



BigFix Client Compliance API

Version 1.4

October 28, 2004

dennis_goodrow@bigfix.com

The Compliance API allows BigFix partners and integrators to expose the results of an endpoint inspection conducted by the BigFix Enterprise Suite (BES) Client to their own logic embedded in 3rd-party applications and clients executing on the client machine.

The results are calculated in the execution environment of the BESClient, which typically has elevated privileges and access rights.

The values exposed by this interface can be calculated using the complete set of inspectors defined in the BigFix inspection framework. This provides a mechanism to look for extremely complex conditions and gives a commanding view of the machine.

The BigFix Compliance API makes explicit the data that can move from the BESClient execution environment to any other account that can execute applications invoking it. It is the responsibility of the developer using the API, to limit the information exposed through this interface.

In order for the information to become available to the API, a BES console user in charge of the managed machine (or a 3rd-party management infrastructure directly or through the BigFix Platform API), must deploy the 3rd-party application or client and the configuration that defines the set of data that is exposed through the interface.

API Terminology –

Compliance Document – A file containing one or more compliance expression items. The document is formatted as XML. The BESClient 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 BESClient after it has processed the compliance document. The response is composed of one or more Compliance Result items.

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

Designator – Up to 64 characters in length, composed of alphanumeric, underscore, or period.

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 appearing 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 concerning the Compliance expression item.

Comment – handled the same as the Description.

Possible use cases

The results of evaluation can be used by access control software to observe the compliance state of the machine, 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 BigFix inspection framework.

Compliance might be determined based on detecting that certain conditions do or do not exist on the machine. Examples include:

- * Detecting if spyware is installed or running. This could 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 are not installed.
- * Detecting if a virus scanner is installed, running and properly configured. This could take the form of a compliance policy that requires that one of a group of virus scanners be installed and running.
- * Detecting if a firewall is installed, running and properly configured. This could take the form of a compliance policy that requires that one of a group of firewall applications is installed and running.
- * Detecting that network shares are turned off. This could take the form of requiring that no network shares be defined on the machine in order for the machine to be in compliance.
- * Detecting that wireless networks are disabled. This could take the form of requiring that wireless networks be turned off during corporate LAN access.
- * Detecting the patch level of the machine. The API will allow one to check if there any 'critical' patches that need to be installed. In version 4.1.8 of the client, the following inspectors were

added to make this possible

- * sites have two new iterators
 - fixlets of <site>
 - relevant fixlets of <site>
- * fixlets have the following properties
 - id of <fixlet>
 - relevance of <fixlet>
 - header "xyz" of <fixlet>
- * headers have the following properties
 - name of <header>
 - value of <header>

Here is an example compliance expression that returns true when there are no critical Fixlet Messages that are relevant on the machine:

```
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 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', and program your application based on the results of evaluating the expression and the contents of your comments.

Possible applications include watchdog software that is configurable after it ships to look for certain conditions and disables, limits, or enables

- network shares
- wireless networks
- network access

From development through deployment through enhancement

These are the steps that are involved:

- Developer builds application and installer that requires relevance results. Installer performs all installation steps to ensure that the API is configured properly (see Runtime Requirements below).
- Developer builds one or more Compliance documents defining designators, relevance expressions, descriptions, and comments.
- Developer deploys Fixlet messages that deploy the installer and Compliance documents. The Fixlet contains an action that runs the installer and copies the compliance documents to the BESClient compliance document store.
- BES Operator uses the action to deploy the application and Compliance documents using BES.
- Application uses API to collect response from BESClient.
- Application uses its own logic to made decisions based on response.
- If developer wants to update his application and/or compliance documents, he uses Fixlet messages to ask Operators to deploy them.

API Development Kit - Contents

The Compliance API SDK contains the following files:

Filename	Description
ClientComplianceAPI.pdf	This file, containing a description of the API, how it might be used, and sample content and source code.
ClientCompliance.h ClientComplianceMain.cpp ClientCompliance.vcproj ClientComplianceMT.lib ClientComplianceST.lib ClientCompliance.exe	C++ sample source and project files to build a self contained application. (Does not require BESClientComplianceMod.dll COM module be installed). Does not require COM or windows scripting be installed.
BESClientComplianceMod.dll test.vbs ComplianceDumpToReg.vbs	COM module containing objects and interfaces to drive the API. Use this in concert with either of the visual basic samples, and the windows script engine to perform a compliance evaluation.
ComplianceDoc.xml	Sample compliance rules document.
BigFix Client Compliance Configuration.efxm	Masthead file for BigFix Fixlets to install and configure the Client Compliance API – automates many of the configuration steps for deploying the API on a BigFix managed computer

The BigFix Client Compliance Configuration Fixlet site automates much of the manual configuration outlined below. Use the “BigFix Client Compliance Configuration.efxm” masthead to install the site into a BigFix deployment.

Runtime Requirements

Applications that use the API will need the components installed and configured. This is accomplished manually as follows:

Install Step	Description
Install COM Module	This can be accomplished using the following command: Regsvr32.exe BESClientComplianceMOD.dll [Only required for COM implementations]
Create Configuration Key	Create the following registry key: • HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\ClientComplianceAPI
Create configuration data	Define these three string values in the configuration registry key: • RequestDir – fullpath where configuration requests are written • ResponseDir – fullpath where configuration responses are written • ConnectDir – fullpath where interface glue scripts and executables are located (by convention)
Create ResponseDir	Create the directory where response files will be written by the API. The protections on this directory must be such that • Anyone may read files in this directory, • The BESClient will write files to this directory.
Create RequestDir	Create the directory where request files will be written by the API. The protections on this directory must be such that • Anyone may write files in this directory • The BESClient will read and delete files in this directory
Create ConnectDir	Create the directory where “glue” scripts and executables that provide the interface between the Client Compliance API and third-party clients are kept (by convention). The protections on this directory must be such that • Third-party clients can execute files in this directory
BESClient.exe	The BESClient should be installed on the endpoint. The version of the client should be at least 4.1.8.

COM Interface– Module and Classes Specifications

The API is delivered as a COM module registering as `BESClientComplianceMod`. The module provides three classes:

- `Session`
- `Progress`
- `Response`

BESClientComplianceMod.Session Object -

This object is used 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`.

Creation	This object is registered for COM creation using 'BESClientComplianceMod.Session'.
Methods	
<code>long Open(BSTR siteURL, BSTR rulesXML, long flags)</code>	Attempts to open a compliance evaluation session with the <code>BESClient</code> . The return value is 0 when an evaluation has been successfully begun. A negative number is returned when this cannot be done. The return value is one of the integer constants whose name begins with <code>COMPLIANCE_ERROR</code> in the API constants. The <code>BESClient</code> will process the compliance document named <code><rulesXML></code> in the <code>__Compliance</code> subfolder of the site data folder of the site whose gather url matches the <code><siteURL></code> .
<code>long Close()</code>	Closes the session. Call this property when you have completed using the results of the evaluation. The return value is 0 on success, or one of the integer constants whose name begins with <code>COMPLIANCE_ERROR</code> in the API Constants.

BESClientComplianceMod.Progress Object -

This object is used to collect the progress of the Open session from the BESClient.

Creation	This object is registered for COM creation using 'BESClientComplianceMod.Progress'.						
Methods							
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: <table><tr><td>COMPLIANCE_PROGRESS_COMPLETE</td><td>1</td></tr><tr><td>COMPLIANCE_PROGRESS_ERROR</td><td>2</td></tr><tr><td>COMPLIANCE_PROGRESS_BUSY</td><td>3</td></tr></table> When COMPLIANCE_PROGRESS_COMPLETE is returned, the results of the compliance check are available. Use the Result object to collect the results.	COMPLIANCE_PROGRESS_COMPLETE	1	COMPLIANCE_PROGRESS_ERROR	2	COMPLIANCE_PROGRESS_BUSY	3
COMPLIANCE_PROGRESS_COMPLETE	1						
COMPLIANCE_PROGRESS_ERROR	2						
COMPLIANCE_PROGRESS_BUSY	3						
Properties							
long GetPercent()	When COMPLIANCE_PROGRESS_BUSY is returned from SessionProgress, the GetPercent property may be accessed to obtain the current estimate of percent completion.						
long GetError()	When COMPLIANCE_PROGRESS_ERROR is returned from SessionProgress, the GetError property may be accessed to obtain the current error. The error codes are defined in the API Constants part of this document.						

BESClientComplianceMod.Response Object -

This object is used to examine the compliance response from BESClient. This object will return results if the session is still open, after a call to Progress.SessionProgress() returns

COMPLIANCE_PROGRESS_COMPLETE status, and until Session.Close() is called.

Creation	Registered for COM creation using 'BESClientComplianceMod.Response'.
Properties	
long ResultCount()	Returns the number of results in the compliance response.
BSTR GetDescription(long index)	Collects the Designator for the result at this index position.
BSTR GetRelevanceResult(long index)	Collects the RelevanceResult for the result at this index position.
BSTR GetDescription(long index)	Collects the Description for the result at this index position.
BSTR GetComment(long index)	Collects the Comment for the result at this index position.

API Constants and Error codes

The following numeric constants are used by API. These are found in the ClientCompliance.h file accompanying this document.

COMPLIANCE_FLAG

The Session.Open method takes a flag argument that is 0 or any of the following bit values

COMPLIANCE_FLAG_GATHER = 1

Gather from all sites before processing rules. Note that gathering may fail depending on network connectivity etc. You can build compliance rules that look at the results of 'exists site whose(now - last gather time of it > day)'

Without this flag enabled, rules processing will not be preceded by a gathering operation.

COMPLIANCE_FLAG_EVALUATE_ALL = 2

Perform a complete evaluation of all fixlets before processing rules.

Without this flag enabled, rules processing will not be preceded by an evaluation pass. Rather, the last computed relevance of any fixlet inspection will be available to rules processing.

COMPLIANCE_PROGRESS

The following values are returned by the Progress.SessionProgress call.

COMPLIANCE_PROGRESS_COMPLETE = 1

The rules document has been processed and may be accessed using the Response object methods. These results will no longer be available when the Session.Close method is called.

COMPLIANCE_PROGRESS_ERROR = 2

An error occurred while processing the rules. This is normal behavior in some cases. An error code of COMPLIANCE_ERROR_CHANGING_CONDITIONS, occurs when the client detects that Fixlet content has changed during the session. In this case, your application should close the existing session and open another session until the client can process the compliance document in a more stable environment. Collect the particular error condition using the Progress.GetError method.

COMPLIANCE_PROGRESS_BUSY = 3

Gathering, Fixlet evaluation, and rule processing are not yet completed. Collect an estimate of the percentage completion from the Progress.GetPercent method.

COMPLIANCE_ERROR

The following error codes are returned

- from Session.Open, Session.Close,
- from Progress.GetError after Progress.SessionState has returned COMPLIANCE_PROGRESS_ERROR, and
- these error codes are placed into the script 'Err' object when the API Response methods are used after the Session has been closed.

COMPLIANCE_ERROR_NOTINSTALLED = -1

Returned if the BESClient is not installed.

COMPLIANCE_ERROR_NOTRUNNING = -2

Returned if the BESClient is not currently executing.

COMPLIANCE_ERROR_NOTRESPONDING = -3

Returned if the BESClient is not responding to compliance requests.

COMPLIANCE_ERROR_NOTCONFIGURED = -4

Returned if the registry is not configured with RequestDir and ResponseDir settings

COMPLIANCE_ERROR_EXISTINGREQUEST = -5

Returned if the BESClient is not picking up requests.

COMPLIANCE_ERROR_API_BUSY = -6

Returned if another session is open. Only one session is allowed at a time.

COMPLIANCE_ERROR_RESPONSE_MALFORMED = -7

The BESClient response was not formatted properly.

COMPLIANCE_ERROR_RESPONSE_CLOSED = -8

Returned if you try to access a response value but you have already closed the session.

COMPLIANCE_ERROR_OLD_VERSION = -9

This error code will be returned when the API determines that a version of the client on the machine does not support the client api.

COMPLIANCE_ERROR_RESPONSE_READING = -10

This error occurs when the client generates unexpected output while processing the rules.

COMPLIANCE_ERROR_DOC_READING = -11

This error occurs when the API cannot parse the output produced from the client.

COMPLIANCE_ERROR_DOC_SITE = -13

This error occurs when the url argument of the Session.Open call does not match any site to which the client is currently subscribed.

COMPLIANCE_ERROR_DOC_MISSING = -14

This error occurs when the doc argument of the Session.Open call does not exist in the __Compliance folder of the fixlet site matching the url argument.

COMPLIANCE_ERROR_PENDING_CLOSE = -15

This error occurs if you attempt to call the Session.Open method and you have not closed the previous session with a Session.Close call.

COMPLIANCE_ERROR_WRITING_REQUEST = -17

This error occurs when the API is unable to write the request to the client.

COMPLIANCE_ERROR_NO_RESPONSE = -18

This error occurs if the BESClient fails to respond within 10 seconds of the request.

COMPLIANCE_ERROR_COMPLIANCE_IO_FAILURE = -19

This error occurs if the BESClient encounters an IO error reading or writing the compliance document or intermediate state files to the client computer's file system.

COMPLIANCE_ERROR_INDEX_TOO_LARGE = -20

You will get this error in the script Err object, when you attempt to access a response whose index is larger than the number supplied by the client.

COMPLIANCE_ERROR_UNEXPECTED = -21

This error frequently occurs if a document or path cannot be found. Check file names and directory paths for the compliance document , RequestDir and ResponseDir as possible causes.

COMPLIANCE_ERROR_CHANGING_CONDITIONS = -22

This error code will be returned when the client detects that fixlet relevance is changing while the rules are being processed. The session should be closed and another session started until conditions stop changing and the results become stable.

COMPLIANCE_ERROR_DIR_MISSING = -23

This error code is returned when the client cannot open the compliance document because the directory in which the compliance document is located, cannot be found. The second argument to Session.Open should be a file name, not a path.

COMPLIANCE_ERROR_FILE_LOCKED = -24

This error code will be returned when the client cannot open the compliance document because it is locked.

COMPLIANCE_ERROR_FILE_PERMISSIONS = -25

This error code is returned when the client does not have permission to read the file.

Sample compliance document and vbscript

This sample assumes the following setup:

- BESClient version 4.1.8 or greater is running and subscribed to a site with gather url
 - `http://sync.bigfix.com/cgi-bin/bfgather/example`
- A file named ComplianceDoc.xml exists in the site data folder for site <site_example>
`C:\Program Files\BigFix Enterprise\BESClient\__BESData\<site_example>\__Compliance`
- BESClientComplianceMod.dll is installed and registered
- The registry is configured with appropriate paths to RequestDir and ResponseDir
- RequestDir and ResponseDir are existing folders

Sample Compliance rules document - ComplianceDoc.xml

ComplianceDoc.xml accompanies this document, and is a plain text file prepared using a simple text editor. It contains a more extensive example than the following, and follows the conventions of compliance documents, namely, a simple xml formatted document such as this:

```
<?xml version="1.0"?>
<BESClientComplianceDocument Version="1.0">
  <Date>13 Jun 2004 13:41:57 -0700</Date>
  <ComplianceItem>
    <Designator>Windows</Designator>
    <Expression>name of operating system starts with "Win"</Expression>
    <Description>You need Windows to be compliant</Description>
    <Comment>Compliant if True</Comment>
  </ComplianceItem>
  <ComplianceItem>
    <Designator>BESRunning</Designator>
    <Expression>exists running application "BESClient.exe"</Expression>
    <Description>The BESClient must be running.</Description>
    <Comment>Compliant if True</Comment>
  </ComplianceItem>
  <ComplianceItem>
    <Designator>AppList</Designator>
    <Expression>running applications</Expression>
    <Description>List of running applications.</Description>
    <Comment>Information only</Comment>
  </ComplianceItem>
</BESClientComplianceDocument>
```

Sample Script – test.vbs

The code below is contained in the test.vbs file accompanying this document. It demonstrates how you might use the COM interface with the above given situation.

You might execute it from a windows command console using the following command:

```
cscript test.vbs
```

Note that before running it, you must edit it and set the url, doc, and flags variables corresponding to your installation before it will work properly.

```
'
' Sample vbscript to show how to use the BESClientComplianceMod objects
' BigFix, 2004

Set session = WScript.CreateObject("BESClientComplianceMod.Session")
Set progress = WScript.CreateObject("BESClientComplianceMod.Progress")
Set response = WScript.CreateObject("BESClientComplianceMod.Response")

'See ClientCompliance.h for numeric constants
COMPLIANCE_PROGRESS_COMPLETE = 1
COMPLIANCE_PROGRESS_ERROR = 2
COMPLIANCE_PROGRESS_BUSY = 3

COMPLIANCE_FLAG_GATHER = 1
COMPLIANCE_FLAG_EVALUATE_ALL = 2

COMPLIANCE_ERROR_CHANGING_CONDITIONS = -22

On Error Resume Next
Err.Clear

'set url, doc, & flags as required using this format:

'url = "http://server.domain.com:52311/cgi-bin/bfgather.exe/actionsite"
'doc = "ComplianceDoc.xml"

flags = 0 'COMPLIANCE_FLAG_GATHER + COMPLIANCE_FLAG_EVALUATE_ALL

If url = "" Or doc = "" Then

    WScript.echo "You must set url, doc, and flags in this script"

Else

    t = session.Open(url, doc, flags)

    If t <> 0 Then

        WScript.echo "Open Error: " + FormatNumber(t)
```

```

Else

    ' To free up resource, make sure to perform session.Close
    ' after session completes and you are done looking at results

    Dim compliance_progress

    Do
        compliance_progress = progress.SessionProgress

        If compliance_progress = COMPLIANCE_PROGRESS_BUSY Then

            percent = progress.GetPercent
            WScript.echo "Percent: " + FormatNumber(percent)
            WScript.Sleep(10)

        End If

    Loop While compliance_progress = COMPLIANCE_PROGRESS_BUSY

    If compliance_progress = COMPLIANCE_PROGRESS_COMPLETE Then

        count = response.ResultCount
        WScript.echo "Count is " + FormatNumber(count)

        For i = 1 to count

            d = response.GetDesignator(i)
            r = response.GetRelevanceResult(i)
            x = response.GetDescription(i)
            c = response.GetComment(i)

            WScript.echo "D: " +d+ "  R: " +r+ "  X: " +x+ "  C: " +c

        Next

    ElseIf compliance_progress = COMPLIANCE_PROGRESS_ERROR Then

        error = progress.GetError

        If error = COMPLIANCE_ERROR_CHANGING_CONDITIONS Then
            WScript.echo "Client reports not available right now."
        Else
            WScript.echo "Error: " + FormatNumber(error)
        End If

    End If

    session.Close

End If
End If

```

Sample c++ source code and project

The SDK contains a set of files that comprise a c++ sample project:

SDK filename	Contents
ClientCompliance.h	Header file defining all the constants and c style entry points provided by the .lib files
ClientComplianceMain.cpp	Console main program source code. Simple example shows how to use the entry points to perform a compliance check.
ClientCompliance.vcproj	Project files necessary to build the sample project.
ClientCompliance.exe	Resulting executable produced by above project.
ClientComplianceMT.lib	Multi-threaded static library containing api entry points.
ClientComplianceST.lib	Single-threaded static library containing api entry points.

This project was built using Microsoft Visual C++ .NET. It statically links against the ClientCompliance library, to produce a self contained compliance test. You statically link this library into your own projects.

The static library exposes 5 c style functions that can be used to perform compliance evaluation. These function definitions and all the manifest constants used by the API are defined in the file ClientCompliance.h. The required flow of execution between these functions is to open, check progress until completion, examine results, and finally to close. The sample source file ClientComplianceMain.cpp demonstrates proper usage.

```
// 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 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_*

```