMapping("**.EBCDIC.TEXT",
ZosFileFormat.EbcdicText),
                   new ZosFileFormatMapping("**.EBCDIC.DATA",
ZosFileFormat.EbcdicData),
                   new ZosFileFormatMapping("**.BINARY",
ZosFileFormat.BinaryCRLF)
               };

               dsFileFormats.FromArray(fileFormatArray);

               ////////////////////////////////////
               // Add the Unix file format entries
               ////////////////////////////////////

               ZosFileFormatMappings uFileFormats =
server.UnixFileFormatMappings;

               uFileFormats.Add(-1, "*.TEXT",  ZosFileFormat.AsciiText);
               uFileFormats.Add(-1, "*.UTEXT", ZosFileFormat.UnicodeText);
               uFileFormats.Add(-1, "*.BIN",   ZosFileFormat.Binary);

               ////////////////////////////////
               // Add the library type entries
               ////////////////////////////////

               ZosLibTypeMappings libTypes = server.LibTypeMappings;

               libTypes.Add(-1, "**.LIBRARY",  ZosLibType.Lib);
               libTypes.Add(-1, "**.PANVALET", ZosLibType.Pan);

               ///////////////////////////////////////////////////////////////
               // The following illustrates a faster way to do the same thing
               ///////////////////////////////////////////////////////////////

               ZosLibTypeMapping[] libTypeArray = new ZosLibTypeMapping[]
               {
                   new ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib),
                   new ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)
               };

               libTypes.FromArray(libTypeArray);

               //////////////////////////////////
               // Add the file extension entries
               //////////////////////////////////

               ZosFileExtensionMappings fileExtensions =
server.FileExtensionMappings;

               fileExtensions.Add(-1, "**.CNTL",  "jcl");
               fileExtensions.Add(-1, "**.COBOL", "cbl");
               fileExtensions.Add(-1, "**.LIST",  "txt");
               fileExtensions.Add(-1, "**.WORD",  "doc");
               fileExtensions.Add(-1, "**.EXCEL", "xls");

               ///////////////////////////////////////////////////////////////
               // The following illustrates a faster way to do the same thing
               ///////////////////////////////////////////////////////////////

               ZosFileExtensionMapping[] fileExtArray = new
ZosFileExtensionMapping[]
               {
                   new ZosFileExtensionMapping("**.CNTL",  "jcl"),
                   new ZosFileExtensionMapping("**.COBOL", "cbl"),
                   new ZosFileExtensionMapping("**.LIST",  "txt"),
                   new ZosFileExtensionMapping("**.WORD",  "doc"),
                   new ZosFileExtensionMapping("**.EXCEL", "xls")
               };

               fileExtensions.FromArray(fileExtArray);

               //////////////////////////////////////
               // Add the profiles for new data sets
               //////////////////////////////////////

               ZosDataSetProfiles dsProfiles = server.DataSetProfiles;

               dsProfiles.Add(-1, "**.DATA", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS1", "STORCLS1", "MGMTCLS1",
ZosSpaceUnit.Trk,   2,  1, 5, "SYSDA", "VOL001");
```

```
               dsProfiles.Add(-1, "**.TEMP", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS2", "STORCLS2", "MGMTCLS2",
ZosSpaceUnit.Cyl,   2,  1, 5, "SYSDA", "VOL002");
               dsProfiles.Add(-1, "**.LIST", ZosDataSetType.Seq,
ZosRecordFormat.VB, 80, 0, "",        "",          "",       ZosSpaceUnit.Blk,
500, 50, 5, "SYSDA", "");

               ///////////////////////////////////////////////////////////////
               // The following illustrates a faster way to do the same thing
               ///////////////////////////////////////////////////////////////

               ZosDataSetProfile[] dsProfileArray = new ZosDataSetProfile[]
               {
                   new ZosDataSetProfile("**.DATA", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS1", "STORCLS1", "MGMTCLS1",
ZosSpaceUnit.Trk,   2,  1, 5, "SYSDA", "VOL001"),
                   new ZosDataSetProfile("**.TEMP", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS2", "STORCLS2", "MGMTCLS2",
ZosSpaceUnit.Cyl,   2,  1, 5, "SYSDA", "VOL002"),
                   new ZosDataSetProfile("**.LIST", ZosDataSetType.Seq,
ZosRecordFormat.VB, 80, 0, "",        "",          "",       ZosSpaceUnit.Blk,
500, 50, 5, "SYSDA", "")
               };

               dsProfiles.FromArray(dsProfileArray);

               ////////////////////////
               // Add data set folders
               ////////////////////////

               ZosDataSetFolders dsfolders = server.DataSetFolders;

               ZosDataSetFolder dsfolder;
               ZosNameFilters filters;
               ZosPrefixMappings prefixes;

               //////////////////////////////////////////////////
               // "My DataSets" folder for all user's data sets
               //////////////////////////////////////////////////

               index = dsfolders.Add("My DataSets");
               dsfolder = dsfolders[index];

               filters  = dsfolder.Filters;
               prefixes = dsfolder.PrefixMappings;

               filters.Add(userID + ".**");

               prefixes.Add(-1, "**", userID);

               ///////////////////////////////////////////////////
               // "My Source" folder for user's source libraries
               ///////////////////////////////////////////////////

               index = dsfolders.Add("Source");
               dsfolder = dsfolders[index];

               filters  = dsfolder.Filters;
               prefixes = dsfolder.PrefixMappings;

               filters.Add(userID + ".**.COBOL");
               filters.Add(userID + ".**.ASM");

               prefixes.Add(-1, "**.MOUSE", "MICKEY");
               prefixes.Add(-1, "**", userID);

               ///////////////////////////////////////////////////////////////
               // The following illustrates a faster way to do the same thing
               ///////////////////////////////////////////////////////////////

               String[] filterArray = new String[]
               {
                   userID + ".**.COBOL",
                   userID + ".**.ASM"
               };

               filters.FromArray(filterArray);

               ZosPrefixMapping[] prefixArray = new ZosPrefixMapping[]
               {
                   new ZosPrefixMapping("**.MOUSE", "MICKEY"),
                   new ZosPrefixMapping("**", userID)
               };

               prefixes.FromArray(prefixArray);

               ///////////////////
               // Add job folders
               ///////////////////

               ZosJobFolders jobfolders = server.JobFolders;

               ZosJobFolder jobfolder;

               //////////////////////////////////////////
               // "My Jobs" folder for jobs owned by user
               //////////////////////////////////////////

               index = jobfolders.Add("My Jobs", ZosJobQueryType.QueueOwner,
userID);
               jobfolder = jobfolders[index];

               ////////////////////////////////////////////////////////
               // "ChangeMan" folder for job names prefixed with "CMN"
               ////////////////////////////////////////////////////////

             index = jobfolders.Add("ChangeMan", ZosJobQueryType.QueueJobname,
"CMN*");
               jobfolder = jobfolders[index];

               //////////////////////////////////////
               // "Active" folder for all active jobs
               //////////////////////////////////////

               index = jobfolders.Add("Active", ZosJobQueryType.ActiveAll);
               jobfolder = jobfolders[index];

               /////////////////////////
               // Add ChangeMan instances
               /////////////////////////

               ZosChangeManInstances instances = server.ChangeManInstances;

               instances.Add("ChangeMan-Prod", 3000, "Production ChangeMan");
               instances.Add("ChangeMan-Test", 3001, "Test ChangeMan");

               foreach (ZosChangeManInstance instance in instances)
               {
```

```
                        ////////////////////////////////////////
                        // Add the ChangeMan file format entries
                        ////////////////////////////////////////

                        ZosFileFormatMappings fileFormats =
     instance.FileFormatMappings;

                        fileFormats.Add(-1, "SRC",  ZosFileFormat.AsciiText);
                        fileFormats.Add(-1, "DOC",  ZosFileFormat.UnicodeText);
                        fileFormats.Add(-1, "BIN",  ZosFileFormat.Binary);
                    }
                }
            }

            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine(e.TargetSite);
            }
        }
    }
}
```

Visual Basic Example

```
'**********************************************************************
' File Name:   NewConfig.vb
'
' Description: Sample for creating a new configuration.
'
' Usage:       NewConfig <userid>
'
' Copyright ©2003-2011, Serena Software. Licensed material. All rights reserved.
'**********************************************************************

Imports System
Imports ZosApi

Module NewConfig

    Sub Main()

        Try

            '----------------------------
            ' Get command line arguments
            '----------------------------

            Dim args As String() = Environment.GetCommandLineArgs()

            If args.Length < 2

                Console.WriteLine("Usage: NewConfig <userid>")
                Environment.Exit(1)

            End If

            Dim userID As String = args(1)

            '----------------------------
            ' Update network properties
```

```
          '----------------------------

          Dim network As ZosNetwork = new ZosNetwork()

          network.CacheFolder = "C:\Temp"
          network.CacheDays   = 3
          network.NotifyPort  = 8000
          network.NotifyJobStep = True
          network.NotifyMessageBox = True

          '---------------------
          ' Add the new servers
          '---------------------

          Dim servers As ZosServers = network.Servers

          servers.Add("Server1", "172.20.20.1", 5000, 1140, "Description1")
          servers.Add("Server2", "172.20.20.2", 5000, 1140, "Description2")
          servers.Add("Server3", "172.20.20.3", 5000, 1140, "Description3")

          '--------------------------------------
          ' Update the properties for each server
          '--------------------------------------

          Dim server As ZosServer

          For Each server In servers

              Dim index As Integer

              '------------------------------------
              ' Add the data set file format entries
              '------------------------------------

              Dim dsFileFormats As ZosFileFormatMappings =
server.DataSetFileFormatMappings

              dsFileFormats.Add(-1, "**.ASCII.TEXT",   ZosFileFormat.AsciiText)
              dsFileFormats.Add(-1, "**.ASCII.DATA",   ZosFileFormat.AsciiData)
              dsFileFormats.Add(-1, "**.UNICODE.TEXT", ZosFileFormat.UnicodeText)
              dsFileFormats.Add(-1, "**.EBCDIC.TEXT",  ZosFileFormat.EbcdicText)
              dsFileFormats.Add(-1, "**.EBCDIC.DATA",  ZosFileFormat.EbcdicData)
              dsFileFormats.Add(-1, "**.BINARY",       ZosFileFormat.BinaryCRLF)

              '------------------------------------------------------------
              ' The following illustrates a faster way to do the same thing
              '------------------------------------------------------------

              Dim fileFormatArray() As ZosFileFormatMapping = _
              { _
                  New ZosFileFormatMapping("**.ASCII.TEXT",
ZosFileFormat.AsciiText), _
                  New ZosFileFormatMapping("**.ASCII.DATA",
ZosFileFormat.AsciiData), _
                  New ZosFileFormatMapping("**.UNICODE.TEXT",
ZosFileFormat.UnicodeText), _
                  New ZosFileFormatMapping("**.EBCDIC.TEXT",
ZosFileFormat.EbcdicText), _
                  New ZosFileFormatMapping("**.EBCDIC.DATA",
ZosFileFormat.EbcdicData), _
                  New ZosFileFormatMapping("**.BINARY", ZosFileFormat.Binary) _
              }

              dsFileFormats.FromArray(fileFormatArray)
```

```
           '----------------------------------
           ' Add the Unix file format entries
           '----------------------------------

           Dim uFileFormats As ZosFileFormatMappings =
server.UnixFileFormatMappings

           uFileFormats.Add(-1, "*.TEXT",  ZosFileFormat.AsciiText)
           uFileFormats.Add(-1, "*.UTEXT", ZosFileFormat.UnicodeText)
           uFileFormats.Add(-1, "*.BIN",   ZosFileFormat.Binary)

           '--------------------------------
           ' Add the library type entries
           '--------------------------------

           Dim libTypes As ZosLibTypeMappings = server.LibTypeMappings

           libTypes.Add(-1, "**.LIBRARY", ZosLibType.Lib)
           libTypes.Add(-1, "**.PANVALET", ZosLibType.Pan)

           '--------------------------------------------------------------
           ' The following illustrates a faster way to do the same thing
           '--------------------------------------------------------------

           Dim libTypeArray() As ZosLibTypeMapping =                 _
           {                                                         _
               New ZosLibTypeMapping("**.LIBRARY",  ZosLibType.Lib), _
               New ZosLibTypeMapping("**.PANVALET", ZosLibType.Pan)  _
           }

           libTypes.FromArray(libTypeArray)

           '----------------------------------
           ' Add the file extension entries
           '----------------------------------

           Dim fileExtensions As ZosFileExtensionMappings =
server.FileExtensionMappings

           fileExtensions.Add(-1, "**.CNTL",  "jcl")
           fileExtensions.Add(-1, "**.COBOL", "cbl")
           fileExtensions.Add(-1, "**.LIST",  "txt")
           fileExtensions.Add(-1, "**.WORD",  "doc")
           fileExtensions.Add(-1, "**.EXCEL", "xls")

           '--------------------------------------------------------------
           ' The following illustrates a faster way to do the same thing
           '--------------------------------------------------------------

           Dim fileExtArray() As ZosFileExtensionMapping =     _
           {                                             _
               New ZosFileExtensionMapping("**.CNTL", "jcl"),  _
               New ZosFileExtensionMapping("**.COBOL", "cbl"), _
               New ZosFileExtensionMapping("**.LIST", "txt"),  _
               New ZosFileExtensionMapping("**.WORD", "doc"),  _
               New ZosFileExtensionMapping("**.EXCEL", "xls")  _
           }

           fileExtensions.FromArray(fileExtArray)

           '--------------------------------------
           ' Add the profiles for new data sets
           '--------------------------------------
           Dim dsProfiles As ZosDataSetProfiles = server.DataSetProfiles

         dsProfiles.Add(-1, "**.DATA", ZosDataSetType.Seq, ZosRecordFormat.FB,
80, 0, "DATACLS1", "STORCLS1", "MGMTCLS1", ZosSpaceUnit.Trk,   2,  1, 5,
"SYSDA", "VOL001")
         dsProfiles.Add(-1, "**.TEMP", ZosDataSetType.Seq, ZosRecordFormat.FB,
80, 0, "DATACLS2", "STORCLS2", "MGMTCLS2", ZosSpaceUnit.Cyl,   2,  1, 5,
"SYSDA", "VOL002")
         dsProfiles.Add(-1, "**.LIST", ZosDataSetType.Seq, ZosRecordFormat.VB,
80, 0, "",        "",         "",       ZosSpaceUnit.Blk, 500, 50, 5, "SYSDA",
"")

           '-------------------------------------------------------------
           ' The following illustrates a faster way to do the same thing
           '-------------------------------------------------------------

Dim dsProfileArray() As ZosDataSetProfile =
_
  {
_
               New ZosDataSetProfile("**.DATA", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS1", "STORCLS1", "MGMTCLS1",
ZosSpaceUnit.Trk,   2,  1, 5, "SYSDA", "VOL001"), _
               New ZosDataSetProfile("**.TEMP", ZosDataSetType.Seq,
ZosRecordFormat.FB, 80, 0, "DATACLS2", "STORCLS2", "MGMTCLS2",
ZosSpaceUnit.Cyl,   2,  1, 5, "SYSDA", "VOL002"), _
               New ZosDataSetProfile("**.LIST", ZosDataSetType.Seq,
ZosRecordFormat.VB, 80, 0, "",        "",         "",       ZosSpaceUnit.Blk,
500, 50, 5, "SYSDA", "")        _
           }

           dsProfiles.FromArray(dsProfileArray)

           '-----------------------
           ' Add data set folders
           '-----------------------

           Dim dsfolders As ZosDataSetFolders = server.DataSetFolders

           Dim dsfolder As ZosDataSetFolder
           Dim filters  As ZosNameFilters
           Dim prefixes As ZosPrefixMappings

           '-----------------------------------------------
           ' "My DataSets" folder for all user's data sets
           '-----------------------------------------------

           index = dsfolders.Add("My DataSets")
           dsfolder = dsfolders(index)

           filters  = dsfolder.Filters
           prefixes = dsfolder.PrefixMappings

           filters.Add(userID + ".**")

           prefixes.Add(-1, "**", userID)

           '------------------------------------------------
           ' "My Source" folder for user's source libraries
           '------------------------------------------------

           index = dsfolders.Add("Source")
           dsfolder = dsfolders(index)
```

```
filters  = dsfolder.Filters
prefixes = dsfolder.PrefixMappings

filters.Add(userID + ".**.COBOL")
filters.Add(userID + ".**.ASM")

prefixes.Add(-1, "**.MOUSE", "MICKEY")
prefixes.Add(-1, "**", userID)

'--------------------------------------------------------------
' The following illustrates a faster way to do the same thing
'--------------------------------------------------------------

Dim filterArray() As String = _
{                             _
    userID + ".**.COBOL",     _
    userID + ".**.ASM"        _
}

filters.FromArray(filterArray)

Dim prefixArray() As ZosPrefixMapping =         _
{                                               _
    New ZosPrefixMapping("**.MOUSE", "MICKEY"), _
    New ZosPrefixMapping("**", userID)          _
}

prefixes.FromArray(prefixArray)

'------------------
' Add job folders
'------------------

Dim jobfolders As ZosJobFolders = server.JobFolders

Dim jobfolder  As ZosJobFolder

'------------------------------------------
' "My Jobs" folder for jobs owned by user
'------------------------------------------

index = jobfolders.Add("My Jobs", ZosJobQueryType.QueueOwner, userID)
jobfolder = jobfolders(index)

'-------------------------------------------------------
' "ChangeMan" folder for job names prefixed with "CMN"
'-------------------------------------------------------

index = jobfolders.Add("ChangeMan", ZosJobQueryType.QueueJobname,
"CMN*")
jobfolder = jobfolders(index)

'--------------------------------------
' "Active" folder for all active jobs
'--------------------------------------

index = jobfolders.Add("Active", ZosJobQueryType.ActiveAll)
jobfolder = jobfolders(index)

'------------------------
' Add ChangeMan instances
'------------------------
```

```
                Dim instances As ZosChangeManInstances = server.ChangeManInstances

                instances.Add("ChangeMan-Prod", 3000, "Production ChangeMan")
                instances.Add("ChangeMan-Test", 3001, "Test ChangeMan")

                Dim instance As ZosChangeManInstance

                For Each instance In instances

                    '---------------------------------------
                    ' Add the ChangeMan file format entries
                    '---------------------------------------

                    Dim fileFormats As ZosFileFormatMappings =
     instance.FileFormatMappings

                    fileFormats.Add(-1, "SRC", ZosFileFormat.AsciiText)
                    fileFormats.Add(-1, "DOC", ZosFileFormat.UnicodeText)
                    fileFormats.Add(-1, "BIN", ZosFileFormat.BinaryCRLF)

                Next

            Next

        Catch e As Exception

            Console.WriteLine(e.Message)
            Console.WriteLine(e.StackTrace)

        End Try

    End Sub

End Module
```

JScript Example

```
/***********************************************************************
* File Name:   NewConfig.js
*
* Description: Sample for creating a new configuration.
*
* Usage:       NewConfig.js <userid>
*
* Copyright ©2003-2011, Serena Software. Licensed material. All rights reserved.
***********************************************************************/

var userID;

var network;
var servers;
var server;
var fileFormats;
var libTypes;
var fileExtensions;
var dsProfiles;
var folders;
var folder;
var subfolders;
var subfolder;
var filters;
var prefixes;

var enumerator;

//////////////////////////////
// Get command line arguments
//////////////////////////////

if (WScript.Arguments.Count() < 1)
{
    WScript.Echo("Usage: NewConfig.js <userid>");
    WScript.Quit(1);
}

userID = WScript.Arguments(0);

/////////////////////////////
// Update network properties
/////////////////////////////

network = new ActiveXObject("ZosShell.ZosNetwork");

network.CacheFolder = "C:\\Temp";
network.CacheDays   = 3;
network.NotifyPort  = 8000;
network.NotifyJobStep = true;
network.NotifyMessageBox = true;

///////////////////////
// Add the new servers
///////////////////////

servers = network.Servers;

servers.Add("Server1", "172.20.20.1", 5000, 1140, "Description1");
servers.Add("Server2", "172.20.20.2", 5000, 1140, "Description2");
servers.Add("Server3", "172.20.20.3", 5000, 1140, "Description3");

////////////////////////////////////////
// Update the properties for each server
////////////////////////////////////////

serverEnum = new Enumerator(servers);

for (; !serverEnum.atEnd(); serverEnum.moveNext())
{
    server = serverEnum.item();

    ////////////////////////////////////////
    // Add the data set file format entries
    ////////////////////////////////////////

    fileFormats = server.DataSetFileFormats;

    fileFormats.Add(-1, "**.ASCII.TEXT",   "AT");
    fileFormats.Add(-1, "**.ASCII.DATA",   "AD");
    fileFormats.Add(-1, "**.UNICODE.TEXT", "UT");
    fileFormats.Add(-1, "**.EBCDIC.TEXT",  "ET");
    fileFormats.Add(-1, "**.EBCDIC.DATA",  "ED");
    fileFormats.Add(-1, "**.BINARY",       "BT");

    ////////////////////////////////////
    // Add the Unix file format entries
    ////////////////////////////////////

    fileFormats = server.UnixFileFormats;

    fileFormats.Add(-1, "*.TEXT",  "AT");
    fileFormats.Add(-1, "*.UTEXT", "UT");
    fileFormats.Add(-1, "*.BIN",   "B" );

    ////////////////////////////////
    // Add the library type entries
    ////////////////////////////////

    libTypes = server.LibTypes;

    libTypes.Add(-1, "**.LIBRARY",  "L");
    libTypes.Add(-1, "**.PANVALET", "P");

    //////////////////////////////////
    // Add the file extension entries
    //////////////////////////////////

    fileExtensions = server.FileExtensions;

    fileExtensions.Add(-1, "**.CNTL",  "jcl");
    fileExtensions.Add(-1, "**.COBOL", "cbl");
    fileExtensions.Add(-1, "**.LIST",  "txt");
    fileExtensions.Add(-1, "**.WORD",  "doc");
    fileExtensions.Add(-1, "**.EXCEL", "xls");

    //////////////////////////////////////
    // Add the profiles for new data sets
    //////////////////////////////////////

    dsProfiles = server.DataSetProfiles;

    dsProfiles.Add(-1, "**.DATA",  "SEQ", "FB", 80, 0, "DATACLS1", "STORCLS1",
     "MGMTCLS1", "TRK",   2,  1, 5, "SYSDA", "VOL001");
```

```
 dsProfiles.Add(-1, "**.TEMP",  "SEQ", "FB", 80, 0, "DATACLS2", "STORCLS2",
  "MGMTCLS2", "CYL",   2,  1, 5, "SYSDA", "VOL002");
dsProfiles.Add(-1, "**.LIST",  "SEQ", "VB", 80, 0, "",        "",        "",
  "BLK", 500, 50, 5, "SYSDA", "");

 ///////////////////////
 // Add data set folders
 ///////////////////////

 folder = server.DataSetFolder;
 subfolders = folder.Subfolders;

 ///////////////////////////////////////////////
 // "My DataSets" folder for all user's data sets
 ///////////////////////////////////////////////

 subfolder = subfolders.Add("My DataSets");

 filters  = subfolder.Filters;
 prefixes = subfolder.Prefixes;

 filters.Add(userID + ".**");

 prefixes.Add(-1, "**", userID);

 ////////////////////////////////////////////////
 // "My Source" folder for user's source libraries
 ////////////////////////////////////////////////

 subfolder = subfolders.Add("Source");

 filters  = subfolder.Filters;
 prefixes = subfolder.Prefixes;

 filters.Add(userID + ".**.COBOL");
 filters.Add(userID + ".**.ASM");

 prefixes.Add(-1, "**", userID);

 //////////////////
 // Add job folders
 //////////////////

 folder = server.JobFolder;
 subfolders = folder.Subfolders;

 //////////////////////////////////////////
 // "My Jobs" folder for jobs owned by user
 //////////////////////////////////////////

 subfolder = subfolders.Add("My Jobs", "QU", userID);

 ///////////////////////////////////////////////////////
 // "ChangeMan" folder for job names prefixed with "CMN"
 ///////////////////////////////////////////////////////

 subfolder = subfolders.Add("ChangeMan", "QN", "CMN*");

 //////////////////////////////////////
 // "Active" folder for all active jobs
 //////////////////////////////////////

 subfolder = subfolders.Add("Active", "A");
```

```
    /////////////////////////
    // Add ChangeMan folders
    /////////////////////////

    folders = server.ChangeManFolders;

    folders.Add("ChangeMan-Prod", 3000, "Production ChangeMan");
    folders.Add("ChangeMan-Test", 3001, "Test ChangeMan");


    folderEnum = new Enumerator(folders);

    for (; !folderEnum.atEnd(); folderEnum.moveNext())
    {
        folder = folderEnum.item();

        /////////////////////////////////////////
        // Add the ChangeMan file format entries
        /////////////////////////////////////////

        fileFormats = folder.FileFormats;

        fileFormats.Add(-1, "SRC",  "AT");
        fileFormats.Add(-1, "DOC",  "UT");
        fileFormats.Add(-1, "BIN",  "B");
    }
}
```

Using Windows Task Scheduler

The Windows Task Scheduler allows you to schedule programs to run at specified times. For example, you can schedule nightly job cycles to run automatically.

The following example shows how to log on to a z/OS server and submit a job. This .bat file contains commands to execute scripts that use the ***Logon*** and ***SubmitJCL*** methods. For examples of these scripts, see "Logging on to a Server" on page 188 and "Submitting JCL to a Server" on page 192.

```
Rem  This is a batch file that logs on to the z/OS Host.
Rem  After logging on, a JCL member on the Host is submitted.

C:\MyJobs\WSCRIPT Logon.cs      HOSTNAME USERID PASSWORD
C:\MyJobs\WSCRIPT SubmitJCL.cs  M:\USER999.CNTL.JCL\MYJOB

Say 'Your Job was Submitted'
Pause
```

You can schedule this .bat file to run automatically using the Windows Task Scheduler. To access the Windows Task Scheduler:

1. Choose **Programs>Accessories>System Tools>Scheduled Tasks** from the Windows **Start** Menu. The **Scheduled Tasks** dialog box appears.
2. Click **Add Scheduled Task** and a wizard will guide you through the process.

Index