BES Server Plugin Service Proposal

From BigFix Wiki

Contents

Overview

As many of our product domains have found it neccessary to install persistent processes/services on the BES server, we are proposing to create a framework for creating plugins that need to run on the BES server, in order to make this process simpler and simplify configuration and mangagement of these services.

Background

- We have found that multiple Fixlet sites require server components to work properly
- Examples
    - Asset Discovery Importer
    - CPM Updater
    - Power Management WoL Scheduler
    - Power Management Medic
    - Power Management Store Historical Reports
    - IBM Integration?
    - Others?
- The nature of these add-on server components often require database read/write access and sometimes also SOAP access
- Several problems appear when adding these services:
    - The services tend not to handle database connections properly
    - These database problems are shared amongst the BigFix Server components (the server installer handles these issues), but for every service that we install, we need to re-implement the solution to these problems
    - The services do not come with installers/uninstallers that are easily used and they are hard to manage.
    - The services look ugly to customers
    - SOAP API authentication has a similar problem to database authentication.
- Our existing service Support calls for Asset Importer are painful because it is hard to manage / troubleshoot the issues…
    - The most dramatic issue recently was with FRIT (multiple-part issue that had remote database as a key part)… I have a 52 email-chain for anyone that wants to read about it (case #35424)… There have been other cases over time related to manageability issues with Asset Discovery… Ask Weylan about remote database support for asset discovery if you are interested..
    - See bug #26797 and bug #17720.
    - Note that our top category of support last month was related to Unmanaged Assets (which a big chunk of that was the importer)

Analysis of issue

- Server-side services are requirement for functionality or advanced functionality of applications.
- We are developing more of these and we probably will continue to develop more for future applications.
- We don’t want to deal with the explosion of independent application services (like asset importer) that each are developed independently… It leads to difficult to manage situations for the customer and for us.
- We need to centralize/standardize database access and SOAP API access.

Note about database access in BigFix

- There are 3 forms of database access that are used by customers:
  - NT Authentication local – Easily supported when services running as local SYSTEM
  - NT Authentication remote (requires domain account) – Requires customer to manually "log on as a service" independently for each service they wish to use.
  - SQL Server Authentication remote (requires username/password) – Requires support from service to know how to look up user info and connect to the database.
- FillDB and GatherDB need to connect to the database using one of these methods and the BigFix installer will configure these or the customer will manually configure them.
- Whatever FillDB is doing to access the database, all the applications services will need to do the same.

Use Cases

- Database maintenance tools like BES Computer Remover, BES Audit Trail Cleaner.
- Download Caching Utilities
- Power Management
- CPM
- Unmanaged Assets Importer
- DSS Aggregator
- Doug's tool that monitors FillDB traffic.
- Dawson is making a download indirection utility that downloads ISO's, extracts them, and pulls out UNIX software files for Fixlet downloads from them.

Requirement Details

Requirements

Provide organized, simple and centralized management over BES Server plugin applications. Needs to present a standardized interface for application engineers, service engineers and partners to develop stand alone applications on top of the existing platform server services.

Future Requirements

We will likely want to add the following features at some point:

1. Ability to plug-in dll's instead of (or in addition to) stand alone applications. Must build a plug-in api to develop against.

2. Advanced scheduling options similar to scheduled tasks. Would probably start with schedules that allow you to run once a day/week/month at a specific time or on every Y days of the week at a specific time.
3. A UI for controlling individual applications, force killing and changing command line arguments. Similar to windows services management interface.
4. Merging in existing BES Services like FillDB and GatherDB.
   This one scares me... I would recommend against it because FillDB and GatherDB are such core components with a long history --bkus 20:10, 3 August 2009 (UTC)
5. Add common tools into the service itself.
6. Allow more than one copy of each application to be run.

Non-requirements

List the detailed non-requirements here.

Notes / Issues

Proposed Implementation

Overview

1. Runs as a service
2. Installs into the [HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\EnterpriseServerFolder] + \Applications folder.
3. Has an interface to register/unregister applications. This should be an inbox style registration where each application will drop a simple formatted registration document into the folder that tells the BES Server Plugin how to launch the application. Applications should be able to re-register and unregister themselves as well to allow for updates.
4. Each registered application should be launched with the user context of the service.
5. Each application should be passed commandline parameters as specified when the application registers when launched.
6. Each application will be run on a schedule as specified during the registration.
7. The BES Server Plugin Service should create a log file next to it's executable as well.
8. Should be a self installing/uninstalling executable. Installation will create the BES Server Plugin Service service.
9. Should be added into the BES Server installer to become a part of the BES installation for future versions of the product.
10. The BES Server installer needs to handle upgrading the BES Server plugin Service.

Details

Plugin Service

1. Should have the Startup type set to Automatic
2. Should have Recovery set to restart the service.
3. Should have the name: BES Server Plugin Service
4. Should have the description: Runs BES Server Plugins

5. Checks the Applications\Config folder on startup and reads in all registration.xml documents as plug-in registrations. See guide below for registration format.
6. Runs each plugin application on the schedule specified by the registration documents.
7. Runs each application in the location of the executable being run (rather than the location of the BES Server Plugin Service executable).
8. Runs each application as the same user as the BES Server Plugin Service.
9. Substitutes in the command line option variables as specified by each applications registration document.

Installation & Configuration of Plugins

1. Applications should put a copy of their executables into the [HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\EnterpriseServerFolder] + \Applications folder. It is ok to make a subfolder for your plugin beneath the Applications folder. Example: C:\Program Files\BigFix Enterprise\BES Server\Applications\UnmanagedAssetImporter \UAImporter.exe
2. Applications should put a copy of their registration.xml document (see below) into the Applications\Config folder during installation and restart the Plugin Service. Put the application name into the name of the registration.xml file as the identifier for your application.
3. To uninstall, delete your registration.xml document and restart the Plugin Service.

Why require restarting of the registration service? That seems like a very likely point of failure. Seems simple enough to just poll the configuration directory periodically (every 60 seconds, say), and reconfigure dynamically. --ploer 21:01, 4 August 2009 (UTC)

> *Updating only on startup of the service is easiest code solution, seems good enough for the initial release. Also, if we have it poll to check for updates, an 'inbox' style registration would be a lot better as then you know when your registration request was processed and aren't left in the dark about whether your request was handled or not. --Tyler*
>> *Polling should not be hard to implement, and if it's every 60 seconds there shouldn't be any real concern about "being in the dark" as far as whether your changes have been picked up. We have had lots of experiences of the confusion caused for users by updating a setting and forgetting to restart the service to make it take effect. Also, restarting a service may be a hassle from fixlet content, which is the most natural way to install apps. --ploer 22:33, 4 August 2009 (UTC)*
>>> *Restarting services is easy for Fixlets. If the list is updated dynamically we have to worry about how to handle updates to existing running plugins. Do you force kill it, wait for it to complete, how do you modify the running schedule dynamically? The stop/start service handles this naturally and requires no additional coding.*
>>>> *I don't see why 'how to handle updates to eixsting running plugins' is a very big deal... you just wait for it to complete and move to the new schedule for its next run. On the flip side, if you need to add a new plugin and there is a long-running plugin in process, it's a drag to have to wait for that other plugin to complete its run before the service can succesfully shut down and your service can start running. (and it seems like this would also probably lead to unpredictable install behaviors) --ploer 22:34, 5 August 2009 (UTC)*

Registration File Specification

To add a new plugin or update an existing plugin, create a file with the name "MyPluginName.xml" in the Applications\Config directory and the contents:

```
<BESPluginRegistration>
<Name>MyPluginName</Name>
<Command>C:\Program File\BigFix Enterprise\BES Server\Applications\MyPluginName\MyPlugin
<Schedule>
 <WaitPeriodSeconds>600</WaitPeriodSeconds>
</Schedule>
<ForceKill>True/False</ForceKill>
</BESPluginRegistration>
```

The Command should specify the path and executable name for the plugin along with the command line arguments to use to run it, be sure to include and of the special %VARIABLE% options needed for the plugin.

The ForceKill flag will cause a kill command to be issued for this application when the BES Server Plugin Service is stopped. This is intended to be used for applications that are always running, like a server application that is responding to network requests. This may also be used with long running applications that may prevent the BES Server Plugin Service from shutting down in a timely manner.

To unregister a plugin, remove associated .xml file.

Be sure to restart the plugin server to get it to notice changes.

Commandline Substitution Specification

Replace any appearance of the following strings with the corresponding value if the string appears in the commandline for running the executable.

- %DATABASEDSN%
  - Default value of 'bes_bfenterprise'
  - Value of the following key if it exists.

```
[HKLM\Software\BigFix\Enterprise Server\Database]
"DSN"[REG_SZ]
```

- %DATABASEUSERNAME% --> changed to %DATABASEUSER%
  - Default value of ""
  - Value of the following key if it exists.

```
[HKLM\Software\BigFix\Enterprise Server\Database]
"User"[REG_SZ]
```

- %DATABASEPASSWORD%

    - Default value of ""
    - Value of the following key if it exists.

```
[HKLM\Software\BigFix\Enterprise Server\Database]
"Password"[REG_SZ]
```

- %SOAPUSESSL%
    - Default value of 0 (disabled)
    - 1 if this registry key is set to anything other than 0.

```
[HKLM\Software\BigFix\Enterprise Client\Settings\Client\_WebReports_HTTPServer_UseSSLFla
Value (REG_SZ)
```

- %BESHOST%
    - Default value of "127.0.0.1"
    - Host value parsed out of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\Installer]
Hostname (REG_SZ)
Example value: 'MyServername:52311'->'MyServername'
```

- %BESHTTP%
    - default value of "http://127.0.0.1:52311"
    - [HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\Installer]

```
Hostname (REG_SZ)
```

- %SOAPPORT%
    - Default value of 80
    - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\EnterpriseClient\Settings\Client\_WebReports_HTTPSer
Value (REG_SZ)
```

- %WRHTTP%
    - Default value of http://127.0.0.1
    - SOAPURL without the wsdl

- %SOAPURL%
    - Default value of http://127.0.0.1/webreports?wsdl
    - Replace 127.0.0.1 with the host/port specified by these registry keys if they exist:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\Installer]
Hostname (REG_SZ)
Example value: 'MyServername:52311'->'MyServername'
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\EnterpriseClient\Settings\Client\_WebReports_HTTPSer
Value (REG_SZ)
```

- %SOAPUSERNAME%
  - Default value of ""
  - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\BESReports]
SOAPUsername (string)
```

- %SOAPPASSWORD%
  - Default value of ""
  - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\BESReports]
SOAPPassword (String)
```

- %REPORTINGDSN%
  - Default value of 'LocalBESReportingServer'
  - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\FillAggregateDB]
"LocalDBDSN"[REG_SZ]
```

- %REPORTINGUSERNAME%
  - Default value of ""
  - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\FillAggregateDB]
"Username"[REG_SZ]
```

- %REPORTINGPASSWORD%
  - Default value of ""
  - Value of the following key if it exists.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\FillAggregateDB]
"Password"[REG_SZ]
```

- %PROXYURL%
  - Default value of ""
  - Value of the following client setting if it exists: _BESClient_Comm_ProxyServer
- %PROXYPORT%
  - Default value of 80
  - Value of the following client setting if it exists: _BESClient_Comm_ProxyPort
- %PROXYUSERNAME%
  - Default value of ""
  - Value of the following client setting if it exists: _BESClient_Comm_ProxyUser
- %PROXYPASSWORD%
  - Default value of ""
  - Value of the following client setting if it exists: _BESClient_Comm_ProxyPass
- %PROXYAUTODETECT%
  - Default value of 0
  - Value of the following client setting if it exists: _BESClient_Comm_ProxyAutoDetect

Log Specification

1. The log should be written to a file specified by the following registry key or defaults to 'BESServerPluginService.log' next to the service executable. [HKEY_LOCAL_MACHINE\SOFTWARE\BigFix\Enterprise Server\BESServerPluginService \DebugOut]
2. Log when it starts up and shuts down with the current time.
3. Log processing of registration.xml document. Give the name of the document being read and indicate whether it was successfully parsed and registered.
4. Log every application launch with the current time and identifies the name of the application as specified in the registration document.
5. Also in the application launch log, include the command line line being used with the command line variable values substituted in. Replace passwords in the log message with ***** but print usernames.
6. Log exceptions
7. Log common problems like the application executable not found, failure to find expected registry keys, failures launching the applications and detecting that the application is already running.
8. Log the return code from each application when it finishes running.
9. If a plugin with a run period of 0 is found to not be running a log message should be created as this indicates the application died and needed to be restarted.

Scheduling

The initial release will only support the following scheduling options:

```
<WaitPeriodSeconds>X</WaitPeriodSeconds >
```

The BES Server Plugin Service should launch the application, wait for it to complete and then wait X seconds before launching it again. This will help ensure only one copy of each application is launched at a time and we don't have to force kill applications. A run period of 0 indicates that the application should be always running and it will be relaunched immediately if it is not currently running with a log message indicating the plugin died.

Service Stop Requests

On a service manager stop request, the BES Server Plugin Service should

- Stop running any new application plugins
- If the ForceKill registration flag is set, issue a kill request to the application.
- Otherwise, wait for any running plugins to complete before returning a successful service stop request.

Impact Analysis

Address the impact of the proposed implementation in the following areas.

- **Scalability**
  - Need to ensure it can handle running least 20 different applications and meet their scheduling demands.
- **Testability**
- **Security**
  - Need to be careful not to expose the username/passwords.
- **Documentation**
  - This wiki will be good enough for internal documentation.
  - Will need to create a KB article for troubleshooting it.
  - May need to create an external guide for partners.
- **Portability**
- **Performance**

Notes / Issues

- The current registration format only really allows for running applications periodically. I think there are cases where we will want to run applications at specific times, like every Saturday at Midnight. Or, run it once on startup and close on shutdown. Or any of the advanced scheduling options available in the Windows Scheduled Tasks API.
- The initial release will likely be released out of band in Davis. So, we'll need to have a way to install it without the BES Server installer initially and make sure the BES Server handles whatever we do on upgrade to Evans.
- We should allow the application registrations to specify a maximum run time after which the process will be killed. Not necessary for initial release though.

General Issues / Questions

Q: What language/environment should the plugin service be implemented in?

- Pros Dev/C++
  - Makes use of existing code for launching processes as a user, running as a service, exception handling, xml, others?
  - Will get updated regularly as a part of our code base.
  - Will get more attention from more people.

- Cons Dev/C++
  - Requires development time to work on it. Has to compete with other coding projects and devs may or may not have time for it.

- Pros Perl
  - Should be fast to implement
  - Can be developed outside of the development organization

- Cons Perl
  - May not run correctly on all our supported server platforms/configurations (Win 2008?)
  - The perl service may fail in different ways than we are used to our service applications failing. We may see situations when the Perl plugin service fails while all the other BES Server services are fine.
  - We'll have to troubleshoot it differently than we would our c++ services. No dump files for example.
  - Exception handling is different.

Q: What are the advantages/disadvantages of a dll vs. process lauch plugin framework?

- Pros dll:
  - More functionality can be put into the plugin framework and moved out of the individual tools, like database operation.
  - More robust and enforcible framework
  - New API calls can be implemented through the dlls.

- Cons dll:
  - Can't use existing application not authored by bigfix.
  - Harder for plugin developers.
  - Harder and more time consuming to implement

The cons likely highly outweigh the pros here.

Q: What would we be interested in doing in terms of only launching a plugin executable if it's not already running, terminating plugin executables on process shutdown, etc.?

- Terminating should send a shutdown request but not force kill.
  - *What does this mean? GUI apps with event-processing loops can be sent a shutdown message, but I would think most plugins would be command line applications, for which this is not an option. We could build some sort of message-passing pathway... or just force-kill the processes. --ploer 21:13, 28 July 2009 (UTC)*
- Should only launch a plugin if it isn't running already.

    - *Should this be a hardwired rule or a configuration option? Maybe there are some use cases where you want multiple instances of a plugin to pile up? (maybe not.) --ploer 21:13, 28 July 2009 (UTC)*
- If there is a UI, we could have a force kill button.
    - *I wouldn't think there would be any UI... (see comments under UI question below). --ploer 21:13, 28 July 2009 (UTC)*

Q: Is there any design or implementation overlap between this plugin API and the server download plugin work for evans?

- Both the MFS plugins and the download plugins are custom apps that will probably be created by application engineering, professional services, or partners. The registration process probably wants to be similar. The download plugins register themselves through a file inbox mechanism, which we might want to reuse.
- However, the server download plugin needs to run in the context of a user who can get through proxies, and this plugin service needs to run in the context of a user who can access a remote database, so we probably can't combine the two in any real way.
    - *Do we see there eventually being a need for some applications to have both types of plugins with IPC between the two? --ploer 21:17, 28 July 2009 (UTC)*

Q: Install/Uninstall of plugin service

- Pros of being in the BES Server installer
    - Handles NT Authentication.
    - Upgraded automatically with the BES Server, keeps the two compatible at all times with each other.
    - Uninstall of the BES Server will also remove the framework.
    - BES Remove will be able to delete all the plug-ins.
    - Uninstall of the BES Server stops plugins from running.

- Cons of BES Server installer
    - Only upgraded on product releases, dunno if it will be helpful to update it out-of-band with the rest of the BES Server.
    - Will be hard to release it in Evans so post-evans would be the soonest and that is a long time away.
    - May break applications when BES Server is upgraded if the framework changes and the applications haven't been updated.

Could we release the intial version as a standalone/fixlet-installed component and then roll it into the platform installer in Evans or Foothill? --ploer 21:18, 28 July 2009 (UTC)

Q: Install/Uninstall of modules – are fixlets a good way to do this? Would we like to eventually make this a part of site subscription?

- Fixlets make things simpler now, but eventually it would be the best user experience if these plugins were installed automatically when they are part of sites.
- We could put a special file in sites that GatherDB would look for after importing a site. The file would specify what file in the site is the plugin and would provide any necessary configuration

information.

- *Do we need to give the user any special warning that they are launching an executable on their server? (this seems like it requires a meaningful extra level of trust in the site author) --ploer 21:20, 28 July 2009 (UTC)*

Q: Need for backwards compatability – if we do a very simple initial implementation now, is there any problem in continuing to support a simple interface as we add new capabilities to the 2.0 interface?

- No, but it's hard to say without having much of an idea of what capabilities would go into the 2.0 interface. Past experience states that backward compatibility is always a pain but just something we deal with.
  - *My initial sense is that backwards compatability here may be pretty simple. --ploer 21:20, 28 July 2009 (UTC)*

Q: Logging / debuggability?

- Needs to create a log message on startup/shutdown with the version and timestamp.
- Needs to log each time in launches a plug-in with a time stamp and the full command that was run.
- Needs to log exceptions.
- Creates its own log file in the '/BES Server/Applications' directory.

Q: Communication between different plugin processes?

- Don't see how this fits in with the current use cases.
- Communication can always be done the database or the filesystem in the /BES Server/Applications directory.
- *So, not something to worry about for 1.0 in any case. --ploer 21:21, 28 July 2009 (UTC)*

Q: Database version compatability, database version upgrade path?

- If it is part of the BES Server installer/upgrade it should be handled automatically.
- Assume applications handle checking the database version on their own.

Q: Should it have a UI?

- Would allow you to setup/control/uninstall of each application quickly and easily.
- Would provide a natural interface to force-kill applications.
- Provides a nice troubleshooting interface for individual applications. Let's you start/stop each individually.

I wouldn't think there would be any UI... we're talking about a service running under a special user account, who would it display the UI to? --ploer 21:13, 28 July 2009 (UTC)

> *Well, I guess you could have BESAdmin provide UI to edit the configuration files... --ploer 21:23, 28 July 2009 (UTC)*
>
> > *The UI would be analogous to the Windows Task Manager interface or the Windows Service interface. We are basically re-implementing them, thus the UI would have all the same features*

as those management interfaces. BES Admin would be a logical place to put it too.

Q: Are there any special implications/requirements for this service in a DSA configuration?

- No, each DSA server is supposed to be stand alone in case the other server is down so you need to have the Plugin Service running on all DSA servers. It is up to the plug-in application engineer to ensure their plug-in handles running on each DSA server.

Other Resources

Include links to external resources (MRDs, bugzilla feature requests, forum comments, third party capability descriptions, etc.) here.

- BES_Server_Application_and_Add-on_Requirements

Retrieved from "https://bfwiki.prod.hclpnp.com/index.php?title=BES_Server_Plugin_Service_Proposal&oldid=38226"

- This page was last modified on 19 August 2010, at 23:37.