IBM Endpoint Manager
Version 9.1

API Reference Guide

IBM

IBM Endpoint Manager
Version 9.1

API Reference Guide

Note

Before using this information and the product it supports, read the information in “Notices” on page 87.

This edition applies to version 9, release 1, modification level 0 of IBM Endpoint Manager and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. API Reference Guide

IBM Endpoint Manager has extensibility built-in to allow you to create content directly from the console in the form of custom Fixlets and tasks. There are times, however, when you need direct programmatic access to the relevance engine at the core of IBM Endpoint Manager. IBM Endpoint Manager therefore makes available a set of application programming interfaces (APIs).

This guide describes the various APIs, including server, client, dashboard, and Web Reports, that you can use to control all aspects of the IBM Endpoint Manager engine from your own scripts.

This guide describes the API available to IBM Endpoint Manager Version 9.1.

Chapter 2. Server API

This section of the guide shows you how to access programming features of the console server engine by using your own scripts.

With the console you can create your own custom tasks and actions, giving you a wide range of options for controlling and monitoring your networks in accordance with your own best practices. As powerful as that is, there are some problems that are best solved with direct access to the console engine itself. The power of the console then becomes yours to control and automate with custom scripts.

Using the server application programming interface, you can create relevance expressions and execute them as if you were logged in to the console. You can use your own scripts to generate fresh content, including Fixlets, tasks, actions, baselines, properties, and analyses. The Server API consists of a set of Component Object Model (COM) objects that are distributed in the form of a DLL file. The API was created with Active Template Library (ATL) and C++ but can be accessed by using COM from a variety of languages and scripting environments, including VBScript, JScript, Perl, and C#.

To use the API, the calling application creates a BESAPI object with an associated set of methods and properties. There are different BESAPI objects for each task one might undertake by using the Tivoli Endpoint Manager console, such as creating an action or task (BESAPI.XMLImporter) or managing site subscriptions (BESAPI.SiteManager).

Note: Tivoli Endpoint Manager was previously called the BigFix Enterprise System (BES), and you can still see this legacy in the names of the API calls.

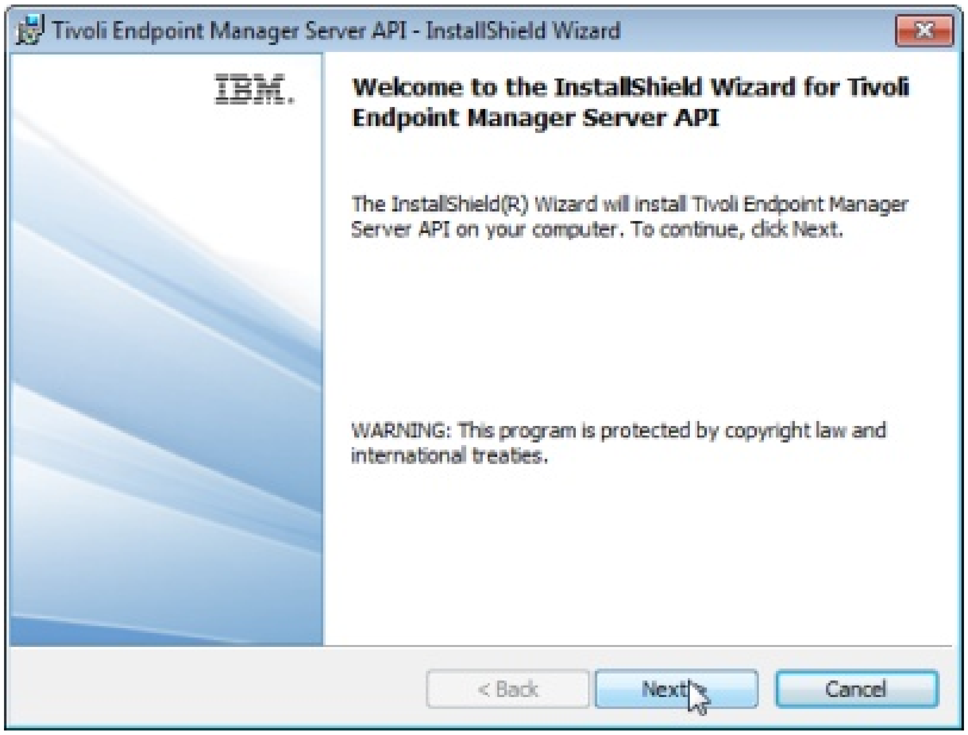
The content that you create for the Server API is in XML format that is saved in files with an extension of *.bes*. This guide provides you with a detailed description of the XML schema that is used by .bes files.

Installation

Before you can install the BESAPI, you must obtain the site certificate from IBM, create the *action site masthead* file, and install the server and console. Verify that the server is functioning correctly by using the *IBM Endpoint Manager Diagnostics Tool*. For more details on these steps, see the*IBM Endpoint Manager Installation Guide*.

After you successfully install IBM Endpoint Manager, you can install the Server API by following these steps:

1. Download the BES-ServerAPI program from the same IBM site where you downloaded the IBM Endpoint Manager program. It has a name of the form: `BigFix-BES-ServerAPI-9.1.933.0.exe`. The numbers in the name correspond to the version. This version must match the version of IBM Endpoint Manager that you want to access with the API.
2. After you download it, run the program. The installer window opens.

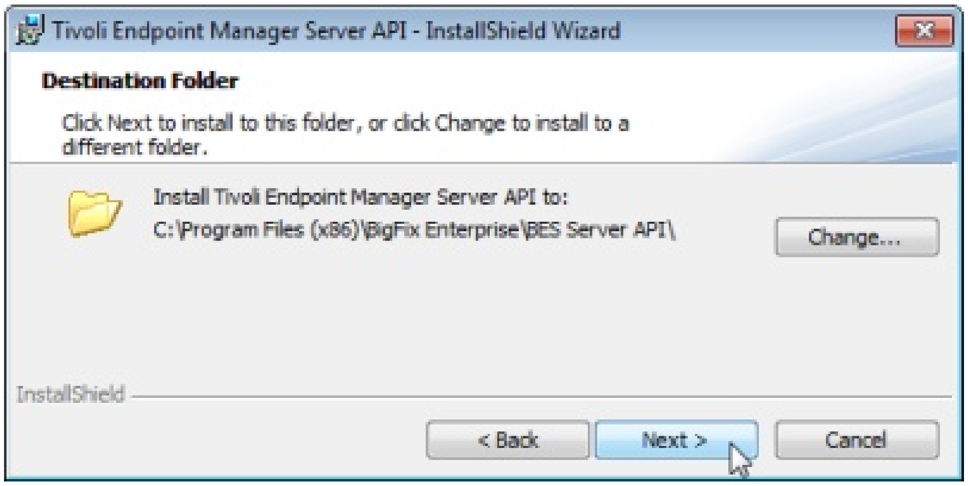

Tivoli Endpoint Manager Server API - InstallShield Wizard

Welcome to the InstallShield Wizard for Tivoli Endpoint Manager Server API

The InstallShield(R) Wizard will install Tivoli Endpoint Manager Server API on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

3. Click Next. The default destination folder is displayed. The location is within your existing IBM Endpoint Manager (or BigFix Enterprise) folder:


Tivoli Endpoint Manager Server API - InstallShield Wizard

Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install Tivoli Endpoint Manager Server API to:
C:\Program Files (x86)\BigFix Enterprise\BES Server API\

4. Click *Change* to select a different folder or click *Next* to proceed.
5. On the subsequent screen, click *Install*.
6. When the installation is done, click *Finish*.

The installer creates a folder named BES Server API, typically in the BigFix Enterprise folder. In this folder are the API DLLs you need to access the platform.

Support and assistance for using the Server API is not directly covered by IBM. However, for information and examples, visit IBM developerWorks Forum.

BES XML documents

The Server API allows you to import content into the console so you can create and automate your own Fixlets, tasks, actions, properties, and more. This content is contained in a BES XML document. The easiest way to learn about BES XML documents is to create one in the console and then export it. Here is how:

1. To export a Fixlet, task, baseline, action, analysis or computer group, just right-click one from the console list panel and select *Export* from the pop-up menu.
2. To export retrieved properties, open *Tools -> Manage Properties*, select a property, and click *Export* .
3. Select a folder for the exported *.bes* file.
4. Open the saved .bes file in an XML editor or plain-text editor.

The BES Schema is defined in the file *BES.xsd*. You can find this file in the*Program Files\BigFix Enterprise\BES Console\Reference* folder on your console computer. Familiarize yourself with this file. It contains the definitions for all the permissible content you can submit to the Server API. Any content that you create must validate against this schema or it is rejected during import. You can also use this file when you create your scripts, because many XML authoring tools include the schema to make XML authoring easier.

Note: In XML, the '<', '>', '&', and '"' characters must be escaped as "&lt;", "&gt;", &amp;", and "&quot;" respectively. Because many elements can contain HTML, it can be easier to wrap the element's unescaped contents in a CDATA tag, as in "<![CDATA[...]]>". CDATA tags cannot be nested, so this technique does not work if the data already contains the CDATA end tag ("]]>").

Connecting to the API

In versions earlier than 8.2, you had to have an ODBC connection that is defined and have access to the signing keys for the deployment to authenticate to the API. In version 8.2, the use of signing keys and ODBC connections was discontinued. You only need the server hostname and login credentials to connect to the API.

Your existing code should continue to work properly, assuming that you still have access to the keys and they have not been changed in the deployment after your upgrade. However, using signing keys results in the execution of additional steps in the connection process, and you might want to consider migrating to the simpler login to speed things up.

In version 8.1 and earlier, you were required to create a SigningKeys object, direct that to the location of the keys, and then ascertain their validity. The sequence was:

```
SetDefaultDSN( $DSN )
 SetPrivateKeyPath( $Username, $Path )
 AreSigningKeysValid( $Username, $Password )
```

In version 8.2, the SigningKeys object is not required. Instead, from any BESAPI object, you can use:

```
SetServer( $Hostname )
```

Each subsequent call uses this server for communication. No ODBC connection must be established and no signing key is required. The new API, however, does need to get an SSL Certificate from the server and store that locally. The location where this key is stored is either the current path of the executable using the API or from a registry key. That key is:

```
 HKEY_LOCAL_MACHINE\Software\BigFix\(ApplicationName)\base.
```

The ApplicationName is set from any object by using:

```
SetAppName( $ApplicationName )
```

If the SSL Certificate on the server changes, it might be necessary to delete the old certificate from this location. It is also the location where information fetched from the server is stored, such as site exports.

Note: For versions earlier than 8.2, Patch 3, the API still requires that you make unnecessary calls to the signing key related methods. To avoid this situation, make sure that you have Patch 3 or later.

Examples

Top-level BES XML elements

A BES XML file has a *BES* document element, containing at least one Fixlet, task, baseline, analysis, SingleAction, MultipleActionGroup, ComputerGroup, or property element. This is collection of possible elements that you can include in your .bes files, formatted as XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<BES xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="BES.xsd">
      <Fixlet>
            ...
      </Fixlet>
      <Task>
               ...
        </Task>
      <Baseline>
               ...
      </Baseline>
        <Analysis>
               ...
      </Analysis>
      <SingleAction>
            ...
      </SingleAction>
      <MultipleActionGroup>
            ...
      </MultipleActionGroup>
      <ComputerGroup>
            ...
      </ComputerGroup>
      <Property>
            ...
      </Property>
</BES>
```

Many of the programs you write to access the API take advantage of .bes XML files. Typically, your program defines the XML file in a variable that you can then process with a relevant import command. That causes your new content to appear in the console.

Running an action

There are several general principles that are involved with all .bes files. To communicate with the Tivoli Endpoint Manager platform, you must create API objects, that you can then use to issue commands or to set and retrieve properties. For example, to propagate an action with a script, first create a BESAPI.FixletActionCreator object:

```
var actionCreator = CreateObject("BESAPI.FixletActionCreator");
```

or

```
var actionCreator = new
ActiveXObject("BESAPI.FixletActionCreator");
```

Then use the appropriate *set* methods on that object to specify the attributes of the action, such as which computers to target, when to run and the content of the action (that is, when it is relevant, and the script to execute). For example:

```
actionCreator.SetTargetComputers("|12345678|");
```

Finally, commit the action to the database and send it to the client computers using the *DoPropagation* method. For example:

```
actionCreator.DoPropagation("joe","mypassword");
window.alert("Propagated action id #" + actionCreator.actionID);
```

Accessing the API with Perl

You can access the Server API with many different programming languages. This section describes access using Perl, a popular choice for creating quick scripts.

The API package in Perl

This is the BESAPI package that you can include with your own scripts:

```
# BESAPI.pm

package BESAPI;

use strict;
use Win32::OLE;

Win32::OLE->Option( Warn => sub { die Win32::OLE->LastError() . "\n"; } );

sub new
{
      my ( $perlType, $besAPIType, $server ) = @_;
   my $self = {};
      $self->{type} = $besAPIType;
      $self->{object} = Win32::OLE->new( "BESAPI." . $besAPIType );
      $self->{object}->SetServer( $server );
   return bless $self, $perlType;
}

our $AUTOLOAD;
sub AUTOLOAD
{
   my $self = shift;
      $AUTOLOAD =~ s/^.*:://;
      if ( $AUTOLOAD eq "DESTROY" )
            { return; }

      my $besAPIObject = $self->{object};
   my $result = $self->{object}->Invoke( $AUTOLOAD, @_ );
   if ( $besAPIObject->DiagnosticMessage() ) {
        die "BESAPI." . $self->{type} . "." . $AUTOLOAD .
        " error: " . $self->{object}->DiagnosticMessage() . "\n";
   }

      return $result;
}

# included for use with versions prior to 8.2
sub CheckSigningKeys
{
      my ( $dsn, $username, $password, $privateKey ) = @_;
      my $signingKeys = BESAPI->new( "SigningKeys" );
      $signingKeys->SetDefaultDSN( $dsn );
      $signingKeys->SetPrivateKeyPath( $username, $privateKey );
      if ( !$signingKeys->AreSigningKeysValid( $username, $password ) )
            { die "SigningKeys not valid\n"; }
}

sub FixletMessage
{
      my ( $siteID, $FixletID, $username, $password, $server ) = @_;
      my $Fixlet = BESAPI->new( "FixletMessage", $server );
      $Fixlet->Load( $siteID, $FixletID, $username, $password );
      return $Fixlet;
}

1;
```

Importing a file with Perl

This Perl script lets you import a .bes file into the console. Pass your user name and password, along with the path of the .bes file and the name of the custom site where it is located. The usage is as follows:

```
ImportFile.pl <username> <password> <bes file path> <custom site name>
```

This is the code:

```
# TakeActionFromFixlet.pl

use strict;
use FindBin;
use lib $FindBin::Bin;
use BESAPI;
use Cwd 'abs_path';
use File::Basename;

if ( @ARGV != 4 && @ARGV != 5 ) {
      print fileparse(abs_path($0)) . " <server> <username> <password>
      <bes file path> [<custom site name>]";
      exit( 0 );
}

my ( $server, $username, $password, $besFile, $customSite ) = @ARGV;
if ( !$customSite )
      { $customSite = ""; }

eval
{
      my $xmlImporter = BESAPI->new( "XMLImporter", $server );
      my $ids = $xmlImporter->ImportFile( $besFile, $customSite, $username,
                                          $password );

      print "IDs: " . join( " ", @$ids ) . "\n";
};
```

```
if ( $@ )
{
      print "$@\n";
}
```

Examining a Fixlet with Perl

This Perl script finds a FixletMessage with a specific site ID and Fixlet ID and prints all the accessible information about that Fixlet. This program requires you to enter your name, password, and Fixlet identification in the program itself. The text that is highlighted in blue is where the assignments are made:

```
# FixletMessage.pl

use strict;
use FindBin;
use lib $FindBin::Bin;
use BESAPI;

my $username = "operator";
my $password = "bigfix";
my $server   = "bes-server-hostname";

my $siteID = 1;
my $FixletID = 177;

my $Fixlet = BESAPI::FixletMessage( $siteID, $FixletID, $username,
   $password, $server );

print "Name: "              . $Fixlet->Name() . "\n";
print "SiteName: "          . $Fixlet->SiteName() . "\n";
print "SiteDisplayName: "   . $Fixlet->SiteDisplayName() . "\n";
print "SiteURL: "           . $Fixlet->SiteURL() . "\n";
print "SiteID: "            . $Fixlet->SiteID() . "\n";
print "FixletID: "          . $Fixlet->FixletID() . "\n";
print "IsTask: "            . $Fixlet->IsTask() . "\n";
print "IsAnalysis: "        . $Fixlet->IsAnalysis() . "\n";
print "IsPlainFixlet: "     . $Fixlet->IsPlainFixlet() . "\n";
print "IsBaseline: "        . $Fixlet->IsBaseline() . "\n";
print "DownloadSize: "      . $Fixlet->DownloadSize() . "\n";
print "Source: "            . $Fixlet->Source() . "\n";
print "SourceID: "          . $Fixlet->SourceID() . "\n";
print "SourceSeverity: "    . $Fixlet->SourceSeverity() . "\n";
print "SourceReleaseDate: " . $Fixlet->SourceReleaseDate() . "\n";
print "Category: "          . $Fixlet->Category() . "\n";
print "IsDeleted: "         . $Fixlet->IsDeleted() . "\n";
print "Relevance: "         . $Fixlet->Relevance() . "\n";
print "Current FIPS Mode: " . $Fixlet->CurrentFIPSMode( $username, $password );

if ( $Fixlet->IsAnalysis() )
{
    print "PropertyIDSet: "       . $Fixlet->PropertyIDSet() . "\n";
    print "IsGloballyActivated: " . $Fixlet->IsGloballyActivated() . "\n";
    print "IsLocallyActivated: "  . $Fixlet->IsLocallyActivated() . "\n";
    print "CanActivate: "         . $Fixlet->CanActivate() . "\n";
    print "CanDeactivate: "       . $Fixlet->CanDeactivate() . "\n";
}

print "\n\n";
print "Message: " . $Fixlet->Message() . "\n";
print "HTML: \n\n" . $Fixlet->HTML() . "\n";

if ( $Fixlet->IsPlainFixlet() || $Fixlet->IsTask() )
{
    my $action = 0;
    print "ActionScript: " . $Fixlet->ActionScript( $action ) . "\n";
    print "ActionScriptMIMEType: " . $Fixlet->ActionScriptMIMEType
```

```
                                  ( $action ) . "\n";
    print "ActionScriptTypeName: " . $Fixlet->ActionScriptTypeName
                                  ( $action ) . "\n";
}
```

Managing a custom site with Perl

You can use a Perl script to manage your custom sites. You can create or delete sites, as well as adding and removing readers, writers, and owners of the site. Run the program without input to see the allowed arguments:

```
Usage:
CustomSiteManager.pl <CreateSite>   <dsn> <username> <password> <pvk file>
                     <sitename>
CustomSiteManager.pl <DeleteSite>   <dsn> <username> <password> <pvk file>
                     <sitename>
CustomSiteManager.pl <AddWriter>    <dsn> <username> <password> <pvk file>
                     <sitename> <writername>
CustomSiteManager.pl <RemoveWriter> <dsn> <username> <password> <pvk file>
                     <sitename> <writername>
CustomSiteManager.pl <AddOwner>     <dsn> <username> <password> <pvk file>
                     <sitename> <ownername>
CustomSiteManager.pl <RemoveOwner>  <dsn> <username> <password> <pvk file>
                     <sitename> <ownername>
CustomSiteManager.pl <AddReader>    <dsn> <username> <password> <pvk file>
                     <sitename> <readername>
CustomSiteManager.pl <RemoveReader> <dsn> <username> <password> <pvk file>
                     <sitename> <readername>
```

This is the Perl script:

```
# CustomSiteManager.pl

use strict;
use FindBin;
use lib $FindBin::Bin;
use BESAPI;

if ( @ARGV == 0 )
{
      Usage();
      exit( 0 );
}

my ( $server,  $username, $password, $sitename, $subject )
 = ( $ARGV[1], $ARGV[2],  $ARGV[3],  $ARGV[4],  $ARGV[5] );

my $customSiteManager = BESAPI->new( "CustomSiteManager", $server );

if ( $ARGV[0] =~ /CreateSite/i )
{
      $customSiteManager->CreateCustomSite( $sitename, $username, $password );
}
elsif ( $ARGV[0] =~ /DeleteSite/i )
{
      $customSiteManager->DeleteCustomSite( $sitename, $username, $password );
}
elsif ( $ARGV[0] =~ /AddWriter/i )
{
      $customSiteManager->AddCustomSiteWriter( $sitename, $subject, $username,
        $password );
}
elsif ( $ARGV[0] =~ /RemoveWriter/i )
{
      $customSiteManager->RemoveCustomSiteWriter( $sitename, $subject, $username,
        $password );
}
```

```
elsif ( $ARGV[0] =~ /AddOwner/i )
{
      $customSiteManager->AddCustomSiteOwner( $sitename, $subject, $username,
        $password );
}
elsif ( $ARGV[0] =~ /RemoveOwner/i )
{
      $customSiteManager->RemoveCustomSiteOwner( $sitename, $subject, $username,
        $password );
}
elsif ( $ARGV[0] =~ /AddReader/i )
{
      $customSiteManager->AddCustomSiteReader( $sitename, $subject, $username,
        $password );
}
elsif ( $ARGV[0] =~ /RemoveReader/i )
{
      $customSiteManager->RemoveCustomSiteReader( $sitename, $subject, $username,
        $password );
}
else
{
      Usage();
}

sub Usage
{
      print "Usage:\n";
      print File::Basename::basename( $0 ) . " <CreateSite>   <server> <username>
       <password> <sitename>\n";
      print File::Basename::basename( $0 ) . " <DeleteSite>   <server> <username>
       <password> <sitename>\n\n";
      print File::Basename::basename( $0 ) . " <AddWriter>    <server> <username>
       <password> <sitename> <writername>\n";
      print File::Basename::basename( $0 ) . " <RemoveWriter> <server> <username>
       <password> <sitename> <writername>\n\n";
      print File::Basename::basename( $0 ) . " <AddOwner>     <server> <username>
       <password> <sitename> <ownername>\n";
      print File::Basename::basename( $0 ) . " <RemoveOwner>  <server> <username>
       <password> <sitename> <ownername>\n\n";
      print File::Basename::basename( $0 ) . " <AddReader>    <server> <username>
       <password> <sitename> <readername>\n";
      print File::Basename::basename( $0 ) . " <RemoveReader> <server> <username>
       <password> <sitename> <readername>\n";
}
```

Creating a Fixlet with an action using Perl

The following script loads a FixletMessage with a specific site ID and Fixlet ID and takes an action from that Fixlet, by using the FixletMessage.ActionXML and XMLImporter.ImportAction methods. This program requires you to enter your name, password, and Fixlet identification in the program itself. The text that is highlighted in blue is where the assignments are made:

```
# TakeActionFromFixlet.pl

use strict;
use FindBin;
use lib $FindBin::Bin;
use BESAPI;

my $username = "bigfix";
my $password = "bigfix";
my $server   = "bes-server-hostname";

my $siteID   = 3096;
my $FixletID = 3;
my $action   = 0;

my $Fixlet = BESAPI::FixletMessage( $siteID, $FixletID, $username, $password,
                                    $server );

my $settingsXML =
            "<?xml version=\"1.0\"?>\n" .
            "<ActionSettings>\n" .
                "<Settings>\n" .
                "<ActionUITitle>title</ActionUITitle>\n" .
                "<PreActionShowUI>true</PreActionShowUI>\n" .
                "<PreAction>\n" .
                    "<Text>preaction description</Text>\n" .
                    "<AskToSaveWork>true</AskToSaveWork>\n" .
                    "<ShowActionButton>true</ShowActionButton>\n" .
                    "<ShowCancelButton>true</ShowCancelButton>\n" .
                    "<DeadlineBehavior>ForceToRun</DeadlineBehavior>\n" .
                    "<DeadlineType>Absolute</DeadlineType>\n" .
                    "<DeadlineOffset>PT23H58M54.000000S</DeadlineOffset>\n" .
                    "<ShowConfirmation>true</ShowConfirmation>\n" .
                    "<Confirmation>confirmation message</Confirmation>\n" .
                "</PreAction>\n" .
                "<HasRunningMessage>true</HasRunningMessage>\n" .
                "<RunningMessage>\n" .
                    "<Text>running text</Text>\n" .
                "</RunningMessage>\n" .
                "<HasTimeRange>false</HasTimeRange>\n" .
                "<HasStartTime>true</HasStartTime>\n" .
                "<StartDateTimeOffset>-PT1M6.000000S</StartDateTimeOffset>\n" .
                "<HasEndTime>true</HasEndTime>\n" .
                "<EndDateTimeOffset>P1DT23H58M54.000000S</EndDateTimeOffset>\n" .
                "<HasDayOfWeekConstraint>false</HasDayOfWeekConstraint>\n" .
                "<ActiveUserRequirement>RequireUser</ActiveUserRequirement>\n" .
                "<ActiveUserType>LocalUsers</ActiveUserType>\n" .
                "<HasWhose>false</HasWhose>\n" .
                "<Reapply>false</Reapply>\n" .
                "<HasReapplyLimit>false</HasReapplyLimit>\n" .
                "<HasReapplyInterval>false</HasReapplyInterval>\n" .
                "<HasRetry>false</HasRetry>\n" .
                "<HasTemporalDistribution>false</HasTemporalDistribution>\n" .
                "<PostActionBehavior Behavior=\"Nothing\"></PostActionBehavior>\n" .
                "<IsOffer>true</IsOffer>\n" .
                "<OfferCategory>offer category</OfferCategory>\n" .
                "<OfferDescriptionHTML>offer description</OfferDescriptionHTML>\n" .
                "</Settings>" .
            "</ActionSettings>";

my $actionXMLDocument = $Fixlet->ActionXML( $action, $settingsXML );

my $targetXMLA =
    "<?xml version=\"1.0\"?>" .
    "<BESActionTarget>" .
        "<ComputerName>a</ComputerName>" .
        "<ComputerName>b</ComputerName>" .
        "<ComputerName>c</ComputerName>" .
    "</BESActionTarget>";

my $targetXMLB =
    "<?xml version=\"1.0\"?>" .
    "<BESActionTarget>" .
        "<ComputerID>34</ComputerID>" .
        "<ComputerID>12704810</ComputerID>" .
    "</BESActionTarget>";

my $targetXMLC =
    "<?xml version=\"1.0\"?>" .
    "<BESActionTarget>" .
        "<CustomRelevance>exists file \"c:\\virus\"</CustomRelevance>" .
    "</BESActionTarget>";

my $xmlImporter = BESAPI->new( "XMLImporter", $server );
my $actionID = $xmlImporter->ImportAction( $actionXMLDocument->XML(), $targetXMLA,
   $Fixlet->SiteID(), $Fixlet->FixletID(), $username, $password );

print "ActionID: $actionID\n";
```

Creating a Fixlet with an action using C#

You can do the same as the previous example with any flavor of C as well as Perl. This is the C# equivalent of the previous script:

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

namespace BESAPITest
{
  class Program
  {
    static void Main(string[] args)
    {
       try
       {
        string username = "bigfix";
        string password = "bigfix";
        string server   = "bes-server-hostname";

        int siteID = -2147481618;
        int FixletID = 32;
        int action = 0;

        BESAPILib.FixletMessage Fixlet = new BESAPILib.FixletMessage();
                  Fixlet.SetServer( server );
        Fixlet.Load( siteID, FixletID, username, password );
        if ( Fixlet.DiagnosticMessage.Length != 0 )
            throw new Exception( Fixlet.DiagnosticMessage );

        string settingsXML =
               "<?xml version=\"1.0\"?>\n" +
               "<ActionSettings>\n" +
                   "<Settings>\n" +
                   "<ActionUITitle>title</ActionUITitle>\n" +
                   "<PreActionShowUI>true</PreActionShowUI>\n" +
                   "<PreAction>\n" +
                       "<Text>preaction description</Text>\n" +
                       "<AskToSaveWork>true</AskToSaveWork>\n" +
                       "<ShowActionButton>true</ShowActionButton>\n" +
                       "<ShowCancelButton>true</ShowCancelButton>\n" +
                       "<DeadlineBehavior>ForceToRun</DeadlineBehavior>\n" +
                       "<DeadlineType>Absolute</DeadlineType>\n" +
                       "<DeadlineOffset>PT23H58M54.000000S</DeadlineOffset>\n" +
                       "<ShowConfirmation>true</ShowConfirmation>\n" +
                       "<Confirmation>confirmation message</Confirmation>\n" +
                    "</PreAction>\n" +
                    "<HasRunningMessage>true</HasRunningMessage>\n" +
                    "<RunningMessage>\n" +
                        "<Text>running text</Text>\n" +
                    "</RunningMessage>\n" +
                    "<HasTimeRange>false</HasTimeRange>\n" +
                    "<HasStartTime>true</HasStartTime>\n" +
                    "<StartDateTimeOffset>-PT1M6.000000S
                     </StartDateTimeOffset>\n" +
                    "<HasEndTime>true</HasEndTime>\n" +
                    "<EndDateTimeOffset>P1DT23H58M54.000000S
                     </EndDateTimeOffset>\n" +
                    "<HasDayOfWeekConstraint>false</HasDayOfWeekConstraint>\n" +
                    "<ActiveUserRequirement>RequireUser
                     </ActiveUserRequirement>\n" +
                    "<ActiveUserType>LocalUsers</ActiveUserType>\n" +
                    "<HasWhose>false</HasWhose>\n" +
                    "<Reapply>false</Reapply>\n" +
                    "<HasReapplyLimit>false</HasReapplyLimit>\n" +
                    "<HasReapplyInterval>false</HasReapplyInterval>\n" +
                    "<HasRetry>false</HasRetry>\n" +
                    "<HasTemporalDistribution>false</HasTemporalDistribution>\n" +
                    "<PostActionBehavior Behavior=\"Nothing\">
                                   </PostActionBehavior>\n" +
                    "<IsOffer>true</IsOffer>\n" +
                    "<OfferCategory>offer category</OfferCategory>\n" +
                    "<OfferDescriptionHTML>offer description
                    </OfferDescriptionHTML>\n" +
                    "</Settings>" +
               "</ActionSettings>";

 BESAPILib.IXMLDOMDocument actionXMLDocument = Fixlet.get_ActionXML
           ( action,System.Text.Encoding.Unicode.GetBytes( settingsXML ) );
 if ( Fixlet.DiagnosticMessage.Length != 0 )
    throw new Exception( Fixlet.DiagnosticMessage );

 string targetXMLA =
     "<?xml version=\"1.0\"?>" +
     "<BESActionTarget>" +
        "<ComputerName>a</ComputerName>" +
        "<ComputerName>b</ComputerName>" +
        "<ComputerName>c</ComputerName>" +
     "</BESActionTarget>";

 string targetXMLB =
     "<?xml version=\"1.0\"?>" +
     "<BESActionTarget>" +
          "<ComputerID>34</ComputerID>" +
          "<ComputerID>12704810</ComputerID>" +
     "</BESActionTarget>";

 string targetXMLC =
     "<?xml version=\"1.0\"?>" +
     "<BESActionTarget>" +
         "<CustomRelevance>exists file \"c:\\virus\"</CustomRelevance>" +
     "</BESActionTarget>";

 BESAPILib.XMLImporter xmlImporter = new BESAPILib.XMLImporter();
         xmlImporter.SetServer( server );
 int actionID = xmlImporter.ImportAction( System.Text.Encoding.Unicode.GetBytes
                                                    ( actionXMLDocument.xml ),
                                          System.Text.Encoding.Unicode.GetBytes
                                                    ( targetXMLA ),
                                          (UInt32) Fixlet.siteID,
                                          (UInt32) Fixlet.FixletID,
                                          username,
                                          password );
 if ( xmlImporter.DiagnosticMessage.Length != 0 )
     throw new Exception( xmlImporter.DiagnosticMessage );

  System.Console.WriteLine( "ActionID: " + actionID );
}
catch ( System.Exception e )
{
     System.Console.WriteLine( e.Message );
```

```
      }
    }
  }
}
```

The API Objects

The objects in the Server API are listed here and are detailed in the following sections:

- *XMLImporter:* Use to import .bes files into the Tivoli Endpoint Manager console.
- *ActionStopper:* Use to stop the specified action.
- *RetrievedProperty:*Use to define and modify retrieved properties.
- *FixletMessage:* Use to set and access Fixlets.
- *SettingsActionCreator:* Use to create and manage settings.
- *SiteManager:* Use to create and assign rights to custom sites.
- *ActionSiteMasthead:* Use to manage the mastheads for your custom sites.
- *SigningKeys:* Use to manage sign-in security for the API under Tivoli Endpoint Manager.

BESAPI.XMLImporter

The XML Importer provides the main functionality of the Server API. Use it to import a special XML file, with the extension of *.bes*, into the platform engine for evaluation.

Note: Although the console lets you import BES XML documents with computer groups in them, the BESAPI.XMLImporter object does not support the creation of computer groups.

XMLImporter Methods	Description

`HRESULT SetAppName (`
`      BSTR appName`
`);`
	Set the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.

`HRESULT SetDSN (`
`      BSTR dsn`
`);`
	Set the name of the ODBC DSN to use when connecting to the database. Default is 'BES_bfenterprise'.

`HRESULT ImportFile (`
`      BSTR xmlFilePath,`
`   BSTR customSiteName,`
`      BSTR username,`
`      BSTR password,`
`      [out, retval] VARIANT *ids`
`);`
	Connects to the database with the username and password. Reads the contents of the specified xml file, and imports it into the database, by using the specified custom site name, or if that is the empty string, then the operator's action site, or the master action site, if the operator is a master operator. It returns an array containing the database IDs of the objects created. Note that only one site is propagated, so importing more than one object, or by using the “auto activate analysis” feature might result in an object being created in the database without propagating that object to the BES agents.

`HRESULT Import ( SAFEARRAY(byte) xml, BSTR customSiteName, BSTR username, BSTR password, [out, retval] VARIANT *ids );`	Connects to the database with the username and password. Imports the xml specified into the database, with the specified custom site name, or if that is the empty string, then the operator's action site, or the master action site, if the operator is a master operator. Returns an array containing the database IDs of the objects created.
`HRESULT ImportAction ( SAFEARRAY(byte) actionXML, SAFEARRAY(byte) targetXML, UINT sourceSiteID, UINT sourceFixletID, BSTR username, BSTR password, [out, retval] long *id );`	Connects to the database with the username and password. Imports the action specified by the actionXML, by using the targetXML for targeting endpoints, and by using the specified source information. The appropriate site (operator site or master action site) is propagated depending on whether or not the username refers to a master operator. Returns an integer containing the database id of the action object created.

XMLImporter Properties	Description
`HRESULT DiagnosticMessage ( [out, retval] BSTR *pVal );`	If a method fails, this property returns a string containing a diagnostic message.
`HRESULT StatusMessage ( [out, retval] BSTR *pVal );`	Returns a string describing the status of the current operation (for example, propagation).
`HRESULT CurrentFIPSMode( BSTR username, BSTR password, [out, retval] BOOL* pVal );`	This is a function added in version 7.1 to every BESAPI interface. It retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.ActionStopper

ActionStopper Methods	Description
`HRESULT SetAppName ( BSTR appName );`	Sets the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.
`HRESULT SetDSN ( BSTR dsn );`	Sets the name of the ODBC DSN to use when connecting to the database. Default is 'BES_bfenterprise'.
`HRESULT StopAction ( UINT actionID, BSTR dbUserName, BSTR password );`	Stops the existing action in the database with the specified ID number, and uses the username and password to propagate a new version of the action site.
`HRESULT DeleteAction ( UINT actionID, BSTR dbUserName, BSTR password );`	Permanently deletes the existing action in the database with the specified ID number, and uses the username and password to propagate a new version of the action site. The action must be stopped or expired. Open actions cannot be deleted.

ActionStopper Properties	Description
`HRESULT DiagnosticMessage ( [out, retval] BSTR *pVal );`	If a method fails, this property returns a string containing a diagnostic message.
`HRESULT StatusMessage ( [out, retval] BSTR *pVal );`	Returns a string describing the status of the current operation (for example propagation).
`HRESULT CurrentFIPSMode( BSTR username, BSTR password, [out, retval] BOOL* pVal );`	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.RetrievedProperty

RetrievedProperty Methods	Description
`HRESULT SetAppName ( BSTR appName );`	Sets the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.
`HRESULT SetDSN ( BSTR dsn );`	Sets the name of the ODBC DSN to use when connecting to the database. Default is 'BES_bfenterprise'.
`HRESULT Create ( BSTR name, BSTR relevance, UINT delaySeconds, BSTR dbUserName, BSTR password );`	Creates a new retrieved property with the specified name, relevance, and evaluation period. Use the username and password to propagate a new version of the action site. The ID property is set to the database ID of the newly created retrieved property. Note that this function fails if a retrieved property with the same name already exists.
`HRESULT Modify ( long ID, BSTR name, BSTR relevance, BSTR evaluation period, BSTR dbUserName, BSTR password );`	Modifies the existing retrieved property with the specified database ID to have the specified name, relevance and evaluation period. Use the username and password to propagate a new version of the action site. Note that this function fails if the name is changed and a retrieved property with the same name already exists.
`HRESULT Delete ( long ID, BSTR dbUserName, BSTR password );`	Deletes the existing retrieved property with the specified database ID, and uses the username and password to propagate a new version of the action site.
`HRESULT Load ( long ID, BSTR dbUserName, BSTR password );`	Connects to the database with the specified username and password, and sets the ID, name, and relevance properties with the values stored in the database for the existing retrieved property with the specified database ID.

RetrievedProperty Properties	Description

`HRESULT ID (`
`     [out, retval] long *pVal`
`);`
	The database ID number of the retrieved property. Not set until after either Load, Modify, Delete, or Create has succeeded.

`HRESULT Name (`
`     [out, retval] BSTR *pVal`
`);`
	The name of the retrieved property. Not set until after Load, Modify or Create has succeeded.

`HRESULT Relevance (`
`     [out, retval] BSTR *pVal`
`);`
	The relevance expression that defines the retrieved property. Not set until after Load, Modify, or Create has succeeded.

`HRESULT DelaySeconds (`
`     [out, retval] UINT *pVal`
`);`
	The evaluation period of this retrieved property in seconds (0 means 'every report'). Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT IsCustom (`
`     [out, retval] BOOL *pVal`
`);`
	True if the property is a custom property. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT IsReserved (`
`     [out, retval] BOOL *pVal`
`);`
	True if the property is a reserved property. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT IsDefault (`
`     [out, retval] BOOL *pVal`
`);`
	True if the property is a predefined property. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT IsPropertyOverride (`
`     [out, retval] BOOL *pVal`
`);`
	True if the property is a reference to a property defined in an analysis. The evaluation period of this object overrides the evaluation period specified in the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourceSiteID (`
`     [out, retval] UINT *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the SiteID of the site which contains the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourceFixletID (`
`     [out, retval] UINT *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the FixletID of the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourcePropertyID (`
`     [out, retval] UINT *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the ID of the property withing that analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourceName (`
`     [out, retval] BSTR *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the original name of the property in the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourceRelevance (`
`     [out, retval] BSTR *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the original relevance of the property in the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT SourceDelaySeconds (`
`      [out, retval] BSTR *pVal`
`);`
	If the property is a reference to a property defined in an analysis, then this is the original evaluation period for the property in the analysis. Not set until after Load, Modify, Delete, or Create has succeeded.

`HRESULT DiagnosticMessage (`
`      [out, retval] BSTR *pVal`
`);`
	If a method fails, this property returns a string containing a diagnostic message.

`HRESULT StatusMessage (`
`      [out, retval] BSTR *pVal`
`);`
	Returns a string describing the status of the current operation (such as propagation).

`HRESULT CurrentFIPSMode(`
`    BSTR username,`
`    BSTR password,`
`    [out, retval] BOOL* pVal );`
	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.FixletMessage

FixletMessage Methods	Description

`HRESULT SetAppName (`
`      BSTR appName`
`);`
	Sets the Application Name string that you use to specify the registry key for the BESAPI. Default is 'BESAPI'.

`HRESULT SetDSN (`
`      BSTR dsn`
`);`
	Sets the name of the ODBC DSN to use when you connect to the database. Default is 'BES_bfenterprise'.

`HRESULT Load (`
`      long siteID,`
`      long FixletID,`
`      BSTR dbUserName,`
`      BSTR password`
`);`
	Connects to the database with the specified username and password. Queries the database for the data of the Fixlet with the specified FixletID contained in the Fixlet site that is identified by the specified siteID. The properties of the FixletMessage object are set with the data read from the database.

`HRESULT Delete (`
`      BSTR dbUserName,`
`      BSTR password`
`);`
	Connects to the database with the specified username and password. Deletes this Fixlet, analysis, task, or computer group, and propagates the action site. This method fails if the Load method is not used to load the object.

`HRESULT Activate (`
`      BSTR dbUserName,`
`      BSTR password`
`);`
	Connects to the database with the specified username and password. Activates this analysis and propagates the action site. This method fails if the Load method is not used to load the object, or if the object is not an analysis.

`HRESULT Deactivate (`
`      BSTR dbUserName,`
`      BSTR password`
`);`
	Connects to the database with the specified username and password. Stops this analysis and propagates the action site. This method fails if the Load method is not used to load the object, or if the object is not an analysis.

FixletMessage Properties	Description

`HRESULT SiteID (`
`    [out, retval] long *pVal`
`);`
	The database ID number of the site that contains the loaded Fixlet. Not set until after Load succeeds.

`HRESULT FixletID (`
`    [out, retval] long *pVal`
`);`
	The database ID number of the loaded Fixlet. Not set until after Load succeeds.

`HRESULT SiteName (`
`    [out, retval] long *pVal`
`);`
	The name of the site that contains the loaded Fixlet. Not set until after Load succeeds.

`HRESULT SiteDisplayName(`
`   [out, retval] BSTR *pVal`
`);`
	Version 7.1 introduced a new feature where content can rename a site, such that the site name in all the console and Web Reports UI is shown with the new name. To maintain compatibility with an earlier version, the SiteName property still returns the old site name, while this method returns the new site display name.

`HRESULT SiteURL (`
`    [out, retval] long *pVal`
`);`
	The gather URL of the site that contains the loaded Fixlet. Not set until after Load succeeds.

`HRESULT Name (`
`    [out, retval] BSTR *pVal`
`);`
	The name of the loaded Fixlet. Not set until after Load succeeds.

`HRESULT Relevance (`
`    [out, retval] BSTR *pVal`
`);`
	The relevance expression, including parent relevance that defines when the problem identified by this Fixlet affects a client. Not set until after Load succeeds.

`HRESULT ActionScript (`
`    long whichAction,`
`    [out, retval] BSTR *pVal`
`);`
	The action script which corrects the problem that is identified by this Fixlet. Because the Fixlet might provide more than one corrective action, you must specify the whichAction parameter. Usually the whichAction parameter is 1. Not set until after Load succeeds.

`HRESULT ActionScriptMIMEType (`
`    long whichAction,`
`    [out, retval] BSTR *pVal`
`);`
	The MIME type of the action script which corrects the problem that is identified by this Fixlet. Because the Fixlet might provide more than one corrective action, you must specify the whichAction parameter. Usually the whichAction parameter is 1. Not set until after Load succeeds.

`HRESULT ActionScriptTypeName (`
`    long whichAction,`
`    [out, retval] BSTR *pVal`
`);`
	The friendly name of the MIME type of the action script which corrects the problem that is identified by this Fixlet. Because the Fixlet might provide more than one corrective action, you must specify the whichAction parameter. Usually the whichAction parameter is 1. Not set until after Load succeeds.

`HRESULT HTML (`
`    [out, retval] BSTR *pVal`
`);`
	The HTML code for the body of the Fixlet which describes the problem and offers links to the corrective actions. This HTML code might reference image files from the site. Those references are relative, and can be resolved if you set the BASE property of the HTML document to the path to the site's data directory. Not set until after Load succeeds.

`HRESULT IsTask (`
`  [out, retval] BOOL *pVal`
`);`
	True if the Fixlet is a task message. Not set until after Load succeeds.

`HRESULT IsAnalysis (`
`    [out, retval] BOOL *pVal`
`);`
	True if the Fixlet is an analysis. Not set until after Load succeeds.

`HRESULT IsPlainFixlet (`
`    [out, retval] BOOL *pVal`
`);`
	True if the Fixlet is neither a task, analysis, baseline nor computer group. Not set until after Load succeeds.

`HRESULT IsBaseline (`
`    [out, retval] BOOL *pVal`
`);`
	True if the Fixlet is a baseline. Not set until after Load succeeds.

`HRESULT DownloadSize (`
`    [out, retval] UINT *pVal`
`);`
	The size of the download for this Fixlet, in bytes. Not set until after Load succeeds.

`HRESULT Source (`
`    [out, retval] BSTR *pVal`
`);`
	A string that describes the source of this Fixlet (for example, Microsoft). Not set until after Load succeeds.

`HRESULT SourceID (`
`    [out, retval] BSTR *pVal`
`);`
	A string that describes an identifier that is specified by the source of the Fixlet (for example Microsoft KB number). Not set until after Load succeeds.

`HRESULT SourceSeverity (`
`    [out, retval] BSTR *pVal`
`);`
	A string that describes the severity rating of the Fixlet as determined by the source. Not set until after Load succeeds.

`HRESULT SourceReleaseDate (`
`    [out, retval] BSTR *pVal`
`);`
	A string indicating when the source released the information. Not set until after Load succeeds.

`HRESULT Category (`
`    [out, retval] BSTR *pVal`
`);`
	A string that describes the category of this Fixlet as specified by the Fixlet author. Not set until after Load succeeds.

`HRESULT Message (`
`    [out, retval] BSTR *pVal`
`);`
	A string that contains the text message that is specified when this Fixlet was created (applies only to custom Fixlets). Not set until after Load succeeds.

`HRESULT IsDeleted (`
`    [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is deleted (applies only to custom Fixlets). Not set until after Load succeeds.

`HRESULT PropertyIDSet (`
`    [out, retval] BSTR *pVal`
`);`
	A string that contains the list of database IDs of the properties that refer to the properties contained in this analysis. The IDs are separated by tabs. This property is set only for analyses. Not set until after Load succeeds.

`HRESULT IsGloballyActivated (`
`     [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is an analysis and is activated by a Master Operator. Not set until after Load succeeds.

`HRESULT IsGloballyActivated (`
`     [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is an analysis and is activated by a Master Operator. Not set until after Load succeeds.

`HRESULT IsLocallyActivated (`
`     [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is an analysis and is activated by a Non-Master Operator. Not set until after Load succeeds.

`HRESULT CanActivate (`
`     [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is an analysis and can be activated by the current user (the user which was used to Load this Fixlet). Not set until after Load succeeds.

`HRESULT CanDeactivate (`
`     [out, retval] BOOL *pVal`
`);`
	True if this Fixlet is an analysis and can be stopped by the current user (the user which was used to Load this Fixlet). Not set until after Load succeeds.

`HRESULT ActionXML (`
`  long whichAction,`
`  SAFEARRAY(byte) settingsXML,`
`  [out, retval] IXMLDOMDocument`
`    **actionXML`
`);`
	Returns an importable XML document that represents an action that is taken from this Fixlet, task or baseline. Specify which action with the whichAction parameter. The settingsXML parameter is the bytes of an XML document that specifies the various settings in an action.

`HRESULT XML (`
`  [out, retval] IXMLDOMDocument`
`    **xml`
`);`
	Returns an importable XML document that represents this Fixlet object.

`HRESULT DiagnosticMessage (`
`     [out, retval] BSTR *pVal`
`);`
	If a method fails, this property returns a string that contains a diagnostic message.

`HRESULT StatusMessage (`
`     [out, retval] BSTR *pVal`
`);`
	Returns a string that describes the status of the current operation (for example propagation).

`HRESULT CurrentFIPSMode(`
`    BSTR username,`
`    BSTR password,`
`    [out, retval] BOOL* pVal );`
	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.SiteManager

SiteManager Methods	Description

`HRESULT SetAppName (`
`     BSTR appName`
`);`
	Set the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.

`HRESULT SetDSN (`
`     BSTR dsn`
`);`
	Set the name of the ODBC DSN to use when connecting to the database. Default is 'BES_bfenterprise'.

HRESULT Subscribe (BSTR mastheadFilePath, BSTR username, BSTR password);	Connects to the database with the username and password. Subscribes to the site specified by the masthead file at the specified file path. Propagates a new version of the action site.
HRESULT SubscribeWithRelevance (BSTR mastheadFilePath, BSTR relevance, BSTR username, BSTR password);	Connects to the database with the username and password. Subscribes computers that evaluate the specified relevance as "true" to the site specified by the masthead file at the specified file path. Propagates a new version of the action site.
HRESULT Unsubscribe (UINT siteID, BSTR username, BSTR password);	Connects to the database with the username and password. Unsubscribes from the site specified by the specified site ID. Propagates a new version of the action site.

SiteManager Properties	Description
HRESULT DiagnosticMessage ([out, retval] BSTR *pVal);	If a method fails, this property returns a string containing a diagnostic message.
HRESULT StatusMessage ([out, retval] BSTR *pVal);	Returns a string describing the status of the current operation (for example propagation).
HRESULT CurrentFIPSMode(BSTR username, BSTR password, [out, retval] BOOL* pVal);	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.ActionSiteMasthead

ActionSiteMasthead Methods	Description
HRESULT SetAppName (BSTR appName);	Sets the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.
HRESULT SetDSN (BSTR dsn);	Sets the name of the ODBC DSN to use when connecting to the database. Default is 'BES_bfenterprise'.
HRESULT Load (BSTR username, BSTR password);	Connects to the database with the specified username and password. Reads the action site masthead from the database.

ActionSiteMasthead Properties	Description
HRESULT DiagnosticMessage ([out, retval] BSTR *pVal);	If a method fails, this property returns a string containing a diagnostic message.
HRESULT StatusMessage ([out, retval] BSTR *pVal);	Returns a string describing the status of the current operation (for example, propagation).

`HRESULT IsAdministrator(`
`    BSTR username,`
`    [out, retval] BOOL *pVal`
`);`
	True if the operator specified is a Master Operator.

`HRESULT IsAuthoringEnabled(`
`    [out, retval] BOOL *pVal`
`);`
	True if the authoring features are enabled for this deployment.

`HRESULT CurrentFIPSMode(`
`    BSTR username,`
`    BSTR password,`
`    [out, retval] BOOL* pVal );`
	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BESAPI.SigningKeys

SigningKeys Methods	Description

`HRESULT SetAppName (`
`    BSTR appName`
`);`
	Sets the Application Name string that is used to specify the registry key for the BESAPI. Default is 'BESAPI'.

`HRESULT SetDSN (`
`    BSTR dsn`
`);`
	Sets the name of the ODBC DSN to use when you connect to the database. Default is 'BES_bfenterprise'.

`HRESULT SetPrivateKeyPath (`
`    BSTR username,`
`    BSTR fullpath`
`);`
	Registers the location of the private key file for the specified user. The fullpath parameter is the full path to the private key file, including the file name that is usually named "publisher.pvk".

`HRESULT SetPublisherCertPath (`
`    BSTR username,`
`    BSTR fullpath`
`);`
	Registers the location of the user's certificate file that corresponds to the private key. The fullpath parameter is the full path to the certificate file, including the file name that is usually named "publisher.crt".

`HRESULT SetLicenseCertPath (`
`    BSTR username,`
`    BSTR fullpath`
`);`
	Registers the location of the site license certificate file for the deployment. The fullpath parameter is the full path to the certificate file, including the file name that is usually named "license.crt".

`HRESULT SetDefaultAppName (`
`  BSTR appName`
`);`
	Sets the default application name string which is used by every BESAPI object, unless its SetAppName method is used to override the default. If no default is specified using this method and SetAppName is not called explicitly on an object, then the object uses 'BESAPI' as the application name.

`HRESULT SetDefaultDSN (`
`  BSTR dsnName`
`);`
	Sets the default name of the ODBC DSN to use when you connect to the database. This DSN is used unless the SetDSN method is used to override the default. If no default is specified with this method, and SetDSN is not called explicitly on an object, then the object uses 'BES_bfenterprise' as the name of the ODBC DSN when you connect to the database.

SigningKeys Properties	Description
`HRESULT DiagnosticMessage ( [out, retval] BSTR *pVal );`	If a method fails, this property returns a string that contains a diagnostic message.
`HRESULT StatusMessage ( [out, retval] BSTR *pVal );`	Returns a string that describes the status of the current operation (for example, propagation).
`HRESULT AreSigningKeysValid( BSTR username, BSTR password, [out, retval] BOOL *pVal );`	True if the keys registered for the specified user are valid, and the specified password is correct. If this is false, then the DiagnosticMessage property specifies the reason that the keys are not valid.
`HRESULT CurrentFIPSMode( BSTR username, BSTR password, [out, retval] BOOL* pVal );`	Retrieves the masthead from the database and returns true if FIPS mode cryptography is enabled in the masthead.

BES schemas

The schemas that are listed in this section are condensed and augmented interpretations of traditional XML schemas. They outline the structure in terms of elements and attributes, simplifying the XML syntax. They also add a repeat range that describes the number of elements and attributes that are expected by Tivoli Endpoint Manager. Although these augmented schemas are not schemas (.xsd) in the strictest sense, for the sake of brevity we describe them as such.

There are two basic types in the augmented schemas, the elements and their attributes:

- *<ElementTagName> ElementType </ElementTagName>*

  This identifies an XML element named ElementTagName of type ElementType.
- *<ElementTagName*

  AttributeName="AttributeType">

  ...

  </ElementTagName>

  This XML element has an attribute named AttributeName of type AttributeType.

There might be multiple entries for these objects. For each element and attribute, there is a stated range. That represents the number of objects of the specified type that is expected. These values are represented by numbers in square brackets, as follows:

- *[x]* This element or attribute must occur exactly x times.
- *[x..y]* This element or attribute occurs a minimum of x times and a maximum of y times.
- *[x..*]* This element or attribute occurs a minimum of x times and has no maximum.

The following example indicates that the schema allows zero or more Relevance elements of type RelevanceString:

```
<Relevance> RelevanceString </Relevance> [0..*]
```

Fixlet or task

The schema for Fixlets is similar to the schema for tasks. Their combined schema is as follows:

```
<Fixlet> | <Task>
      <Title> xs:normalizedString </Title> [1]
            This is the Fixlet name.
      <Description> xs:string </Description> [1]
            The description is treated as HTML that is used to construct
            the Fixlet body
            for the "Description" tab of the console Fixlet document.
      <Relevance> RelevanceString </Relevance> [0..*]
            Each relevance element is shown as a separate relevance clause
            in the console Fixlet document. The Fixlet is reported as
            relevant only for computers for which every relevance clauses
            evaluates to true.
      <GroupRelevance JoinByIntersection="xs:boolean"> [1]
            The content of this tag includes any number of the search component
            types in any order:
      <SearchComponentRelevance> SearchComponentRelevance
      </SearchComponentRelevance> [0..*]
      <SearchComponentPropertyReference> SearchComponentPropertyReference
      </SearchComponentPropertyReference> [0..*]
      <SearchComponentGroupReference> SearchComponentGroupReference
      </SearchComponentGroupReference>  [0..*]
      </GroupRelevance>
            As of version 7.0, it became possible for Fixlets, Tasks,
            Baselines, and Analyses to have Relevance definitions in
            the same form as computer group definitions. You can
            access this functionality in the console by going to
            the Relevance tab of the Create Fixlet Dialog and selecting
            the option for "computers matching the following criteria".
            The XML for such a Fixlet would have a GroupRelevance tag
            instead of a Relevance tag, in the same form as the XML
            for computer groups.
      <Category> xs:normalizedString </Category> [0..1]
            Displayed on the "details" tab of the Fixlet document
            and in the Fixlet tree/list.
      <WizardData> ... </WizardData> [0..1]
            For use by wizards.  Not used for importing through
            the console or through the Server API.
      <DownloadSize> xs:nonNegativeInteger </DownloadSize> [0..1]
            The total number of bytes of all downloads in the
            Fixlet's action.
      <Source> xs:normalizedString </Source> [0..1]
      <SourceID> xs:normalizedString </SourceID> [0..1]
      <SourceReleaseDate> NonNegativeDate </SourceReleaseDate> [0..1]
            Must be of the form YYYY-MM-DD.
      <SourceSeverity> xs:normalizedString </SourceSeverity> [0..1]
      <CVENames> xs:normalizedString </CVENames> [0..1]
      <SANSID> xs:normalizedString </SANSID> [0..1]
            DownloadSize, Source, SourceID, SourceReleaseDate,
            SourceSeverity, CVENames, and SANSID are extra
            information about a Fixlet that are displayed
            on the "details" tab of the Fixlet document and
            in the Fixlet tree/list.
      <DefaultAction> FixletAction </DefaultAction> [0..1]
      <Action> FixletAction </Action> [0..*]
      <MIMEField> [0..*]
       <Name> xs:string </Name> [1]
       <Value> xs:string </Value> [1]
       </MIMEField>
            Some external Fixlets are tagged with special pieces
            of data in the form of MIME fields. They are then
            used by dashboards and wizards. These fields are
            preserved when Fixlets are exported and imported.
            You should only add these special fields if you want
            to access them from dashboards or wizards.
</Fixlet> | </Task>
```

Baseline

You can also specify baselines with .bes files. This is the schema:

```
<Baseline>
      Baselines have the following fields in common with Fixlets and tasks:
      <Title> xs:normalizedString </Title> [1]
      <Description> xs:string </Description> [1]
      <Relevance> RelevanceString </Relevance> [0..*]
      <GroupRelevance> ... </GroupRelevance> [0..1]
      <Category> xs:normalizedString </Category> [0..1]
      <WizardData> ... </WizardData> [0..1]
      <DownloadSize> xs:nonNegativeInteger </DownloadSize> [0..1]
      <Source> xs:normalizedString </Source> [0..1]
      <SourceID> xs:normalizedString </SourceID> [0..1]
      <SourceReleaseDate> NonNegativeDate </SourceReleaseDate> [0..1]
      <SourceSeverity> xs:normalizedString </SourceSeverity> [0..1]
      <CVENames> xs:normalizedString </CVENames> [0..1]
      <SANSID> xs:normalizedString </SANSID> [0..1]
      <MIMEField> ... </MIMEField> [0..*]

      Baselines are also composed of a collection of named baseline component
      groups, which contain baseline components:
      <BaselineComponentCollection> [1]
            <BaselineComponentGroup
                  Name="xs:normalizedString [0..1]"> [0..*]
                  <BaselineComponent
                        Name="xs:normalizedString [0..1]"
                              The "Name" attribute corresponds to the content ID
                              of the action, which is a short identifier for the
                              component that is used to match it up with a
                              particular action of the source Fixlet/task/baseline
                              of the component.
                        ActionName="xs:normalizedString [0..1]"
                              The "ActionName" attribute is a longer description
                              of the action shown on the "Components" tab of the
                              baseline document.
                        IncludeInRelevance="xs:boolean [0..1]"
                              If true, then the relevance of the component will be
                              included in the relevance for the baseline. If false,
                              then the baseline will be relevant regardless of
                              whether this component is relevant; the individual
                              component will still not be executed if it is not
                              relevant.
                        SourceSiteURL="xs:anyURI [0..1]"
                              The gather site URL for the source Fixlet/task/
                              baseline that this component comes from.
                        SourceID="xs:nonNegativeInteger [0..1]"
                              The ID of the Fixlet/task/baseline that this
                              component comes from.
                        > [0..*]
                        <Relevance> RelevanceString </Relevance> [1]
                              This component will be executed only on computers
                              for which the relevance clause evaluates to true.
                        <ActionScript> ActionScript </ActionScript> [1]
                        <SuccessCriteria> ActionSuccessCriteria
                        </SuccessCriteria> [0..1]
                  </BaselineComponent>
            </BaselineComponentGroup>
      </BaselineComponentCollection>
      <Settings> ActionSettings </Settings> [0..1]
      <SettingsLocks> ActionSettingsLocks </SettingsLocks> [0..1]
</Baseline>
```

Single actions

You can specify Actions in a .bes file. This is the schema:

```
<SingleAction>
      <Title> xs:normalizedString </Title> [1]
            The name of the action.
      <Relevance> RelevanceString </Relevance> [1]
            The action will run only on computers for which the relevance
            expression evaluates to true.
      <ActionScript> ActionScript </ActionScript> [1]
      <SuccessCriteria> ActionSuccessCriteria </SuccessCriteria> [0..1]
      <Settings> ActionSettings </Settings> [0..1]
      <SettingsLocks> ActionSettingsLocks </SettingsLocks> [0..1]
      <SuccessCriteriaLocked> xs:boolean </SuccessCriteriaLocked> [0..1]
            If this element is present and set to true, then the action will have
            a success criteria that the user taking the action will not be able
            to change in the Take Action Dialog.
      <IsUrgent> xs:boolean </IsUrgent>
            This marks the action as urgent for client processing. It should
            be left out in most circumstances (defaults to false).
</SingleAction>
```

Multiple action groups

You can specify multiple action groups in a .bes file. This is the schema:

```
<MultipleActionGroup>
      <Title> xs:normalizedString </Title> [1]
            The name of the multiple action group.
      <PreGroupActionScript> ActionScript </PreGroupActionScript> [0..1]
            An action to run before all the member actions of the multiple
            action group; corresponds to the Pre-Execution Action Script
            tab of the Take Action Dialog when taking multiple actions.
      <MemberAction> [1..*]
            <Title> xs:normalizedString </Title> [1]
                  The name of the member action.
            <Relevance> RelevanceString </Relevance> [1]
                  The member action will run only on computers for which
                  the relevance expression evaluates to true.
            <ActionScript> ActionScript </ActionScript> [1]
            <SuccessCriteria> ActionSuccessCriteria </SuccessCriteria> [0..1]
            <IncludeInGroupRelevance>true|false</IncludeInGroupRelevance>
                  A value of true for IncludeInGroupRelevance for a member
                  action of a multiple action group means that the group
                  as a whole is relevant if this member action's
                  relevance is true. Thus, if any of a group's members
                  that have IncludeInGroupRelevance set to true are relevant,
                  then the group as a whole is relevant. If no members
                  have IncludeInGroupRelevance set, then the group should be
                  relevant on all computers, as long as the group's top-level
                  relevance is true.
      </MemberAction>
      <PostGroupActionScript> ActionScript </PostGroupActionScript> [0..1]
            An action to run once all member actions have finished executing;
            corresponds to the Post-Execution Action Script when taking
            multiple actions.
      <Settings> ActionSettings </Settings> [0..1]
      <SettingsLocks> ActionSettingsLocks </SettingsLocks> [0..1]
</MultipleActionGroup>
```

Analyses

You can specify analyses in a .bes file. This is the schema:

```
<Analysis>
      <Title> xs:normalizedString </Title> [1]
            The name of the analysis.
      <Description> xs:string </Description> [1]
            The description is treated as HTML that is shown on the "Description"
            tab of the analysis document.
      <Relevance> RelevanceString </Relevance> [1..*]
            Only computers for which all relevance clauses are true will report
            results.
      <Property
            Name="xs:normalizedString [1]"
                  The name of the property.
            EvaluationPeriod="NonNegativeTimeInterval [0..1]"
                  Controls how often the property is evaluated.
            ID="xs:nonNegativeInteger [1]"
                  Each property in the analysis must have a unique ID attribute.
            KeepStatistics="xs:boolean [0..1]"
                  If true, then enables statistical inspection of the results
                  for this property. This statistical data is then available
                  to dashboards and wizards. You should only capture these
                  properties if you want to use statistical data from
                  dashboards or wizards.
            > [0..*]
            RelevanceString
      </Property>
      <MIMEField> ... </MIMEField> [0..*]
      <GroupRelevance> ... </GroupRelevance> [0..1]
</Analysis>
```

Computer groups

You can specify automatic computer groups in the .bes file. Note that only automatic groups can be created by importing through the console, and no computer groups can be created by using the Server API. This is the schema:

```
<ComputerGroup>
      <Title> xs:normalizedString </Title> [1]
            The name of the computer group.
      <JoinByIntersection> xs:boolean </JoinByIntersection> [1]
            If true, then a computer is in the group only if it meets
            the requirements of all of the group components.  If false, a
            computer is in the group if it meets any of the requirements
            of the group components.
      <IsDynamic> xs:boolean </IsDynamic> [1]
            Must be true.  For internal use.
      <EvaluateOnClient> xs:boolean </EvaluateOnClient> [1]
            Must be true.  For internal use.
            The rest of the computer group definition includes any number
            of the search component types in any order:
      <SearchComponentRelevance> SearchComponentRelevance
      </SearchComponentRelevance> [0..*]
      <SearchComponentPropertyReference> SearchComponentPropertyReference
      </SearchComponentPropertyReference> [0..*]
      <SearchComponentGroupReference> SearchComponentGroupReference
      </SearchComponentGroupReference> [0..*]
</ComputerGroup>
```

Properties

This element creates a global retrieved property that you can specify in a .bes file. This is the schema:

```
<Property
      Name="xs:normalizedString [1]"
      EvaluationPeriod="NonNegativeTimeInterval [0..1]">
            controls how often the property is evaluated.
      RelevanceString
</Property>
```

Shared BES XML elements

The following are API elements that can be mixed with other elements. For example, you can have several actions that are associated with a Fixlet, all sharing the same schema.

FixletAction

You can specify multiple actions for each Fixlet you define in a .bes file. This is the schema:

```
<...
ID="xs:normalizedString [1]">
      Each action inside a Fixlet or task must have a unique ID, which is displayed
      on the actions tab of the Edit Fixlet Dialog and on the Details tab of the
      Fixlet document.
      <Description> [0..1]
            <PreLink> xs:normalizedString </PreLink> [1]
            <Link> xs:normalizedString </Link> [1]
            <PostLink> xs:normalizedString </PostLink> [1]
      </Description>
      The description of the action is the HTML that is displayed in the actions
      section of the Fixlet description tab.  The content of the Prelink tag is
      the HTML that is displayed before the link that takes the action
      (for instance "Click").  The content of the Link tag is HTML that the user
      can click on to take the action (for instance "here").
      The content of the PostLink tag is HTML following the link (for instance
      "to deploy this action.")
      <ActionScript> ActionScript </ActionScript> [1]
            See the "ActionScript" type.
      <SuccessCriteria> ActionSuccessCriteria </SuccessCriteria> [0..1]
            See the "ActionSuccessCriteria" type.  If this element is not preset
            and the action is inside a Fixlet, the success criteria will default
            to match the relevance of the Fixlet.  If the action is inside a task,
            the success criteria will default to run to completion.
      <SuccessCriteriaLocked> xs:boolean </SuccessCriteriaLocked> [0..1]
            If this element is present and set to true, then the action will
            have a success criteria that the user taking the action will not
            be able to change in the Take Action Dialog.
      <Settings> ActionSettings </Settings> [0..1]
      <SettingsLocks> ActionSettingsLocks </SettingsLocks> [0..1]
</...>
```

ActionScript

You can include multiple scripts in a .bes file. This is the schema:

```
<...
MIMEType="xs:normalizedString [0..1]">
      The attribute MIMEType specifies the type of the actionscript; if absent,
      it defaults to "application/x-Fixlet-Windows-Shell".  For AppleScript, use
      "application/x-AppleScript".  For a sh script, use "application/x-sh".
      The contents of the tag specify the contents of the action script.
      xs:string
</...>
```

ActionSuccessCriteria

You can specify an action success criteria in a .bes file. The ActionSuccessCriteria element corresponds to the 'Success Criteria' tab of the Take Action Dialog. This is the schema:

```
<...
Option="xs:string (possible values: {'RunToCompletion'|'OriginalRelevance'|
'CustomRelevance'}) [0..1]">
      If the 'Option' attribute is 'RunToCompletetion', the action will
      be considered successful when all lines of the action script have
      been executed.  If the option is 'OriginalRelevance', the action
      is considered successful when the applicability relevance of
      the action becomes false.  If the option is 'CustomRelevance',
      then the action is considered false when the custom relevance
      expression inside the tag evaluates to false.

      RelevanceString
</...>
```

ActionSettings

The contents of the ActionSettings element correspond to the options available on the *Execution, Users, Messages, Offer,* and *Post-Action* tabs of the Take Action Dialog. This is the schema:

```
<...>
      <ActionUITitle> xs:normalizedString </ ActionUITitle > [0..1]
      The title of the message action as displayed in the client UI.
      <PreActionShowUI> xs:boolean </PreActionShowUI> [0..1]
      If true, a message is displayed before running the action.
      <PreAction> [0..1]
            <Text> xs:string </Text> [0..1]
              The text of the message shown before running the action.
            <AskToSaveWork> xs:boolean </AskToSaveWork> [0..1]
              If true, the user is asked to save work before
                 the action is run.
            <ShowActionButton> xs:boolean </ShowActionButton> [0..1]
              If true, the user is allowed to view the action script
                 before running it.
            <ShowCancelButton> xs:booelan </ShowCancelButton> [0..1]
                  If true, the user is allowed to cancel running
                  the action.
            <DeadlineBehavior> xs:string (possible values {ForceToRun|
                               RunAutomatically}
            </DeadlineBehavior>            [0..1]
                  All pre-action messages have a deadline. If the
                  deadline behavior is 'ForceToRun', the user will
                  be presented with a dialog that must be acknowledged
                  when the deadline is reached. If the deadline behavior
                  is 'RunAutomatically', the action will run when the
                  deadline is reached, regardless of whether the user has
                  acknowledged the action.
            <DeadlineType> xs:string (possible values {Interval|Aboslute})
            </DeadlineType> [0..1]
            <DeadlineInterval> ActionMessageTimeInterval
            </DeadlineInterval> [0..1]
            <DeadlineOffset> TimeInteval </DeadlineOffset> [0..1]
                  The pre-action deadline can be specified as either a
                  time interval from when the client UI is shown, or as
                  an absolute date and time. If the deadline type is
                  'Interval', then the 'DeadlineInterval' element must
                  be present with an appropriate time interval of type
                  ActionMessageTimeInterval. If the deadline type is
                  'Absolute', then the 'DeadlineOffset' element must
                  be present with a TimeInterval, which will create
                  an absolute deadline for the action that is offset
                  from the date and time the action is taken in the console.
            <ShowConfirmation> xs:boolean </ShowConfirmation> [0..1]
            <Confirmation> xs:string </Confirmation> [0..1]
                  If 'ShowConfirmation' is true, an extra confirmation message
                  is shown to the user with text from the
                  'Confirmation' element.
      </PreAction>
      <HasRunningMessage> xs:boolean </HasRunningMessage> [0..1]
            If true, a message is displayed while running the action.
      <RunningMessage> [0..1]
            <Title> xs:normalizedString </Title> [0..1]
                  The title of message displayed while running the action.
            <Text> xs:string </Text> [0..1]
                  The text of the message displayed while running the action.
      </RunningMessage>
      <HasTimeRange> xs:boolean </HasTimeRange> [0..1]
      <TimeRange> [0..1]
            <StartTime> xs:time </StartTime> [0..1]
            <EndTime> xs:time </EndTime> [0..1]
      </TimeRange>
            If HasTimeRange is true, then the action will run only between
            the StartTime and EndTime in client local time.
            Times have the form hh:mm:ss.
      <HasStartTime> xs:boolean </HasStartTime> [0..1]
      <StartDateTimeOffset> TimeInterval </StartDateTimeOffset> [0..1]
            If HasStartTime is true, then the action will start at a date
            and time computed by adding the StartDateTimeOffset to
            the time the action is taken.  For example, to have an
            action start one day from the time it is taken, specify
            "P1D".  Note that this time can be negative – to create
            an action that starts a day before the action is taken
            (so that clients in every timezone will start executing
            immediately), specify "-P1D". See TimeInterval for possible values.
      <HasEndTime> xs:boolean </HasEndTime> [0..1]
      <EndDateTimeOffset> NonNegativeTimeInterval </EndDateTimeOffset> [0..1]
            If HasEndTime is true, then the action will start at a date and
            time computed by adding the EndDateTimeOffset to the time the
            action is taken. See NonNegativeTimeInterval for possible values.
      <HasDayOfWeekConstraint> xs:boolean </HasDayOfWeekConstraint> [0..1]
      <DayOfWeekConstraint> [0..1]
            <Sun> xs:boolean </Sun> [0..1]
            <Mon> xs:boolean </Mon> [0..1]
            <Tue> xs:boolean </Tue> [0..1]
            <Wed> xs:boolean </Wed> [0..1]
            <Thu> xs:boolean </Thu> [0..1]
            <Fri> xs:boolean </Fri> [0..1]
            <Sat> xs:boolean </Sat> [0..1]
      </DayOfWeekConstraint>
            If HasDayOfWeekConstraint is true, then the action will run only
            on those days of the week that are specified and whose contents
            are true.
      <ActiveUserRequirement> xs:string (value comes from list: {'NoRequirement'|
                              'RequireUser'|'RequireNoUser'})
      </ActiveUserRequirement> [0..1]
            NoRequirement = Run independently of user presence
            RequireUser = Run when at least one of the selected users is logged on
            RequireNoUser = Run only when no user is logged on
      <ActiveUserType> xs:string (value vomes from list: {'AllUsers'|'LocalUsers'|
                      'UsersOfGroups'})
      </ActiveUserType> [0..1]
```

```
<UIGroupConstraints>
      <Win9xGroup /> [0..1]
      <WinNTGroup /> [0..1]
      <LocalGroup Name="xs:string" /> [0..*]
      <DomainGroup Name="xs:string" Sid="xs:string" /> [0..*]
</UIGroupConstraints>
      If the ActiveUserType is 'UsersOfGroups', then the client UI will
      only be shown to a user if a user is in at least one of
      the specified groups.
<HasWhose> xs:boolean </HasWhose> [0..1]
<Whose> [0..1]
      <Property> xs:string </Property> [0..1]
      <Relation> xs:string </Relation> [0..1]
      <Value> xs:string </Value> [0..1]
</Whose>
      If HasWhose is true, then the action will run only on computers
      where the retrieved property named in Property has the relationship
      given in Relation to the value in Value. For example, to add
      the constraint that the action runs only on computers where the
      value of the retrieved property OS starts with Win:
<Property>OS</Property>
<Relation>starts with</Relation>
<Value>Win</Value>
      The possible values of relation are {matches, does not match,
      contains, does not contain, starts with, ends
      with, =, <, >, <=, >=, !=}
<Reapply> xs:boolean </Reapply> [0..1]
      If true, the action will automatically reapply if it becomes
      relevant again after it has successfully executed.
<HasReapplyLimit> xs:boolean </HasReapplyLimit> [0..1]
<ReapplyLimit> xs:nonNegativeInteger </ReapplyLimit> [0..1]
      If the action is set to reapply and HasReapplyLimit is true, then
      the action will only reapply the specified number of times.
<HasReapplyInterval> xs:boolean </HasReapplyInterval> [0..1]
<ReapplyInterval> NonNegativeTimeInterval (possible values:
      {'PT10M'|'PT15M'|'PT30M'|'PT1H'|'PT2H'|'PT4H'|'PT8H'|
      'PT12H'|'P1D'|'P2D'|'P5D'|'P7D'|'P15D'|'P30D'})
</ReapplyInterval> [0..1]
      If the action is set to reapply and HasReapplyInterval is true,
      then the client will wait the specified time interval between
      reapplications.  The possible values are in the list above.
      See TimeInterval for information about the value format.
<HasRetry> xs:boolean </HasRetry> [0..1]
<RetryCount> xs:nonNegativeInteger </RetryCount> [0..1]
      If HasRetry is true, the action is retried on failure
      the number of times specified in RetryCount.
<RetryWait
      Behavior="xs:string (value comes from list: {'WaitForReboot'|
                 'WaitForInterval'}) [0..1]">  [0..1]
      RetryWaitInterval (TimeInterval)
</RetryWait>
      If the action is set to retry and the attribute Behavior of
      the RetryWait element is WaitForReboot, the computer must
      be rebooted before the action is retried. If the
      Behavior is WaitForInterval, then the action is retried
      after the time interval specified inside the RetryWait tag.
      The possible values are:
      {'PT15M'|'PT30M'|'PT1H'|'PT2H'|'PT4H'|'PT8H'|
      'PT12H'|'P1D'|'P2D'|'P3D'|'P5D'|'P15D'|'P30D'}
      See TimeInterval for more information about the value format.
<HasTemporalDistribution> xs:boolean </HasTemporalDistribution> [0..1]
<TemporalDistribution> NonNegativeTimeInterval
</TemporalDistribution> [0..1]
      If HasTemporalDistribution is true, then the action will
      be distributed over the time duration specified in
      TemporalDistribution to reduce network load.
<PostActionBehavior
```

```
            Behavior="xs:string (value comes from list: {'Nothing'|'Restart'|
                                                 'Shutdown'}) [0..1]"> [0..1]
                  If the Behavior attribute is Restart or Shutdown, the
                  computer is restarted or shutdown (respectively) after
                  the action completes.
            <AllowCancel> xs:boolean </AllowCancel> [0..1]
                  If true, the user is allowed to cancel the
                  restart/shutdown.
            <PostActionDeadlineBehavior> xs:string (value comes from list
                  {'ForceToRun'|'RunAutomatically'}
            </PostActionDeadlineBehavior> [0..1]
            <PostActionDeadlineInterval> ActionMessageTimeInterval
            </PostActionDeadlineInterval> [0..1]
                  When a restart/shutdown is specified, the restart/shutdown will
                  always have a deadline. If the deadline behavior is 'ForceToRun',
                  the user is forced to acknowledge the restart/shutdown when
                  the deadline is reached.  If the deadline behavior is
                  'RunAutomatically', the restart/shutdown will happen
                  automatically, regardless of user acknowledgement.  The deadline
                  interval is specified in the PostActionDeadlineInterval.
            <Title> xs:normalizedString </Title> [0..1]
                  The title of the message displayed before the restart/shutdown.
            <Text> xs:string </Text> [0..1]
                  The text of the message displayed before the restart/shutdown.
      </PostActionBehavior>
</...>
```

ActionSettingsLocks

By default, all the action settings specified are used as the new defaults to the Take Action Dialog. Certain settings might be locked so that the user cannot change them through the dialog. If a setting in ActionSettingsLocks is set to true, then the corresponding group of settings in ActionSettings is locked.

For example, if the TimeRange element of ActionSettingsLocks is true, then the values specified in ActionSettings for HasTimeRange, StartTime, and EndTime cannot be changed in the Take Action Dialog.

This is the schema:

```
<...>
      <ActionUITitle> xs:boolean </ActionUITitle> [0..1]
      <PreActionShowUI> xs:boolean </PreActionShowUI> [0..1]
      <PreAction>
            <MessageTitle> xs:boolean </MessageTitle> [0..1]
            <MessageText> xs:boolean </MessageText> [0..1]
            <AskToSaveWork> xs:boolean </AskToSaveWork> [0..1]
            <ShowActionButton> xs:boolean </ShowActionButton > [0..1]
            <ShowCancelButton> xs:boolean </ShowCancelButton > [0..1]
            <DeadlineBehavior> xs:boolean </DeadlineBehavior > [0..1]
            <ShowConfirmation> xs:boolean </ShowConfirmation> [0..1]
      </PreAction>
      <HasRunningMessage> xs:boolean </HasRunningMessage> [0..1]
      <RunningMessage> [0..1]
            <Title> xs:boolean </Title> [0..1]
            <Text> xs:boolean </Text> [0..1]
      </RunningMessage>
      <TimeRange> xs:boolean </TimeRange> [0..1]
      <StartDateTimeOffset> xs:boolean </StartDateTimeOffset> [0..1]
      <EndDateTimeOffset> xs:boolean </EndDateTimeOffset> [0..1]
      <DayOfWeekConstraint> xs:boolean </DayOfWeekConstraint> [0..1]
      <ActiveUserRequirement> xs:boolean </ActiveUserRequirement> [0..1]
      <Whose> xs:boolean </Whose> [0..1]
      <Reapply> xs:boolean </Reapply> [0..1]
      <ReapplyLimit> xs:boolean </ReapplyLimit> [0..1]
      <ReapplyInterval> xs:boolean </ReapplyInterval> [0..1]
      <RetryCount> xs:boolean </RetryCount> [0..1]
      <RetryWait> xs:boolean </RetryWait> [0..1]
      <TemporalDistribution> xs:boolean </TemporalDistribution> [0..1]
      <PostActionBehavior> [0..1]
            <Behavior> xs:boolean </Behavior> [0..1]
            <AllowCancel> xs:boolean </AllowCancel> [0..1]
            <Deadline> xs:boolean </Deadline> [0..1]
            <Title> xs:boolean </Title> [0..1]
            <Text> xs:boolean </Text> [0..1]
      </PostActionBehavior>
</...>
```

Search components

Search components are used to describe a group of computers for creating a computer group, or for specifying the relevance for a Fixlet, task, baseline, or analysis. Applicability is defined by a combination of three types of components, relevance, property reference and group reference.

This is the schema:

```
<SearchComponentRelevance
      Comparison="xs:normalizedString (possible values: {'IsTrue'|
                                       'IsFalse'}) [0..1]"> [0..*]
      <Relevance> RelevanceString </Relevance> [1]
</SearchComponentRelevance>
      Contains a relevance expression and a comparison {'IsTrue'|'IsFalse'}.
      A computer is in the group if the expression is true for that computer
      and the comparison is 'IsTrue' or the expression is false and
      the comparison is 'IsFalse'.
<SearchComponentPropertyReference PropertyName="xs:normalizedString [0..1]"
      Comparison="xs:normalizedString (possible values:
      {'Contains'|'DoesNotContain'|'Equals'|'DoesNotEqual'}) [0..1]"> [0..*]
      <SearchText> xs:normalizedString </SearchText> [1]
      <Relevance> RelevanceString </Relevance> [1]
</SearchComponentPropertyReference>
   Contains a retrieved property name, a comparison {'Contains' |
   'DoesNotContain' | 'Equals' |       'DoesNotEqual'}, and text
   against which to compare the property result.  A computer is in the group
   if its result for the property meets the comparison with the specified text.
<SearchComponentGroupReference
  GroupName="xs:normalizedString [0..1]"
  Comparison="xs:normalizedString (value comes from list:
  {'IsMember'|'IsNotMember'}) [0..1]" />Contains another computer
  group name and a comparison {'IsMember'|'IsNotMember'}.  A computer
  is in the current group if it is in the other group and
  the comparison is 'IsMember', or if it is not in  the other group and
  the comparison is 'IsNotMember'.
```

Miscellaneous types

RelevanceString

Equivalent to xs:string.

TimeInterval

Values of this type have the format (-)PdDThHmMsS, where dD represents the number of days, T is the date/time separator, hH is the number of hours, mM is the number of minutes, and sS is the number of seconds (that can have up to six decimal digits).

Examples:

PT1M – one minute

P32DT4H24M43.52S – 32 days, 4 hours, 24 minutes, 43.52 seconds

-P12H – negative 12 hours

NonNegativeTimeInterval

TimeInterval that cannot be negative.

ActionMessageMaxPostponementInterval

Possible values:

```
{'PT15M'|'PT30M'|'PT1H'|'PT2H'|'PT4H'|'PT6H'|'PT8H'|
'PT12H'|'P1D'|'P2D'|'P3D'|'P5D'|'P7D'|'P15D'|'P30D'}
```

ActionMessageTimeoutInterval

Possible values:

```
{'PT1M'|'PT2M'|'PT3M'|'PT4M'|'PT5M'|'PT10M'|'PT15M'|'PT30M'|'PT1H'|'PT2H'|
'PT4H'|'PT6H'|'PT8H'|'PT12H'|'P1D'|'P2D'|'P3D'|'P5D'|'P7D'|'P15D'|'P30D'}
```

Chapter 3. Client API

The Client API allows you to use the client (also called the BES client) to interrogate your networked endpoints. Through the API, you gain access to thousands of client properties that you can then reuse in your own agent programs or pass on to other third-party programs. The interface is mediated by a rules document (XML) that defines your queries. The results are calculated in the execution environment of the client that typically has elevated privileges and access rights.

You define the values that are exposed by the Client API using relevance expressions and the complete set of inspectors available for clients. You can use expressions of arbitrary complexity to finely target your search. Note that inspectors are powerful and can also reveal sensitive data; take care to monitor the information that is exposed through this interface.

For the information to become available to the API, a console operator must propagate the program and the rules document to the client computers. Alternatively, another program that uses the Server API can propagate the required files.

The Client API is general-purpose, driven by a rules document and an agent to process the output of the API. However, it is largely used to support compliance of networked endpoints to various policies. As such, the rules document is typically called the compliance document, and both terms are used in this guide.

From a compliance point of view, the API offers many pertinent features. It can target just the computers that are out of compliance and use that same analysis to drive the remediation. Because the Tivoli Endpoint Manager client is under the control of the console, its network role can be modified based on feedback from the API. Among other things, this means you can quarantine any endpoint that is out of compliance. There are two ways to enforce quarantine:

- Self Quarantine: Enables network access control software (such as VPN clients and firewalls) to quarantine the computer based on the compliance evaluation results from the client.
- Network Enforced Quarantine: Enables network admission control frameworks and technologies (such as Cisco Network Admission Control, InfoExpress CyberGatekeeper, Sygate Secure Enterprise or ZoneLabs Integrity) to quarantine the computer based on the compliance evaluation results from the client.

Using either of these methods, you can specify a compliance policy that checks the following:

- Security Configuration: Check that all security policies are in place and there are no security vulnerabilities (weak passwords, open shares, unauthorized USB/wireless devices, insecure settings, and so on).
- Patch Status: Check that the computer has all the latest patches that are required by company policy.
- AntiVirus Status: Check that the AntiVirus agent is installed and enabled, the definitions are up-to-date, and no viruses are currently detected.
- AntiSpyware Status: Check that the computer has AntiSpyware protection installed and working.

- Configuration Standards: Custom compliance checks can easily be added to allow for additional policies.

There is a software developer kit to help you implement these capabilities. The SDK can be found at: http://software.bigfix.com/download/bes/misc/BESClientSDK-6.0.21.5.zip.

Client API terminology

There are some terms you need to know to follow this section of the guide.

Compliance Document A file that contains one or more compliance expression items. The document is formatted as XML. The client maintains a file storage location for each Fixlet site where it locates compliance documents.

Compliance Expression Item Contains a designator, relevance expression, and optionally, a description and comment.

Compliance Response The response made available to the API by the client after it processes the compliance document. The response contains of one or more Compliance Result items.

Compliance Result item Contains a designator, relevance result, description, and a comment.

Designator A string up to 64 characters in length, composed of alphanumeric characters, underscores, or periods.

Relevance Expression Arbitrary relevance expression to be evaluated by the client and reported to the API with the corresponding designator.

Relevance Result The result of evaluating the relevance expression. Plural expressions result in multiple Compliance Result Items that are shown in the response from the API.

Description Some text that is carried from the Compliance Expression item to the Compliance result item. It is passed from the document to the API without processing and can therefore be used to convey arbitrary data about the Compliance expression item.

Comment This is text handled the same as the Description.

Potential use cases

The results of evaluation can be used by access control software to observe the compliance state of the endpoint, by diagnostic software to observe the failure state of particular components, or by startup scripts to verify any aspect of the system computable within the Tivoli Endpoint Manager inspection framework. Compliance might be determined based on detecting that certain conditions do or do not exist on the endpoint. Examples include:

- Detecting if spyware is installed or running. This use might take the form of a compliance policy that a particular spyware detection program is installed, running, and up-to-date, or that a set of executables is not installed.
- Detecting if a virus scanner is installed, running and properly configured.

- Detecting if a firewall is installed, running and properly configured. This use might take the form of a compliance policy that requires the installation of a specific firewall.
- Detecting that network shares are turned off. This use might take the form of requiring that no network shares be defined on the endpoint for the endpoint to be in compliance.
- Detecting that wireless networks are disabled. This use might take the form of requiring that wireless networks be turned off during corporate LAN access.
- Detecting the patch level of the endpoint. The API allows you to check whether there are any critical patches that require installation.

This is an example compliance expression that returns true when there are no critical patches that are relevant on the endpoint:

```
number of relevant fixlets whose (value of header "x-fixlet-source-severity" of it
 as lowercase = "critical") of sites = 0
```

The description and comment fields of the compliance expression item can be used to provide content for your custom application. This technique can help mitigate the need to update your application executables when requirements change. For example, you might mark certain compliance expression items with comments like "Compliant if true" or "Quarantine if true." Then, you might program your application that is based on the results of evaluating the expression and the comments that are returned.

Possible applications include configurable watchdog software that is designed to look for certain conditions and then can disable, limit, or enable the following functions:

- Network shares
- Wireless networks
- Network access

Deploying an agent

To create an agent that can query the Client API and return properties, you can complete the following steps:

- *Build your agent application.* Design it to provide feedback that is based on client properties (such as compliance) returned from relevance expressions.
- *Build compliance documents* to define designators, relevance expressions, descriptions, and comments. These documents might describe compliance with company or government policies.
- *Build an installer* that ensures the correct configuration of the API before copying the program to the client.
- *Create a Fixlet* to target the chosen clients and include an action to copy your documents and run the installer on the client.
- *Deploy the Fixlet* from the console to the chosen end points. You can use extra targeting or computer grouping to reach just the subsets of endpoints you want.
- *Use the API to query the client* when your agent is installed and starts running.
- *Use your agent to respond* with comments, warnings, or quarantine that is based on the data received.

Because your agent can be easily installed with a Fixlet action, you can quickly update the agent or the documents whenever compliance policies change. Including concepts like quarantine involve only a few more steps:

- *Configure a network enforcement agent.* The network enforcement agent must be configured to query the client for the compliance status that is based on the compliance document. The details of the configuration depend on which network enforcement agent is used. The client can automatically configure the network enforcement agent for many network enforcement products.
- *Assess and quarantine.* The network enforcement agent periodically queries the client for the compliance status of the computer. If the computer is not in compliance, it is automatically quarantined by the network enforcement agent.
- *Automate remediation.* Even in quarantine, the client can automatically remediate the computer into compliance, enabling the computer to be taken out of quarantine and seamlessly placed back onto the network.

Software development kit

The Client Compliance API SDK contains the following files:

Table 1. Software Components

File name	Description
ClientCompliance.h ClientComplianceMain.cpp ClientCompliance.vcproj ClientComplianceMT.lib ClientCompliance.exe	C++ sample source and project files to build a self-contained application. (Does not require BESClientComplianceMod.dll COM module to be installed). Does not require COM or Windows scripting to be installed.
BESClientComplianceMod.dll test.vbs ComplianceDumpToReg.vbs	COM module that contains objects and interfaces to drive the API. Use this module together with either of the visual basic samples, and the windows script engine to evaluate compliance.
ComplianceDoc.xml	Sample compliance rules document.
BigFix Client Compliance Configuration.efxm	Masthead file for Fixlets to install and configure the Client Compliance API – automates many of the configuration steps for deploying the API on a managed computer.

The *BigFix Client Compliance Configuration* Fixlet site automates much of the manual configuration that is outlined in the following sections. Run the masthead (the .efxm file) to install the site.

C++ source code

This is a listing of the sample source file ClientComplianceMain.cpp that demonstrates the correct use of the Client API.

```
// COMPLIANCE Function definitions
// **************************************************

CLIENTCOMPLIANCEDLL_API int COMPLIANCE_Open(
 const char *siteurl,
 const char *complianceDocument,
 unsigned int flag // COMPLIANCE_FLAG_*
);
// return value < 0 is COMPLIANCE_ERROR_*
// return value == 0 if Open successfully talks to client
// Make sure to call COMPLIANCE_Close if this succeeds


// **************************************************
CLIENTCOMPLIANCEDLL_API int COMPLIANCE_Close();
// return value < 0 is COMPLIANCE_ERROR_*
// return value = 0 is success
// ****************************************************
CLIENTCOMPLIANCEDLL_API int COMPLIANCE_Progress(
 unsigned int* progressPercent,
 unsigned int* error
);
// return value of COMPLIANCE_PROGRESS_COMPLETE
// return value of COMPLIANCE_PROGRESS_BUSY; progressPercent set
// return value of COMPLIANCE_PROGRESS_ERROR; error set
// After this returns COMPLIANCE_PROGRESS_COMPLETE
// use COMPLIANCE_ResultCount and
// COMPLIANCE_IndexedValue to look at results


//COMPLIANCE response accessors
// ****************************************************
CLIENTCOMPLIANCEDLL_API int COMPLIANCE_ResultCount();
// return value is the number of all values
// return value is COMPLIANCE_ERROR_*


// ****************************************************
CLIENTCOMPLIANCEDLL_API int COMPLIANCE_IndexedValue (
 unsigned int index,  // value to retrieve (starts at 0)
 const char **designator, // designator of value found
 const char **result,  // result of evaluating relevance
 const char **description, // description accompanying designator
 const char **comment  // comment accompanying designator
);
// return value is 0 on success, otherwise COMPLIANCE_ERROR_*
```

Configuring the components

Applications that use the API require that several components be installed and properly configured. In general, you must ensure that the following are correctly configured:

- The client is running.
- You created and saved a file named `ComplianceDoc.xml` in your site data folder: `C:\Program Files\BigFix Enterprise\BES Client\__BESData\<your_site_name>\__Compliance`.
- `BESClientComplianceMod.dll` is installed and registered.
- The registry is configured with appropriate paths to `RequestDir` and `ResponseDir`.
- `RequestDir` and `ResponseDir` are existing folders.

These are the steps to perform:

1. Install COM module.

   If you are using a COM implementation, you can install COM modules with the following command:

   ```
   Regsvr32.exe BESClientComplianceMOD.dll
   ```
2. Create Configuration Key and add Path Data.

   On Windows computers, create the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\ClientComplianceAPI`

   Define the following three string values in the configuration registry key:

   - `RequestDir` – full path where configuration requests are written
   - `ResponseDir` – full path where configuration responses are written

- `ConnectDir` – full path where interface glue scripts and executables are located

3. Create `RequestDir`.

   Create the directory where request files are written by the API. The protections on this directory must permit anyone to write files to it and the client to read and delete files. For Windows systems, this value is set in the RequestDir entry in the registry. For UNIX and Mac systems, the expected paths are:

   UNIX: `/var/opt/BESClient/RequestDir`

   Mac: `/Library/Application Support/BigFix/BES Agent/RequestDir`

4. Create ResponseDir

   Create the directory where response files are written by the API. The protections on this directory must permit anyone to read files from it and the client to write to it. For Windows, this value is set in the ResponseDir entry in the registry. For UNIX and Mac, the expected paths are:

   UNIX: `/var/opt/BESClient/ResponseDir`

   Mac:`/Library/Application Support/BigFix/BES Agent/ResponseDir`

5. Create ConnectDir

   Create a directory where you can keep "glue" scripts and other executables that provide the interface between the Client Compliance API and third-party agents. The protections on this directory must be such that third-party clients can execute files in this directory. For Windows, this value is set in the ConnectDir entry in the registry. For UNIX and Mac, the expected paths are:

   UNIX: `/var/opt/BESClient/ConnectDir`

   Mac: `/Library/Application Support/BigFix/BES Agent/ConnectDir`

6. Create __Compliance.

   Create a data directory that is required by the UNIX compliance executable:

   UNIX only: `/var/opt/BESClient/__BESData/__Compliance`

7. Install the Client.

   The client must be installed on the endpoint. The client must be running to evaluate compliance.

8. Store the Compliance XML

   After you create your compliance document, store it in your action site data directory.

   UNIX: `/var/opt/BESClient/__BESData/actionsite/__Compliance/`*`your_compliance_doc`*`.xml`

   Mac: `/Library/Application Support/BigFix/BES Agent/__BESData/actionsite/__Compliance/`*`your_compliance_doc`*`.xml`

9. Run your agent.

   When everything is properly configured, you can run your agent by submitting your compliance document to the client. Do this by using the gather application:

   UNIX: `./compliance -c http://<server>:52311/cgi-bin/bfgather.exe/actionsite` *`your_compliance_doc`*`.xml`

   Mac: `./BESClientCompliance -c http://<server>:52311/cgi-bin/bfgather.exe/actionsite` *`your_compliance_doc`*`.xml`

API specifications

The BES Client Compliance API is provided as a COM module that registers as BESClientComplianceMod. The module provides three classes:

- Session
- Progress
- Response

These classes are described in the following sections.

BESClientComplianceMod.Session

This object is used to interact with the client. After construction, the object properties *Open* and *Close* can be called to start a compliance evaluation session with the client.

BESClientComplianceMod.Session

Use this object to interact with the BESClient. After construction, the object properties *Open* and *Close* can be called to initiate a compliance evaluation session with the BESClient.

Session Methods	Description
long Open(BSTR siteURL, BSTR rulesXML, long flags);	Attempts to open a compliance evaluation session with the BESClient. The return value is 0 when an evaluation is successfully started. A negative number is returned when an evaluation cannot be started. The return value is one of the integer constants whose name begins with COMPLIANCE_ERROR in the API constants. The BESClient processes the compliance document named <rulesXML> in the __Compliance subfolder of the site data folder of the site whose gather url matches the <siteURL>.
long Close();	Closes the session. Call this property when you are finished with your evaluation. The return value is 0 on success, or one of the integer constants whose name begins with COMPLIANCE_ERROR in the API Constants.

BESClientComplianceMod.Progress

Use this object to collect the progress of the Open session from the BESClient.

Progress Methods	Description
long SessionProgress()	Requests progress of current session. When a session is currently open, contacts the BESClient to collect the current progress state of the current compliance operation. The return values of this property are one of: 1: COMPLIANCE_PROGRESS_COMPLETE 2: COMPLIANCE_PROGRESS_ERROR 3: COMPLIANCE_PROGRESS_BUSY When COMPLIANCE_PROGRESS_COMPLETE is returned, the results of the compliance check are available. Use the Result object to collect the results.

Progress Properties	Description
long GetPercent();	When COMPLIANCE_PROGRESS_BUSY is returned from SessionProgress, the GetPercent property might be accessed to obtain the current estimate of percent completion.
long GetError();	When COMPLIANCE_PROGRESS_ERROR is returned from SessionProgress, the GetError property might be accessed to obtain the current error. The error codes are defined in the API Constants part of this document.

BESClientComplianceMod.Response

Use this object to examine the compliance response from BESClient. This object returns results if the session is still open, after a call to Progress.SessionProgress() returns COMPLIANCE_PROGRESS_COMPLETE, and until Session.Close() is called.

Response Properties	Description
long ResultCount();	Returns the number of results in the compliance response.
BSTR GetDesignator (long index);	Collects the Designator for the result at this index position. An index of 0 gives the first item.
BSTR GetRelevanceResult (long index);	Collects the RelevanceResult for the result at this index position. An index of 0 gives the first item.
BSTR GetDescription (long index);	Collects the Description for the result at this index position. An index of 0 gives the first item.
BSTR GetComment (long index);	Collects the Comment for the result at this index position. An index of 0 gives the first item.

Examples of Client Documents

ComplianceDoc.xml is included in the Client API package as an example of the kind of document you might create to specify compliance with a given policy. It is a plain text file containing XML that can be prepared with any text editor.

Sample Compliance Document

The following is a snippet from that file that tests for the existence on the client of the Windows operating system:

```
<?xml version="1.0"?>
<BESClientComplianceDocument Version="1.0">
 <Date>13 Jun 2004 13:41:57 -0700</Date>
 <ComplianceItem>
  <Designator>IsWindowsOS</Designator>
  <Expression>name of operating system starts with "Win"</Expression>
  <Description>Is a Windows computer</Description>
  <Comment>Compliant if True</Comment>
 </ComplianceItem>

</BESClientComplianceDocument>
```

The file is composed of one or more items each with multiple parts, including:

Designator: An identifier for the retrieved value.

Expression: The relevance expression that is evaluated by the API and returned to the specified designator.

Description: A brief description of the retrieved value.

Comment: A comment about this retrieved value.

All compliance documents follow this format, with as many compliance items as wanted. The following sections illustrate the concept with some more samples.

Check service pack

The following compliance document requires that the client computer is running Windows 7 with at least Service Pack 2 Installed.

```
<BESClientComplianceDocument Version="1.0">
 <ComplianceItem>
  <Designator>XPServicePack</Designator>
  <Expression>
   name of operating system = "Win7"
   AND csd version of operating system
   >= "Service Pack 2"
  </Expression>
  <Description>
   Minimum Windows 7 Service Pack requirement:
   Service Pack 2
  </Description>
  <Comment>Compliant if True</Comment>
 </ComplianceItem>
</BESClientComplianceDocument>
```

Check for a running app

This compliance document requires that the application DefWatch.exe is running on the computer.

```
<BESClientComplianceDocument Version="1.0">
 <ComplianceItem>
  <Designator>RequiredProgram</Designator>
  <Expression>exists running application "DefWatch.exe"</Expression>
  <Description>DefWatch.exe must be running.</Description>
  <Comment>Compliant if True</Comment>
 </ComplianceItem>
</BESClientComplianceDocument>
```

Update patches

The following compliance document requires that there are less than 10 unapplied patches with severity rating Critical, and the elapsed time since the oldest unapplied critical patch is less than 30 days.

```
<BESClientComplianceDocument Version="1.0">
 <ComplianceItem>
  <Designator>NumCritical</Designator>
  <Expression>
   10 > number of relevant fixlets whose
   (value of header "x-fixlet-source-severity" of
   it as lowercase = "critical") of sites
  </Expression>
  <Description>
   Total # of critical patches must be < 10
  </Description>
  <Comment>Compliant if True</Comment>
 </ComplianceItem>
<ComplianceItem>
 <Designator>MaxPatchAge</Designator>
 <Expression>
  if (exists relevant fixlet whose ((value of header "X-Fixlet-Source-Severity" of it
   as lowercase = "critical") AND (value of header "X-Fixlet-Source-Release-Date"
   of it does not contain "Unspecified")) of sites "bessecurity")
   then ((30 > ((it - 1) of maximum of ((preceding texts of firsts " day"
   of ((now + 1*day - it) as string) as integer)
   of ((((it as string & " 00:00:00 -0700")
   of ((value of header "X-Fixlet-Source-Release-Date" of it) of relevant fixlets
   whose ((value of header "x-fixlet-source-severity" of it as lowercase = "critical")
   AND (value of header "X-Fixlet-Source-Release-Date" of it
   does not contain "Unspecified")) of sites)) as time))))) as string) else "True"
 </Expression>
 <Description>
  Elapsed time since the oldest unapplied critical patch
  is less than 30 day(s)
 </Description>
 <Comment>Compliant if True</Comment>
</ComplianceItem>
</BESClientComplianceDocument>
```

Check for anti-virus

This compliance document requires that Norton Anti-Virus Corporation Edition is running and the Anti-Virus definitions are less than 10 days old.

```
<BESClientComplianceDocument Version="1.0">
 <ComplianceItem>
  <Designator>AVDefinitionAge</Designator>
  <Expression>
   FALSE OR ((exists running application "vptray.exe" OR exists
    running application "rtvscan.exe") AND exists
    key "HKEY_LOCAL_MACHINE\Software\Symantec\SharedDefs"
    of registry AND exists value "NAVCORP_70"
    of key "HKEY_LOCAL_MACHINE\Software\Symantec\SharedDefs"
    of registry AND (now - ((following text of position 6
    of (preceding text of last "." of following text of last "\"
    of (value "NAVCORP_70"
    of key "HKEY_LOCAL_MACHINE\Software\Symantec\SharedDefs"
    of registry as string)) & " " & first 3 of following text
    of position (first 2 of following text of position 4
    of (preceding text of last "." of following text
    of last "\" of (value "NAVCORP_70"
    of key "HKEY_LOCAL_MACHINE\Software\Symantec\SharedDefs"
    of registry as string)) as integer * 3 - 3)
    of "JanFebMarAprMayJunJulAugSepOctNovDec"
    & " " & first 4 of (preceding text of last "."
    of following text of last "\" of (value "NAVCORP_70"
    of key "HKEY_LOCAL_MACHINE\Software\Symantec\SharedDefs"
    of registry as string)) & " 00:00:00 - 0700")
    as time)) < 10 * day)
  </Expression>
  <Description>
   Elapsed time since the virus definition was last updated is less than
    the specified number of day(s).
  </Description>
  <Comment>Compliant if True</Comment>
 </ComplianceItem>
</BESClientComplianceDocument>
```

Chapter 4. Dashboard API

Tivoli Endpoint Manager exposes an interface that allows you to author your own dashboards, granting you the power to create customized views into whatever aspect of your network you want to monitor. Dashboards, wizards and Web Reports are three different manifestations of the same underlying concept. The main differences are:

- *Dashboards* Typically open in a document window and provide for easily updated views. They usually consist of a collection of tabular and graphical widgets that provide a condensed view of your network.
- *Wizards* Typically open in a stand-alone dialog window. They often have multiple pages to ease the user through a complicated installation or procedure. Both wizards and dashboards have the file extension of OJO.
- *Web Reports* Are similar to dashboards, but offer static web-facing views of your data. They have a file extension of BESWRPT.

Despite these differences, the interfaces are driven by the same fundamental language elements, referred to as the Dashboard API. This guide describes how Relevance elements work within the Dashboard API, letting you analyze, aggregate, and visualize multiple facets of your networked clients.

The Dashboard API is based on HTML and XML. The XML defines hooks into the console or Web Reports and it has a single primary section that holds HTML. The HTML section in turn can hold anything that a browser can render. In addition to the basic browser environment, the console or the Web Reports application injects a library of JavaScript hooks that provide access to various API functions.

The content that is embedded in the HTML section is typically a mix of JavaScript, HTML, Flex, and Relevance expressions that are evaluated with the JavaScript function *EvaluateRelevance*. As soon as they are received, the results of the evaluation are embedded directly into the page and it is redrawn.

The document can also specify how your dashboard is attached to the console UI, for example as a menu, a list, or an item on the navigation bar. Dashboard documents can also be linked to or from other documents. They can be imported or exported and used by internal or external Fixlet sites.

Linking

The console intercepts all links before they are followed. If the "link:" protocol is used, then they are interpreted as an instruction to the console. This is typically something like:

```
link:opendoc?siteid=123&objectid=456
```

This link opens the given document from the specified site. Additional capabilities include launching wizards and various console dialogs (such as Take Action or Visualization), executing search operations, and more.

Relevance in dashboards

Dashboard markup is passed through a pre-processor that recognizes Relevance tags. These Relevance expressions are evaluated and the tag is then replaced by those results. The results are then wrapped in a special <span> tag so they can be reevaluated and updated as they change, creating a dynamic document. You can also use JavaScript to evaluate Relevance and for other document object model manipulations.

You can evaluate relevance in dashboards in two ways, both of which are compatible with wizards and Web Reports.

The first technique is to use the <?relevance expression ?> tag. This method is used when you want to create sections of properly formatted HTML containing Relevance results. These instructions are parsed at load time and replaced by the result of evaluating the given expression. The result is coerced into the *html* inspector type that means that the string is escaped, ensuring that it does not interfere with any surrounding HTML code. The html type is described in greater detail later in this guide.

The second technique works when you must evaluate relevance from within JavaScript. In this instance, use the *EvaluateRelevance* function. This function is defined in an external JavaScript source file that is automatically included by console documents that support dashboard functions, including Fixlets, tasks, baselines, analyses, wizard documents, and Web Reports. In Web Reports, the external definition is somewhat different, but it functions the same.

From any script code, you can evaluate a relevance expression and get the results back as a string, with a statement like this:

```
myDiv.innerText = EvaluateRelevance( "expression" );
```

The expression is a relevance expression string just like in the <?relevance ?> case. The result of EvaluateRelevance depends on whether the expression is a singular expression or a plural expression. If theexpression is singular, the result is a string. If it is plural, the result is an array of strings. Unlike the results of relevance in processing instructions, none of the strings are HTML escaped unless you use the "as html" cast explicitly. If an error is encountered, EvaluateRelevance throws an exception. You can get a descriptive error string as follows:

```
try {
 myDiv.innerText = EvaluateRelevance( "expression" );
}
catch (e){
 window.alert( "Error encountered evaluating relevance: " + e.description );
}
```

Debugging dashboards

The *Debug* menu in the console offers some helpful tools for creating, editing, and debugging dashboards. Among them is the *Presentation Debugger* (also called the Dashboard Debugger in some versions of the program) that is turned off by default. To enable the Debug menu:

1. Press Ctrl-Shift-Alt-D to display the Debug Dialog.
2. Select the check box next to *Show Debug Menu*.

Alternatively, you can edit the registry:

1. Locate the key HKCU\Software\BigFix\Enterprise Console\NoEditMenus.

2. Set the DWORD value Show Debug Menu to 1.
3. Restart the console.

The Presentation Debugger dialog lets you write a Relevance Expression and test it. You can enter the expression in String, HTML, or Presentation (XML) style. There are some buttons that are attached to this dialog, although they differ depending on the context of the dialog:

- *Cancel*: Available when the debugger is called from a wizard tool, use this button to cancel the insertion of the wizard.
- *Insert*: Also available when the debugger is called from a wizard tool, this button lets you insert the wizard icon into a custom Fixlet, task, baseline, or analysis.
- *Open File*: Opens a File Open dialog to import an existing Relevance statement or a dashboard-style file.
- *Evaluate*: Click this button to evaluate the specified Relevance statement. The results are displayed in the box below.

You can also attach a site (from the pull-down menu) and a Fixlet ID to your expression, by using the appropriate input boxes.

This dialog can be accessed whenever you create a new or custom Fixlet, task, baseline, or analysis. In the description tag, enter the text that you desire, and then from the toolbar at the top, insert the Wizard Hat icon. This displays the Presentation Debugger dialog.

Editing dashboards

The Presentation Debugger is suitable for debugging a few lines at a time, but it is inconvenient for full-fledged development. Some suggestions for improving the process include:

- Use the debugger, but develop iteratively, a short section at a time.
- Put the content in a Fixlet site and subscribe to it. The changes that you make are automatically replaced as you edit.
- Use *Web Reports* > *Create Report* > *Advanced: Blank Report*that allows you to create, edit and view your dashboard as a Web Report.
- Create a wizard *.ojo* file on your local drive, and then use *Debug* > *Load Wizard* to load and view it.

Chapter 5. Database API

This section of the guide describes a set of SQL views, or virtual tables, that constitute the database application programming interface (API). These views are provided to enable your applications to query the database directly by using MSSQL-compatible interfaces such as ADO or ODBC. A typical application might be a Perl cgi program that creates an HTML report for online viewing. Perl uses the database Interface (dbi) to connect to the SQL database. Any programming language that has an ODBC interface can be used to access the database.

The SQL format of the database makes it easy to create various views of the tables, including Fixlet, action, computer, and retrieved property tables. With a few simple SELECT commands, you can create filtered and sorted views of the various databases. These can be used to prepare custom reports, audit trails or capture snapshots of the environment.

The database API provides backwards compatibility across releases: applications written against them should continue to work in later releases of Tivoli Endpoint Manager unless product functionality or underlying content changes in a way that renders these views inapplicable. In this guide, there are references to BigFix Enterprise Suite, or BES. This refers to what we now call Tivoli Endpoint Manager, and is a legacy of previous releases.

Access to the database for the SELECT commands listed here is granted to all authorized users of the Console. Because these views are intended for output only, users are not able to update, delete, or otherwise modify the database with this API. For information about how to create actions and tasks that might modify the database, see the section of this guide titled *Platform API.*

The following sections describe the views that are supplied to support the SQL Database API.

BES_FIXLETS

This view provides a list of all Fixlets in the BES Database. This table is useful in conjunction with the BES_RELEVANT_FIXLETS and BES_ACTIONS table to get the Fixlet name. Custom Fixlet content is provided under the "ActionSite" sitename.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Unique Fixlet ID
Name	varchar(255)	Fixlet name

Example:

```
select Sitename, ID, Name from BES_FIXLETS where Sitename = 'Enterprise Security'
order by Sitename, ID
```

BES_TASKS

This view provides a list of all tasks (including custom tasks) in the BES Database. This table is useful in conjunction with the BES_RELEVANT_TASKS and BES_ACTIONS table to get the task name.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Unique task ID
Name	varchar(255)	Task name

Example:

```
select Sitename, ID, Name from BES_TASKS where Sitename = 'Enterprise Security'
order by Sitename, ID
```

BES_ANALYSES

This view provides a list of all Analyses (including custom Analyses) in the BES Database.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Unique analysis ID
Name	varchar(255)	Analysis name

Example:

```
select Sitename, ID, Name from BES_ANALYSES where Sitename = 'BES Support'
order by Sitename, ID
```

BES_BASELINES

This view provides a list of all Baselines in the BES Database. This table is useful in conjunction with the BES_RELEVANT_BASELINES and BES_ACTIONS table to get the baseline name.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Unique baseline ID
Name	varchar(255)	Baseline name

Example:

```
select Sitename, ID, Name from BES_BASELINES where Sitename = 'Enterprise Security'
order by Sitename, ID
```

BES_COMPUTERGROUPS

This view provides a list of all Computer Groups in the BES Database.

Column	Type	Description
ID	int	Unique Group ID

Name	varchar(255)	Computer group name

Example:

```
select ID, Name from BES_ComputerGroups where Name LIKE 'Chicago Office%' order by ID
```

BES_COLUMN_HEADINGS

This view provides access to all the retrieved property information collected about client computers by the BES Database. Retrieved properties that return multiple results are expressed in this view by a value field that contains the multiple results separated by a newline character. Column headings whose "Value" contains more than 8000 characters are truncated to 8000 characters in this view.

Column	Type	Description
ComputerID	int	Computer ID
Name	varchar(255)	Retrieved property name
Value	varchar(8000)	Newline separated list of retrieved property values
IsFailure	Tinyint	Nonzero if the retrieved property failed to evaluate on the BES Client

Example:

```
select ComputerID, Name, Value, IsFailure from BES_COLUMN_HEADINGS where
Name = 'Total HD Space' order by ComputerID
```

BES_RELEVANT_FIXLETS

This view contains an entry for every Fixlet/computer pair in which the Fixlet is relevant on that computer. This view includes custom Fixlet content.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Fixlet ID
ComputerID	int	Computer ID
Version	int	Number of times the Fixlet is modified

Example:

```
select F.Sitename, F.ID, F.Name, R.ComputerID from BES_FIXLETS F, BES_RELEVANT_FIXLETS R
where F.Sitename = R.Sitename AND F.ID = R.ID
```

BES_RELEVANT_TASKS

This view contains an entry for every task/computer pair (including custom Tasks) in which the task is relevant on that computer.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Task ID

ComputerID	int	Computer ID
Version	int	Number of times the task is modified

Example:

```
select T.Sitename, T.ID, T.Name, R.ComputerID from BES_TASKS T, BES_RELEVANT_TASKS R
where T.Sitename = R.Sitename AND T.ID = R.ID
```

BES_RELEVANT_BASELINES

This view contains an entry for every baseline/computer pair in which the baseline is relevant on that computer.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Baseline ID
ComputerID	int	Computer ID
Version	int	Number of times the baseline is modified

Example:

```
select B.Sitename, B.ID, B.Name, R.ComputerID from BES_BASELINES B, BES_RELEVANT_BASELINES R
where B.Sitename = R.Sitename AND B.ID = R.ID
```

BES_ACTIONS

This view contains an entry for every action/computer pair where the action was received by the computer.

Column	Type	Description
ActionID	int	Action ID
ComputerID	int	Computer ID
Name	varchar(255)	Title of the action
Username	varchar(32)	Database user name of action issuer
StartTime	datetime	Time at which the action was issued
FixletID	int	Source Fixlet ID
Sitename	varchar(128)	Source Fixlet site name
ActionStatus	text	A brief summary of the state of the action for this computer

Example:

```
select * from BES_ACTIONS where ActionStatus = 'Executed'
```

BES_RELEVANT_FIXLET_HISTORY

This view contains an entry for every Fixlet/computer pair that has ever been relevant, with timestamps indicating the first time it became relevant, the last time it became relevant (the same as FirstBecameRelevant if it became relevant only once), and the last time it became non-relevant. Some of these fields might be NULL if the event in question never occurred. This view includes custom Fixlet content.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Fixlet ID
ComputerID	int	Computer ID
FirstBecameRelevant	datetime	Time at which Fixlet first became relevant
LastBecameRelevant	datetime	Time at which Fixlet last became relevant
LastBecameNonRelevant	datetime	Time at which Fixlet last became non-relevant
Version	int	Fixlet version

BES_RELEVANT_TASK_HISTORY

This view contains an entry for every task/computer pair (including custom tasks) that has ever been relevant, with timestamps indicating the first time it became relevant, the last time it became relevant (the same as FirstBecameRelevant if it became relevant only once), and the last time it became non-relevant. Some of these fields might be NULL if the event in question never occurred.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Task ID
ComputerID	int	Computer ID
FirstBecameRelevant	datetime	Time at which task first became relevant
LastBecameRelevant	datetime	Time at which task last became relevant
LastBecameNonRelevant	datetime	Time at which task last became non-relevant
Version	int	Task version

BES_RELEVANT_BASELINE_HISTORY

This view contains an entry for every baseline/computer pair that has ever been relevant, with timestamps indicating the first time it became relevant, the last time it became relevant (the same as FirstBecameRelevant if it became relevant only once), and the last time it became non-relevant. Some of these fields might be NULL if the event in question never occurred.

Column	Type	Description

Sitename	varchar(128)	Source Fixlet site name
ID	int	Baseline ID
ComputerID	int	Computer ID
FirstBecameRelevant	datetime	Time at which baseline first became relevant
LastBecameRelevant	datetime	Time at which baseline last became relevant
LastBecameNonRelevant	datetime	Time at which baseline last became non-relevant
Version	int	Baseline version

BES_FIXLET_PROPERTIES

This view lists the different properties associated with each Fixlet (including custom Fixlets), such as the severity.

Column	Type	Description
Sitename	varchar(128)	Fixlet site name
ID	int	Fixlet ID
PropertyName	varchar(32)	Property name
PropertyValue	text	Property value

Example:

```
select BF.Sitename, BF.ID, BF.Name, BFP.PropertyValue AS 'Severity'
from BES_FIXLETS BF, BES_FIXLET_PROPERTIES BFP
where BF.Sitename = BFP.Sitename
and BF.ID = BFP.ID AND BFP.PropertyName = 'Source Severity'
```

BES_TASK_PROPERTIES

This view lists the different properties associated with each task (including custom Tasks), such as the severity.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Task ID
PropertyName	varchar(32)	Property name
PropertyValue	text	Property value

Example:

```
select BT.Sitename, BT.ID, BT.Name, BTP.PropertyValue AS 'Severity'
from BES_TASKS BT, BES_TASK_PROPERTIES BTP
where BT.Sitename = BTP.Sitename AND BT.ID = BTP.ID
and BTP.PropertyName = 'Source Severity'
```

BES_BASELINE_PROPERTIES

This view lists the different properties associated with each baseline, such as the severity.

Column	Type	Description
Sitename	varchar(128)	Source Fixlet site name
ID	int	Baseline ID
PropertyName	varchar(32)	Property name
PropertyValue	text	Property value

Example:

```
select BB.Sitename, BB.ID, BB.Name, BBP.PropertyValue AS 'Severity'
from BES_BASELINES BB, BES_BASELINE_PROPERTIES BBP
where BB.Sitename = BBP.Sitename AND BB.ID = BBP.ID
and BBP.PropertyName = 'Source Severity'
```

Example report generator

The following Perl script, with the appropriate dsn name and login supplied in the DBI->connect line, accesses the database and print the contents of the four principal views in HTML tables.

```
#
# Example Perl cgi script which shows the contents of a Database
#

use strict;
use CGI;
use DBI;
use CGI::Carp qw(fatalsToBrowser);

$| = 1;

# Insert your own database details here
my $dbh = DBI->connect ("dbi:ODBC:bes_locke", "bigfix", "bigfix")
or die "unable to connect to db";

#--------------------------------------------------------------------------------
# Create the HTML to output your report. Here, we refer to a computer named 'LOCKE':
print "content-type: text/html\n\n";
print "<html><body>";
print "<h1>Contents of Database on LOCKE</h1>";

#...............................................
# Print out all column headings:
{
 print "<h3>Column Headings</h3>";
 print "<table width=100% bgcolor=#b0b0f0 border=1><tr>";
 print "<td>ComputerID</td><td>Name</td>";
 print "<td>Value</td><td>IsFailure</td></tr>";

 # set up the SQL query:
 my $query = "select ComputerID, Name, Value, IsFailure ";
 $query .= "from BES_COLUMN_HEADINGS";
 my $sth = $dbh->prepare($query);
 $sth->execute();
 my @row;
 while(@row = $sth->fetchrow_array){
  print "<tr><td>";
  print join("</td><td>", @row);
  print "</td></tr>";
 }
 print "</table>";
}

#...............................................
# Print out all relevant fixlets
{
 print "<h3>Relevant Fixlets</h3>";
 print "<table width=100% bgcolor=#f0b0b0 border=1>";
 print "<tr><td>Sitename</td><td>ID</td>";
 print "<td>ComputerID</td></tr>";

 # set up the SQL query:
 my $query = "select Sitename, ID, ComputerID from BES_RELEVANT_FIXLETS";
 my $sth = $dbh->prepare($query);
 $sth->execute();
 my @row;
 while(@row = $sth->fetchrow_array){
  print "<tr><td>";
  print join("</td><td>", @row);
  print "</td></tr>";
 }
 print "</table>";
}

#...............................................
# Print out all actions
{
 print "<h3>Actions</h3>";
 print "<table width=100% bgcolor=#d080ff border=1>";
 print "<tr><td>ActionID</td><td>ComputerID</td>";
 print "<td>Name</td><td>Username</td><td>Start Time</td>";
 print "<td>FixletID</td><td>Sitename</td><td>ActionStatus</td></tr>";

 # set up the SQL query:
 my $query = "select ActionID, ComputerID, Name, Username, StartTime, ";
 $query .= "FixletID, Sitename, ActionStatus from BES_ACTIONS";
 my $sth = $dbh->prepare($query);
 $sth->execute();
 my @row;
 while(@row = $sth->fetchrow_array){
  print "<tr><td>";
  print join("</td><td>", @row);
  print "</td></tr>";
 }
 print "</table>";
}

#...............................................
# Print out all known fixlets
{
 print "<h3>Known Fixlets</h3>";
 print "<table width=100% bgcolor=#b0f0b0 border=1>";
 print "<tr><td>Sitename</td><td>ID</td><td>Name</td></tr>";

 # set up the SQL query:
 my $query = "select Sitename, ID, Name from BES_FIXLETS";
 my $sth = $dbh->prepare($query);
 $sth->execute();
 my @row;
 while(@row = $sth->fetchrow_array){
  print "<tr><td>";
  print join("</td><td>", @row);
  print "</td></tr>";
 }
}

print "</body></html>";
```

Chapter 6. WebReports API

Tivoli Endpoint Manager exposes an API to help you author your own Web Reports, granting you the power to create customized views into whatever aspect of your network that you want to monitor. The Web Reports API is based on the Dashboard API, but it operates with static web-facing views of your data. These views have a file extension of BESWRPT.

The Web Reports API is based on HTML and XML, with a new interface based on SOAP. The XML defines hooks into the console or Web Reports and it has a single primary section that holds HTML. The HTML section in turn can contain anything that a browser can render. In addition to the basic browser environment, there is a library of JavaScript hooks available to provide access to various functions.

The content that is embedded in the HTML section is typically a mix of JavaScript, HTML, Flex, and Relevance expressions that are evaluated with the JavaScript function EvaluateRelevance. As soon as they are received, the results of the evaluation are embedded directly into the page and it is redrawn.

Web Reports markup is passed through a pre-processor that recognizes Relevance tags. The Relevance language uses a special class of inspectors called Datastore inspectors that can help you query and aggregate statistics on your data. These Relevance expressions are evaluated and the tag is then replaced by those results. You can also use JavaScript to evaluate Relevance and for other document object model manipulations.

Authoring Web Reports

Although there is much commonality between dashboards and Web Reports, they have several important differences.

- The Web Reports application maintains Fixlet history inspectors, but the console does not. Fixlet history inspectors are of the form: *first became relevant of*.
- Certain current inspectors work only in the console. These include inspectors such as *current console user* or *current computer*.
- Inspectors that work with Locally Hidden/Visible Fixlets always return nothing. Global Fixlets, however, still work as expected.
- While the dashboard can report only on the local server, Web Reports can span multiple servers. The ActionSite has the server name added: ActionSite (BES_Server_Name).
- Javascript *EvaluateRelevance* calls are instantaneous on the console, but in a Web Report, they generate HTTP requests that might average up to one second per call. Web Reports has an *AsyncEvaluateString* call that does not have the same penalty, because it is asynchronous. It takes an HTML object ID as a parameter, and the results are set to the object's innerHTML for use with tags like <div>, <span>, or <p>.
- Certain links that work in the console do not work in the Web Reports. Links to Fixlet filters do not work, while links to computer groups, computers, Fixlets, tasks, and analyses do.

Converting a dashboard

The following process converts a dashboard-style *.ojo* file into a custom report:

1. Open the *.ojo* file and rename it to *.besrpt*.
2. Delete everything outside of the *![CDATA]* tags. This typically consists of the first line and last few lines of the file.
3. Resolve any links that would not be available in Web Reports, such as style sheet references or links to pictures.
4. Import the .besrpt file into BES Custom Reports and analyze it for correctness.

Propagating Web Reports

Web Reports has a generalized report format that can incorporate reports from content sites, allowing new reports to be added, edite, or removed. The report file must have the extension .beswrpt, and is written in XML:

```
<BESWebReport>
 <Name>Report name</Name>
 <Description>Report description</Description>
 <Type>[TemplateReport] | [CustomReport] | [ExternalReport]</Type>
 <Category>Report category</Category>
 <Source>Report source</Source>
 <URLParameters>param1=value1&amp;param2=value2</URLParameters>
 <Data>Custom data or external URL</Data>
 </BESWebReport>
```

The .beswrpt file can also contain multiple reports in one xml file:

```
 <BESWeb Reports>
 <BESWebReport>
 .
 .
 .
 </BESWebReport>
 <BESWebReport>
 .
 .
 .
 </BESWebReport>
 </BESWeb Reports>
```

The *Name*, *Category*, and *Source* are listed in the basic report list, and *Description* shows up if the question mark (?) next to it is clicked. If the category is blank, a suitable default is filled in, depending on the type of the report. A blank source just appears blank.

URLParameters and Data depend on the Type tag. The type tag can be one of three strings, *TemplateReport*, *CustomReport*, or *ExternalReport* (case-sensitive).

Template report

A template report provides arguments for, and then runs, a previously existing Web Report. The URL for the report is specified in the <URLParameters> tag. The *Data* tag does not exist for this report.

For example, to create a particular *Issue Assessment* report you might create a template report like the following:

```
 <BESWebReport>
 <Name>Example Issue Assessment Report</Name>
 <Description>
```

```
  An example report. Runs the Issue Assessment report with
  specific Fixlets, BES Support/129 and BES Support/173.
 </Description>
 <Type>TemplateReport</Type>
 <Category>Issue Assessment</Category>
 <Source>IBM</Source>
 <URLParameters>
  FixletParam=BES Support%2f129&amp;FixletParam=BES Support%2f173&amp;page=VAReport
 </URLParameters>
 </BESWebReport>
```

Note: The URL must be properly escaped inside the XML.

Custom report

A custom report is constructed from HTML, relevance expressions, and JavaScript. The <Data> tag contains a definition of the report in a CDATA block to allow the HTML and JavaScript portions to be presented normally, without the need for escape characters.

This is a custom web report that retrieves and prints the names of your networked computers:

```
<BESWebReport>
 <Name>Example Custom Report</Name>
 <Description>
  A customized report, prints all computer names.
 </Description>
 <Type>CustomReport</Type>
 <Category>Custom Report</Category>
 <Source>IBM</Source>
 <Data>
  <![CDATA[
   <div>Computer Names</div>
   <?Relevance names of bes computers?>
  ]]>
 </Data>
 </BESWebReport>
```

External report

An External Report places an absolute link into the report list that can point to any valid web page. It allows, among other things, linking to another reporting engine or service. The Data tag contains the full external URL and must start with http or https.

This is an example report that displays the IBM home page:

```
<BESWebReport>
 <Name>Example External Report</Name>
 <Description>
  Home Page
 </Description>
 <Type>ExternalReport</Type>
 <Category>External Report</Category>
 <Source>IBM</Source>
 <Data>
  http://www.ibm.com
 </Data>
</BESWebReport>
```

Exporting to PDF

You can export content reports by using EvaluateRelevance to PDF. By default, Web Reports allows only ten seconds for generating the PDF. If your reports are taking longer than that, you can adjust the wait, by performing the following steps:

1. Include an element with id='wr_content_will_signal_completion' to your report, like this:
   ```
   <div id='wr_content_will_signal_completion'></div>
   ```
2. When your report is ready to print, set the DOM property document.wr_content_complete to true with the following JavaScript:
   ```
   document.wr_content_complete = true;
   ```
3. If the report never sets the document.wr_content_complete property, it times out after an hour and reports an error. To avoid delays with PDF generation, you can wrap your report in a try/catch block and, in the catch portion, set
   ```
   document.wr_content_complete = true
   ```

Setting a title

You might need to specify a custom title <div>. Here is how:

```
var titleDiv = document.getElementById( 'wr_titlediv' );
titleDiv.className = 'my title class';
titleDiv.innerHTML = 'my new title';
```

Setting OnLoad events

If you want to add an onload event, make sure that you call the original Web Reports onload event at some point. This is how you would typically create an onload event:

```
window.onload = function() {
     hourglass(document.body);
     RetrieveData();
     pointer (document.body);
}
```

To work with Web Reports, make sure that you include the original onload event as follows:

```
var oldonload = window.onload;
window.onload = function() {
    hourglass(document.body);
    RetrieveData();
    pointer(document.body);
    oldonload();
}
```

This ensures that the Web Reports UI is displayed correctly.

SOAP API

In addition to HTTP GET, Web Reports provides a Simple Object Access Protocol (SOAP) interface to allow external applications to interact with Web Reports and retrieve data. SOAP specifies a protocol for exchanging structured data, by using XML format. The protocol has three parts:

Envelope
Defines the contents of the message and how it is parsed.

Header
Web Reports headers are used to send login information verifying that the user of the SOAP application has permission to take the action.

Body The contents of the Web Report request.

Most client libraries require a Web Services Description Language (WSDL) file as well as the method name and parameters to pass into the method. The WSDL file describes the network endpoints required to query the database by using Web Reports. Some of the available methods and their parameters include:

```
GetRelevanceResult( String relevanceExpression, String username, String password )
GetRelevanceResult( String relevanceExpression, String username, String password )
StoreSharedVariable( dashID, variableName, variableValue,
 [success/failure callback], [database id] )
DeleteSharedVariable( dashID, variableName, [success/failure callback], [database id] )
```

Configuring SOAP

The only configuration setting required for SOAP is the location of the *WSDL* file that is stored in the registry. The setting name is wsdl and the value is the path to the WSDL file. The default value is the sample file, *relevance.wsdl*, in the installation directory. This WSDL file defines *relevanceExpr*that allows you to evaluate a relevance expression. If you installed Tivoli Endpoint Manager to a folder named BigFix Enterprise, you can find the file in this directory:

```
BigFix Enterprise\BES Server\BESReportsServer\wwwroot\soap
```

The Registry contains the actual location for your particular installation:

```
HKLM\SOFTWARE\BigFix\Enterprise Server\BESReports\Paths
```

On a 64-bit machine, you must select the Wow6432 node:

```
HKLM\SOFTWARE\Wow6432Node\BigFix\Enterprise Server\BESReports\Paths
```

This registry key has a string value named wsdl that contains the path name for the WSDL file.

SOAP URL

Interacting with Web Reports SOAP is different depending on what library you are using. However, there are two things that most libraries require: The location of the WSDL, and a location to send it requests. The location of the Web Reports WSDL is:

```
http://example.com/webreports?wsdl
```

Where "example.com" is the URL of the Web Reports server. The location for posting requests is:

```
http://example.com/webreports/soap
```

SOAP headers

Web Reports uses request and response headers in all SOAP methods except for GetRelevanceResult. Headers are used to send login information to verify that the user associated with the SOAP request has the correct permissions to execute it. The data that are returned from the SOAP call is filtered by those permissions.

Two kinds of headers are used, login or authentication. A login header takes a username and a password to authenticate. The server replies to the request with a response header that contains a session token. In subsequent requests, until the user's session times out, SOAP requests can be authenticated by using an authentication header that requires a username and the provided session token.

Table 2. Request Header

Name	Type	Occurs	Description
RequestHeaderElement	LoginHeader or AuthenticationHeader	1	Login information to verify that a user has permission to perform this action.

Table 3. Response Header

Name	Type	Occurs	Description
RequestHeaderElement	ResponseHeader	1	Contains a session token passed back by the server so that subsequent requests do not need to pass the user's password to the server.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
    xmlns:h="http://schemas.example.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.example.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>example</username>
      <password>pswd</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    ...
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <bf:ResponseHeaderElement
    xmlns:bf="http://schemas.example.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
      <bf:sessionToken>3jPHTrTJSea2o76xiGM4K8fQuSE</bf:sessionToken>
    </bf:ResponseHeaderElement>
  </soapenv:Header>
  <soapenv:Body>
    ...
  </soapenv:Body>
</soapenv:Envelope>
```

Subsequent requests

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="AuthenticationHeader"
    xmlns:h="http://schemas.example.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.example.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>example</username>
      <sessionToken>3jPHTrTJSea2o76xiGM4K8fQuSE</sessionToken>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    ...
  </s:Body>
</s:Envelope>
```

Subsequent requests require the username and the session token that is provided by the primary request.

SOAP methods

There are several methods available for you to query the Web Reports server or add to the database. Each method is described in detail in the following sections.

GetRelevanceResult

GetRelevanceResult is a SOAP method that evaluates a relevance expression and returns the result as a simple array of strings. This is the only method that does not require request and response headers.

Table 4. Parameters

Name	Type	Occurs	Description
relevanceExpr	string	1	The relevance expression to be evaluated.
username	string	1	Login name of Web Reports user used to evaluate relevance.
password	string	1	Password of Web Reports user used to evaluate relevance.

Table 5. Return Values

Name	Type	Occurs	Description
a	string	0+	If there is only 1 occurrence, it represents the singular result from the expression. If there are 0 occurrences, it represents an empty plural result. If there are 2 or more occurrences, each occurrence represents one element of a plural result.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <GetRelevanceResult xmlns="http://schemas.bigfix.com/Relevance">
      <relevanceExpr>now</relevanceExpr>
      <username>bigfix</username>
      <password>bigfix</password>
    </GetRelevanceResult>
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <GetRelevanceResultResponse xmlns="http://schemas.bigfix.com/Relevance">
```

```
      <a>Mon, 13 Sep 2010 14:42:55 -0700</a>
    </GetRelevanceResultResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

GetStructuredRelevanceResult

GetStructuredRelevanceResult is a SOAP method that evaluates a relevance expression and returns the result as a complex (structured) object. This object gives access to information about the type of the result, evaluation time, errors, along with a strongly typed list of results. Note that this method requires request and response headers.

Table 6. Parameters

Name	Type	Occurs	Description
relevanceExpr	string	1	The relevance expression to be evaluated.

Table 7. Return Values

Name	Type	Occurs	Description
StructuredRelevanceResult	StructuredRelevanceResult	1	An object containing information about the result of the relevance query, along with a list of results.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
    xmlns:h="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.bigfix.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>bigfix</username>
      <password>bigfix</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <GetStructuredRelevanceResult xmlns="http://schemas.bigfix.com/Relevance">
      <relevanceExpr>ids of bes computers</relevanceExpr>
    </GetStructuredRelevanceResult>
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <bf:ResponseHeaderElement
    xmlns:bf="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
      <bf:sessionToken>3jPHTrTJSea2o76xiGM4K8fQuSE</bf:sessionToken>
    </bf:ResponseHeaderElement>
  </soapenv:Header>
  <soapenv:Body>
    <GetStructuredRelevanceResultResponse
    xmlns="http://schemas.bigfix.com/Relevance">
      <StructuredRelevanceResult>
        <results>
          <Integer>10697214</Integer>
          <Integer>14519782</Integer>
        </results>
        <plural>true</plural>
        <type>integer</type>
        <evaltime>24</evaltime>
      </StructuredRelevanceResult>
    </GetStructuredRelevanceResultResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

CreateUser

CreateUser is a Web Reports SOAP method that allows a Web Reports administrator to programmatically create new Web Reports users. Note that this method requires request and response headers to authenticate.

Table 8. Parameters

Name	Type	Occurrences	Description
user	UserAccount	1	The new Web Reports user account to be created.

Table 9. Return Values

Name	Type	Occurrences	Description
success	boolean	1	True if the user was successfully created, otherwise false.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
    xmlns:h="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.bigfix.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>bigfix</username>
      <password>bigfix</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <CreateUser xmlns="http://schemas.bigfix.com/Relevance">
      <user>
        <logonName>new_user</logonName>
        <fullName>Mr. New User</fullName>
        <password>new_password</password>
        <role>Normal</role>
      </user>
    </CreateUser>
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <bf:ResponseHeaderElement xmlns:bf="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
```

```
      <bf:sessionToken>uWGgqjb91IyheW7x+EPGMWERZiU</bf:sessionToken>
    </bf:ResponseHeaderElement>
  </soapenv:Header>
  <soapenv:Body>
    <CreateUserResponse xmlns="http://schemas.bigfix.com/Relevance">
      <success>true</success>
    </CreateUserResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

StoreSharedVariable

StoreSharedVariable is a SOAP method that inserts a variable into the dashboard datastore of one or more databases being aggregated by Web Reports. You can save data into the database by using a script function, and retrieve that data with a session inspector. Each dashboard has its own name space, so it can use common names (such as settings) without causing name collisions with another dashboard. In addition, variables can be flagged as private, in which case they are visible only to a particular user. So a private settings variable stores the settings for a particular dashboard for a particular user. A shared settings variable stores the settings for a particular dashboard and is shared by all users of that dashboard.

If dashboards must share data with each other, they can. The dashboard ID can be specified in both the script function for writing variable data and in the session inspectors used to access that data, although typically it is expected that the current dashboard inspector is used to supply that ID. Note that this method requires request and response headers to authenticate.

Table 10. Parameters

Name	Type	Occurs	Description
dashboardVariableIdentifier	DashboardVariableIdentifier	1	A dashboard ID, variable name, and optional database ID that identify the dashboard variable to be inserted.
variableValue	string	1	The value to insert for the variable.

Table 11. Return Values

Name	Type	Occurs	Description
success	boolean	1	True if variable was successfully inserted, otherwise false.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
    xmlns:h="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.bigfix.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>bigfix</username>
      <password>bigfix</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <StoreSharedVariable xmlns="http://schemas.bigfix.com/Relevance">
      <dashboardVariableIdentifier>
        <dashboardID>testID</dashboardID>
        <variableName>testVariable</variableName>
      </dashboardVariableIdentifier>
      <variableValue>testValue</variableValue>
    </StoreSharedVariable>
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <bf:ResponseHeaderElement
    xmlns:bf="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
      <bf:sessionToken>sPu1faAdFGLzqummbcy5ScTYMEE</bf:sessionToken>
    </bf:ResponseHeaderElement>
  </soapenv:Header>
  <soapenv:Body>
    <StoreSharedVariableResponse xmlns="http://schemas.bigfix.com/Relevance">
      <success>true</success>
    </StoreSharedVariableResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

DeleteSharedVariable

DeleteSharedVariable is a Web Reports SOAP method that deletes a variable from the dashboard datastore of one or more databases being aggregated by Web Reports. Note that this method requires request and response headers to authenticate.

Table 12. Parameters

Name	Type	Occurs	Description
dashboardVariableIdentifier	DashboardVariableIdentifier	1	A dashboard ID, variable name, and optional database ID that identify the dashboard variable to be deleted.

Table 13. Parameters

Name	Type	Occurs	Description
success	boolean	1	True if variable was successfully deleted, otherwise false.

Example request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
    xmlns:h="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.bigfix.com/Relevance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>bigfix</username>
      <password>bigfix</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <DeleteSharedVariable xmlns="http://schemas.bigfix.com/Relevance">
      <dashboardVariableIdentifier>
        <dashboardID>testID</dashboardID>
        <variableName>testVariable</variableName>
        <databaseID>2147485678</databaseID>
      </dashboardVariableIdentifier>
    </DeleteSharedVariable>
  </s:Body>
</s:Envelope>
```

Example response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <bf:ResponseHeaderElement xmlns:bf="http://schemas.bigfix.com/Relevance"
    xmlns:xsi="http://www.w3.org/2001/XML-Schema-instance">
      <bf:sessionToken>QVExtc5suqwQUfBcdw9+Ozs3Aio</bf:sessionToken>
    </bf:ResponseHeaderElement>
  </soapenv:Header>
  <soapenv:Body>
    <DeleteSharedVariableResponse xmlns="http://schemas.bigfix.com/Relevance">
      <success>true</success>
    </DeleteSharedVariableResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP examples

The following sample scripts help you understand how to access the SOAP API:

Shell script

This script is a simple Perl shell that allows you to retrieve a single relevance evaluation. This example requires the SOAP::Lite module, version 0.71 or later. It uses the GetRelevanceResult SOAP method.

```
use SOAP::Lite;
#arguments: [hostname] [username] [password] [relevance expression]
#hostname only, e.g. 'example.com' rather than 'http://example.com/webreports'
my $host = $ARGV[0];
my $username = SOAP::Data->name('username' => $ARGV[1] );
my $password = SOAP::Data->name('password' => $ARGV[2] );
my $expr = SOAP::Data->name('relevanceExpr' => $ARGV[3] );
my $service = SOAP::Lite -> uri( 'http://' . $host . '/webreports?wsdl' )
    -> proxy( $host );
my $result = $service -> GetRelevanceResult( $expr, $username, $password );
if( $result->fault ) {
 print "faultcode: " . $result->faultcode . "\n";
 print "faultstring: " . $result->faultstring . "\n";
}
else {
  foreach my $answer ( $result->valueof( "//GetRelevanceResultResponse/a" ) )  {
    print $answer . "\n";
  }
}
```

You pass four arguments to this script: host, username, password, and the relevance expression that you want to evaluate. The host name is something like example.com, denoting the location of your Web Reports server. After passing the parameters to the SOAP module, a call is made to GetRelevanceResult. If there is an error, it can be parsed here. Otherwise, the results of the relevance evaluation (typically plural) are printed in a loop.

Add the following line at the top of the file to help with debugging:

```
use SOAP::Lite +trace => 'debug';
```

The following line might be required to avoid a bug in SOAP:Lite

```
$SOAP::Constants::DO_NOT_CHECK_CONTENT_TYPE = 1;
```

Create user example

This code creates a new user. All the required information about the new user is embedded in the code:

```
use SOAP::Lite;

my $host = "http://localhost/webreports";

# The login credentials for an existing admin user
my $username = 'name';
my $password = 'password';

# The new user information
my $newUserLogin = 'new_admin';
my $newUserFullName = 'Mr. New User';
my $newUserPassword = 'new_password';
my $newUserRole = 'Administrator';

my $service = SOAP::Lite->uri( $host . '?wsdl' )->proxy( $host );
# this string includes embedded quotes by using backslash characters (\")
my $loginXML = "<h:RequestHeaderElement xsi:type=\"LoginHeader\" " .
" xmlns:h=\"http://schemas.example.com/webreports\
" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " .
" xmlns=\"http://schemas.example.com/webreports\
" xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\">" .
" <username>$username</username>" .
" <password>$password</password>" .
"</h:RequestHeaderElement>";

my $loginHeaders = SOAP::Header->type( 'xml' => $loginXML );

my $newUserInfo = SOAP::Data->name( 'user' => \SOAP::Data->value(
SOAP::Data->name( 'logonName' => $newUserLogin ),
SOAP::Data->name( 'fullName' => $newUserFullName ),
SOAP::Data->name( 'password' => $newUserPassword ),
SOAP::Data->name( 'role' => $newUserRole ) ) );

my $result = $service->CreateUser( $newUserInfo, $loginHeaders );
PrintResult( $result, "CreateUserResponse" );

sub PrintResult {
    my $result = shift;
    my $responseName = shift;

    if ( $result->fault ) {
        print "faultcode: " . $result->faultcode . "\n";
        print "faultstring: " . $result->faultstring . "\n";
    }
    else {
        print $result->result . "\n";
        foreach my $answer ( $result->valueof( "//$responseName/a" ) )
            { print $answer . "\n"; }
    }
}
```

SOAP error handling

Web Reports uses standard SOAP faults for most error handling. This includes malformed requests, invalid login credentials, and other unexpected errors. It also includes relevance errors from the GetRelevanceResult method, and notrelevance errors from the GetStructuredRelevanceResult method. Relevance errors from GetStructuredRelevanceResult are listed in the StructuredRelevanceResult object returned from the server. For more information, see GetStructuredRelevanceResult.

The following demonstrates a request that results in an “invalid user name or password” error:

Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <h:RequestHeaderElement xsi:type="LoginHeader"
  xmlns:h="http://schemas.example.com/Relevance"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://schemas.example.com/Relevance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <username>example</username>
      <password>wrong_password</password>
    </h:RequestHeaderElement>
  </s:Header>
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <GetStructuredRelevanceResult xmlns="http://schemas.example.com/Relevance">
      <relevanceExpr>now</relevanceExpr>
    </GetStructuredRelevanceResult>
  </s:Body>
</s:Envelope>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <env:Fault>
      <faultcode>env:Client</faultcode>
      <faultstring xml:lang="ENU">Invalid username or password.</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Chapter 7. Creating HTML with inspectors

There are several inspectors that allow you to more easily create HTML content. The following sections describe how you can use them with the APIs to create custom reports.

HTML inspectors

There are several inspectors that facilitate the generation of HTML text from string literals and property results. These inspectors have the type *html* that allows special characters, such as angle brackets and ampersands, to be properly escaped. Neglecting to escape these characters when you output text that is based on user input or database content can lead to cross-site scripting vulnerabilities. Using these inspectors, you can safely format HTML strings with any regular query. For example:

```
<?relevance name of company?>
```

If *name of company* is be Big&Bad, the processing instruction is replaced in the HTML with the properly escaped string:

```
Big&amp;Bad
```

You can include an "as html" cast explicitly to ensure that special characters print properly, rather than being interpreted as markup:

```
<?relevance "<h1>Heading</h1>" as html?>
```

This expression returns:

```
"&lt;h1&gt;Heading&lt;/h1&gt;"
```

There are some situations where escaping reserved characters is not appropriate. The most common cases are where you have a literal HTML string, or properties whose type is string but that already produce appropriately formatted HTML. In these cases, you can use the html indexed property:

```
<?relevance html "<h1>Heading</h1>"?>
```

This phrase results in:

```
"<h1>Heading</h1>"
```

Depending on how the dashboard is used, try to avoid the use of the html indexed property that potentially allows a script insertion attack. As an alternative, consider creating your own expressions by concatenating strings and html. For example, the following two expressions return the same result:

```
<?relevance html "<h1>" & name of company & html "</h1>"?>
<?relevance concatenation of (html "<h1>"; name of company as html; html "</h1>")?>
```

These lines return the following:

```
"<h1>Big&amp;Bad</h1>"
```

Because items in a list must all have the same type, the following does not work:

```
<?relevance concatenation of (html "<h1>"; name of company; html "</h1>")?>
```

This code chunk produces the error: *Incompatible types (html and string)* because the company name was not cast as html.

HTML tag inspectors

Another way to avoid using the html indexed property is to use the html tag inspectors:

```
<?relevance html tag "h1" of name of company?>
```

that returns:

```
"<h1>Big&amp;Bad</h1>"
```

The html tag embeds the requested text in the specified HTML tag. The requested text can be either a string or html. If it is a string, it is HTML-escaped. The index parameter can also include attributes that are separated from the tag name by whitespace:

```
<?relevance html tag "h1 id='companyName'" of name of company?>
```

that returns:

```
"<h1 id='companyName'>Big&amp;Bad</h1>"
```

You can also nest tags:

```
<?relevance html tag "div id='header'" of html tag "h1" of name of company?>
```

That returns:

```
"<div id='header'><h1>Big&amp;Bad</h1></div>"
```

Most common HTML elements have a shorthand tag property:

```
<?relevance h1 of name of company?>
```

That returns:

```
"<h1>Big&amp;Bad</h1>"
```

Like the generic html tag inspector, each shorthand tag property can embed both strings and html. The shorthand tags also accept HTML attributes:

```
<?relevance h1 "id='companyName' class='header'" of name of company?>
```

That returns:

```
"<h1 id='companyName' class='header'>Big&amp;Bad</h1>"
```

The following shorthand tags are supported:

```
html     head     title   meta     body    div
span     address  h1      h2       h3      h4
h5       h6       em      strong   dfn     code
samp     kbd      var     cite     abbr    acronym
blockquote q      sub     sup      p       pre
ins      del      ul      ol       li      dt
dd       table    caption thead    tfoot   tbody
colgroup col      tr      th       td      link
base     tt       i       b        big     small
```

Because “a” is ignored by the relevance evaluator, the “a” shorthand property is represented by “anchor”.

```
<?relevance anchor "href='http://www.bigfix.com'" of "BigFix"?>
```

That returns:

```
"<a href='http://www.bigfix.com'>BigFix</a>"
```

There are a few special-purpose aggregating properties:

- Ordered list
- Unordered list
- Definition list

These produce HTML lists (of the respective types) of their plural string or html direct object. For example an ordered (numbered) list can be specified like this:

```
<?relevance ordered list of ("<"; ">"; "&")?>
```

That returns:

```
"<ol><li>&lt;</li><li>&gt;</li><li>&amp;</li></ol>"
```

A simple bulleted list can be specified like this:

```
<?relevance unordered list of ("<"; ">"; "&")?>
```

That returns:

```
"<ul><li>&lt;</li><li>&gt;</li><li>&amp;</li></ul>"
```

Definition lists alternate between *dt* and *dd* elements. They are used where you have a natural set of name-value pairs:

```
<?relevance definition list of (name of it; free space of it as string) of
drives whose (exists free space of it)?>
```

That returns:

```
"<dl><dt>C:</dt><dd>32183602176</dd><dt>G:</dt><dd>4845355008</dd></dl>"
```

Issues with the "it" statement

Throughout this guide you have seen several examples of the “it” statement, such as this one:

```
<?relevance definition list of (name of it; free space of it as string) of
drives whose (exists free space of it)?>
```

In this example, if you also want to use “it” to refer to attributes, you encounter a problem because "it" in the html tag parameter must refer to the clause that defines the tag contents. In the following example, it refers to statementA, not statementB:

```
(html tag ("tag value='" & name of it & "'") of statementA) of statementB
```

The html inspectors provide an alternative syntax to avoid this problem:

```
html tag (<name>, <contents>)
html tag (<name>, attr list of (<name1>, <value1>; ...; <nameN>, <valueN>), <contents>)
```

Where <name> is a string, <contents> is a string or html, and names and values are strings. Use this syntax to form complex html statements without the use of “it”:

```
html tag ("h1", attr list of ("id", "Big&Bad"), "Big&Bad")
```

That returns:

```
"<h1 id="Big&amp;Bad">Big&amp;Bad</h1>"
```

Or this line:

```
html tag ( "p", attr list of (("class", "myclass"); ("align", "left")),
 html "html <i>italic snippet</i>" )
```

That returns:

```
<p class="myclass" align="left">html <i>italic snippet</i></p>
```

Introducing datastore inspectors

Dashboards have access to a persistent data store allowing them to remember configuration options. You can get a good overview of these datastore inspectors by evaluating the following relevance expression in the Presentation debugger:

```
properties whose (direct object type of it as string starts with "bes")
```

These keywords are also referred to as Session inspectors. Some of the basic types are:

- BES Fixlet
- BES action
- BES property
- BES computer

The names of these types double as global iterated properties. For example:

```
bes properties
```

This inspector returns a list of all property objects.

```
bes computers
```

This returns a list of all computers that can be administered by the current user. For a complete guide to these inspectors, see the *Session Inspector Library.*

Creating statistical properties

There are two primary methods of getting statistical properties into a deployment:

- Import an analysis that contains properties with the *KeepStatistics* attribute set to true.
- Author an analysis in a Fixlet site by using hand-edited action script MIME. Add the header "X-Keep-Statistics: true" to the property headers.

Properties must be one of the following types to keep statistics:

- Integer
- Boolean
- Floating point

Attempts to keep statistics on properties with other types are ignored.

Plural properties are acceptable, if used carefully. For example, "free spaces of drives" results in statistics about all drives on all computers.

Linking to other documents

You can use the link inspector to create an HTML hyperlink that opens the appropriate document window for any specified Fixlet, computer, action, or user. The link inspector typically uses the name or the ID of the object to identify it:

```
link of bes fixlets whose (id of it is 17)
```

This inspector resolves the Fixlet ID and then generates a tag of the form:

```
<A href="linkid:openfixlet(1,17)">Name Of Fixlet</A>
```

Where "Name Of Fixlet" is the name-string that is retrieved for the specified linked Fixlet and the number pair specifies the site number and the Fixlet number.

You can also specify your own custom content for the anchor tag by using the indexed form of the inspector:

```
link "Click Here" of bes fixlets whose (id of it is 17)
```

or

```
link (b of "Click Here") of bes fixlets whose (id of it is 17)
```

These commands generate links of the form:

```
<A href="linkid:openfixlet(1,17)">Click Here</A>
<A href="linkid:openfixlet(1,17)"><b>Click Here</b></A>
```

In a dashboard or wizard, click the link to open the associated MDI document. For example:

```
link of bes wizards whose (name of it = "AntiPest")
```

This code generates an anchor of the form:

```
<A href="linkid:openWizard(5211,bfantipest_dashboard.ojo)">TEM
AntiPest</A>
```

This HTML link, when clicked, opens the AntiPest dashboard. Similarly, in Web Reports, the link you create opens a report page for the object.

The link ID protocol

Several of the previous examples illustrated anchor links by using the linkid: protocol. You might want to create a link such as this directly. This style of linking is typically used in the Navigation bar, but it is also available for dashboards and wizards. It takes the form:

```
linkid:openfixlet(siteid,fixletid)linkid:openaction(actionid)
linkid:opencomputer(computerid)
linkid:openuser(username)
linkid:openunmanagedasset(unmanagedassetid)
```

Linkids open the specified object in an MDI document window. However, the IDs of Fixlets, actions and other objects are rarely predictable, so it is problematic to specify them as literals. In addition, linkids do not work in Web Reports. To circumvent these limitations, use the link href inspector:

```
link href of bes fixlets whose (id of it is 17)
```

This code generates a corresponding linkid for dashboards and wizards:

```
linkid:openfixlet(1,17)
```

Here, the first number refers to the site and the second number (17) refers to Fixlet. Note that the direct linkid protocol does not work for Web Reports. Instead, use the link href inspector that in the context of a Web Report, generates an ordinary HTML link. This technique is recommended for writing portable scripts that can run in both environments without rewriting.

There is also a linkid for the Tivoli Endpoint Manager support site:

```
linkid:supportSite
```

Clicking this link opens the Tivoli Endpoint Manager support website in a new application window with the default browser.

A list of link IDs

This is a table of some useful link IDs:

Table 14.

Link Name	Dialog invoked
fixletMessages	mainTabDialog->ShowAllRelevantFixlets();
fixletActions	mainTabDialog->ShowAllFixletActions();
customFixlets	mainFrame.OnToolsCreateNewFixletMessage();
viewTasks	mainTabDialog->ShowAllRelevantTasks();
taskActions	mainTabDialog->ShowAllTaskActions();
customTask	mainFrame.OnToolsCreateNewTaskMessage();
customBaseline	mainFrame.OnToolsCreateNewBaseline();
customActions	mainFrame.OnToolsTakeCustomAction();
allComputers	mainTabDialog->OnViewComputerList();
viewAnalyses	mainTabDialog->OnViewAnalysesList();
customAnalyses	mainFrame.OnToolsCreateCustomAnalysis();
ConsoleOperators	mainTabDialog->OnViewUserList();
unmanagedAssets	mainTabDialog->OnViewUnmanagedAssetsList();
visualization	mainFrame.OnToolsVisualization();
webReports	mainFrame.OnToolsViewwebreports();
clientSettingsTasks	mainFrame.GetMainTabDialog()->OnToolsEditClientSettings();
serverRelaySettingsTasks	mainFrame.GetMainTabDialog()->OnToolsEditServerRelaySettings();
onlineHelp	mainFrame.OnHelpContents();
manageSiteSubscriptions	mainFrame.OnToolsManagesites();
editRetrievedProperties	mainFrame.OnToolsCustomProperties();

Refreshing relevance

Ideally, dashboards and wizards in the console should be updated as new information arrives from the database. To make <?relevance ?> instructions automatically update, you must specify another pair of processing instructions to enclose the time-sensitive block of your document:

```
<?BeginRefreshRelevance?>
<?EndRefreshRelevance?>
```

These tags cause every <?relevance ?> tag that is contained between them to be reevaluated every time something in the console database changes.

The actual implementation of this update is important because it can affect the way that you code your HTML. During execution, the <?BeginRefreshRelevance?>

tag is replaced by a <span> tag, and the <?EndRefreshRelevance?> tag is replaced by a </span> tag. When the console detects that one of the <?relevance ?> tags is changed, it updates the entire section of the document by replacing the contents of the <span> tag. You must therefore make sure that these relevance tags do not interrupt any existing <span> tags.

To correctly identify which <span> must be updated, the console assigns an ID attribute to the <span> tag that it generates to replace the <?BeginRefreshRelevance?> tag. By default, that ID is __DRRSN (an acronym for Default Refresh Relevance Section Name).

If necessary, you can specify a different ID in the refresh tags like the following code:

```
<?BeginRefreshRelevance id="MyRefreshSpan"?>
<?EndRefreshRelevance id="MyRefreshSpan"?>
```

The IDs must match exactly. Specifying your own IDs gives you a way to nest RefreshRelevance tags.

By default, anything that changes a Relevance evaluation triggers a refresh of the code block. However, you can specify which changes must trigger a refresh as well as a minimum time interval to wait. For example:

```
<?BeginRefreshRelevance ActionResults="00:01:00" ?>
 <?relevance (link of it & "(" & (number of results of it as string) & ")" & br)
  of bes actions whose (state of it is "Open") ?>
<?EndRefreshRelevance ?>
```

This example displays a list of all the open actions as links that you can click to open the associated action document. Next to the link, the number of results for each action is displayed in parentheses. The number indicates how many of the targeted computers reported on the action.

If the action results are not changing, this block of code is static. However, when a change occurs, the "00:01:00" value for ActionResults specifies that this block is refreshed at most one time per minute. The form of these values is the standard TimeInterval string format "hh:mm:ss". This is a list of the built-in refresh triggers:

Table 15.

Trigger Type	**Refreshes Whenever...**
Computers	A computer is added or removed (ComputerDataStore)
ReportTimes	A computer's last report time changes
ExternalContent	External Fixlet site content changes (FixletStore)
CustomContent	Custom content changes, not including actions (ActionSiteStore)
Actions	Actions are taken, stopped, restarted, and so on. (ActionStore)
ActionResults	A client reports on the status of an action (ActionResultStore)
FixletResults	A client reports on the relevance of a Fixlet (FixletResultStore)
PropertyResults	A client reports a new value for a retrieved property (RPResultStore)
RefreshCycle	See the following text
ManualRefresh	See the following text

Refreshes are done only at the end of each refresh cycle, not when the change is first detected. At the end of a cycle, if any of your specified types are triggered and its time interval is expired, then a refresh occurs. For more frequent updating, the *RefreshCycle* attribute can be used to force a refresh at the end of the refresh cycle, regardless of any changes.

You can also create blocks that can be refreshed manually by using the *ManualRefresh* attribute. It works in combination with a predefined script that takes the ID of the Refresh block as an argument, for example:

```
ManualRefresh("Clock")
```

This works to refresh a code block with an id="Clock":

```
<?BeginRefreshRelevance id="Clock" ManualRefresh="00:00:00" ?>
 <P> The current time is: <?relevance now ?> </P>
<?EndRefreshRelevance id="Clock" ?>
<P> <Button onclick='ManualRefresh("Clock")'>Refresh</Button> </P>
```

You can also call the script function with no parameters:

```
ManualRefresh()
ManualRefresh("")
```

This code refreshes the default unnamed refresh block. To refresh all the blocks, use:

```
ManualRefreshAll()
```

You can set multiple clocks to satisfy different refresh needs:

```
<?BeginRefreshRelevance ManualRefresh="00:00:00"?>
 <?relevance now ?>
<?EndRefreshRelevance?>
<?BeginRefreshRelevance id="Foo" ManualRefresh="00:00:00"?>
 <?relevance now ?>
<?EndRefreshRelevance id="Foo"?>
<P> <Button onclick='ManualRefresh()'>Refresh first clock</Button> </P>
<P> <Button onclick='ManualRefresh("Foo")'>Refresh second clock</Button> </P>
<P> <Button onclick='ManualRefreshAll()'>Refresh all clocks</Button> </P>
```

Note: Inside the refresh block, all the Relevance expressions are replaced by their results. This means that any ManualRefresh calls you place inside the block are also replaced, invalidating your code. Therefore, always put the actual call outside the refresh block.

You can also associate refresh operations with a JavaScript by using *RegisterRefreshHandler*. You can use the RegisterRefreshHandler function in both the console and Web Reports, but in Web Reports it is ignored, and your handler is never called. Because a Web Report is static, it does not support refreshing relevance.

Appendix. Support

For more information about this product, see the following resources:

- http://pic.dhe.ibm.com/infocenter/tivihelp/v26r1/topic/com.ibm.tem.doc_9.1/welcome/welcome.html
- IBM Endpoint Manager Support site
- IBM Endpoint Manager wiki
- Knowledge Base
- Forums and Communities

Notices

This information was developed for products and services that are offered in the USA.

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

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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

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

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

Programming interface information

Trademarks

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

Rights

Printed in USA