Colledia Control Configuration

Client / Server

Description

The Configuration server is a TCP server application that runs on the file server for a V4.5 Colledia Control cluster.

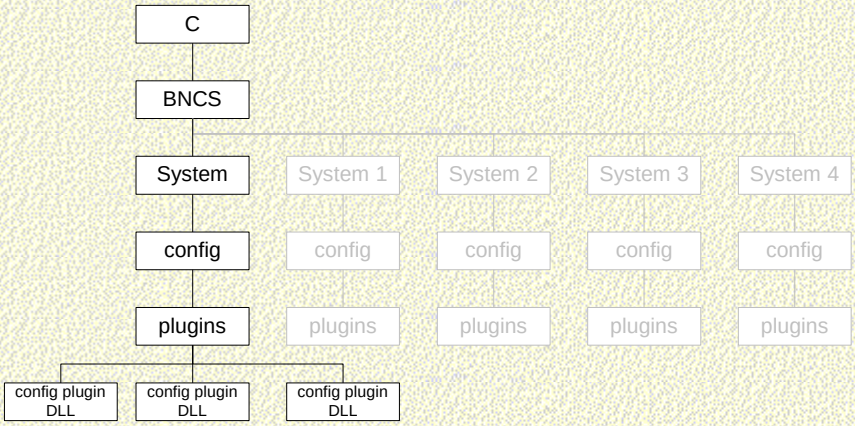
The interface is XML. The configuration server application provides a "configuration gateway" onto the file server.

The configuration server provides services to:

- login – this is not so much for security (though it does this too) but also for revision tracking and file locking
- system listings i.e. what systems are available to me on this server
- load files from the server
  this also locks the files as being "under edit".
- save files to the server
  this also notifies the update mechanism that this file needs to be distributed to all machines on the cluster.
  it also ensures that a backup is taken when the new file is submitted allowing for rollbacks

Directory Structure

The minimum directory structure for the config server to operate is (assuming that the data is stored off the c:\ path)



Points to note:

There can be multiple systems on a single server.

Each server has it's own tree, and its own copy of the configuration tool plugin DLLs

Usually there will be sub folders for each System containing the binaries and support files for that system

XML Interface

The client and server communicate using a simple XML interface.

There is a single top level tag:

```
<xml>
     … session in here
</xml>
```

The session must start with an <xml> tag and the session closed with </xml>

The next level of tags specifies the operation to be undertaken.

The general format of these 2nd level tags is:

```
<tag type="reply" uid="xxxx" system="bncsi" >
     <moretags/>
</tag>
```

type can have the following values:

Type	Description
reply	A tag usually containing data
ack	An acknowledge of sent command where there is no return data
error	A command that caused an error

uid is a value that the client generates and is simply returned by the server in responses to commands. The server doesn't use this value for any purpose. It is present for the client to keep track of the context of the replies it is receiving. Usually this value will contain the unique handle of the plugin that initiated the command.

system specifies which subfolder on the server

These 2nd level tags (which describe the fundamental features available) are:

Tag	**Description**
login	authenticate this user (no other operations are possible until this step is completed)
download	download single file
synclist	provide or request a list of files for the client and server to synchronise directories
lock	lock a file or path
systems	list the systems available to this config server
dir	list files on a config server
locklist	List of files locked to this user

See table below for a complete explanation of the parameters and return values for each 2nd level tag.

Client transactions

These are how the raw configuration server commands noted above are presented at the user.

The following table contains the commands/responses as seen from a client plugin. These responses are generated based on data returned from the server and data stored locally in the configuration client shell.

In the following table please note the following convention:

<filename> refers to single named file
<dir> refers to a directory only
<path> refers to either a file or directory

Client command	Parameters	Server Response	Notes
download	<filename>	OK FAIL READONLY	Download single file. If this file already has been downloaded by another plugin on this machine then an immediate response is returned. Paths which are locked to this user return “OK” otherwise they return “READONLY”
upload	<filename>	OK or FAIL	Upload single file. The file need not already exist on the server but it must have been locked by this user
syncDirDownload	<dir>	“download” response as above for each file followed by DONE when the download is complete	Download entire directory – this will synchronise (add and delete files) the local directory with the server **except** where files in that local directory have individual locks
syncDirUpload	<dir>	OK or FAIL for each file followed by	Upload entire directory – this will synchronise (add

		DONE when the upload is complete	and delete files) the server with the local directory. The directory (or one of it's ancestors) must have been locked by this user
lock	<path>	OK or FAIL	Lock path (files or directories) that they are available to this workstation. The path need not exist to be locked.
unlock	<path>	OK or FAIL	Unlock path that have been locked by this workstation
dirList	<path>	OK or FAIL	Return directory listing for the specified path
locklist	(none)	OK or FAIL	Return list of locks for this user

File Locking

For a configuration editor to edit a system file that file must be locked-out to the user.

In reality this just means the configuration server making a note that a particular user has a lock on a path – whether that path is a file or directory.

Directory locking makes things a bit complicated as it refers to multiple files.

Locking from two different PCs

So if a *UserA* has locked

```
config/alarms
```

and *UserB* tries to lock

```
config/alarms/stuff.ini
```

then *UserB* lock request will fail as *UserA* already has the lock on that path.

On another occasion, if a *UserA* has locked

```
config/alarms/stuff.ini
```

and *UserB* tries to lock

```
config/alarms
```

then *UserB* lock request will fail as *UserA* already has the lock on something in that path.

The general guide is not to lock more files or directories than is necessary. If your system splits nicely into sub-systems then give each sub-system its own configuration directory so that each may be locked independently – it's what directories are for after all (grouping together associated files).

Locking from two different plugins on the same PC

This only works if the lock paths are the same. *UserA* loads two plugins on a single PC that use the same files

So if a *PluginA* has locked

```
config/alarms
```

and *PluginB* tries also tries to lock

```
config/alarms
```

then *PluginB* lock request will succeed even though *PluginA* already has the lock on that path. When either releases the local lock count is reduced but the path is still locked out to *UserA*

On another occasion, if a *PluginA* has locked

```
config/alarms
```

and *PluginB* tries to lock

```
config/alarms/mystuff.ini
```

then *PluginB* lock request will fail as *PluginA* already has the lock on a different part of the path.

In theory different plugins locking the same file but via a different path is a valid thing to do as the file is locked to the same user. However, keeping track of what the locks are, particularly when you cannot determine the order in which they are locked an unlocked is difficult (too hard for this version).

Ideally there should only be one editor for each file and only one that writes to it. Other files may request files read-only but they do not require it to be locked. Ideally context sensitive menus should guide the user from where read-only data appears to the one-and-only editor for that data.

Lists

Config Server must maintain lists of:

Param	Param	Persistence	Filename
locked path	user	persistent	%CC_ROOT%\%CC_SYSTEM%\server_lock_list.txt

Config Client must maintain lists of:

Param	Param	Persistence	Filename
Path already locked by this user	Plugin	persistent – checked against server at startup	%TEMP%\BBCTechnology\ConfigurationClient\lockLists\%CC_SYSTEM%\lock_list.txt
downloaded files	Plugin	for each session – not persistent Used to determine what files are updated off the server when they change and which plugins get notified of the changes	%TEMP%\BBCTechnology\ConfigurationClient\downloadLists\%CC_SYSTEM%\download_list.txt

File Downloading / Synchronising

It is inefficient to transfer files that may already exist on the destination machine so a smart file-synchronising mechanism is used.

Files aren't transferred directly – a list of the files that would be transferred is sent instead. This is then compared with the local copy of the files (if there are any) and lists of additions, changes and deletions generated.

Client Startup

Transfers are noted by → and ← symbols

Client Action	Server Action
Server select dialog Broadcast for servers on this subnet →	
	← return server name to client
User selects system to connect to – connect session	
	← send <xml> packet to start session
send request for list of systems on this server →	
	← return list of systems on this server
User selects current system and prompts user for user credentials	
Send login string to server →	
	← send login ack/error back to client
Show main client window	
Send request for plugin directory synclist →	
	← reply with list of files in the config plugins directory
Work out what files are required (different to what we've already got)	
Request plugin files →	
	← return plugin files
Send request for lock list →	
	← return list of locks for this user
reconcile list of locked files received with that we've already got	
load plugin DLLs, requesting keys and icons etc. from each DLL to fill configuration treeview. Plugins are loaded only for the purposes of getting this information	

Done!	

Client Loads Plugin (on selecting in treeview or navigating from another plugin)

Shell	**Plugin**
Send deactivate signal to existing plugin	
Send activate signal for new plugin	
Send request for menus	
	return menus
Send request for toolbars	
	return toolbars
send navigate event to plugin	
	set context using navigate context (note: at startup this may require storing the navigate context so that it can be used when all the files that are required have been downloaded).

Transferring file from server to client (download)

Transfers are noted by → and ← symbols

Client Action	Server Action
Request file →	
	← return file to client
Remember list of downloaded files	
Check if this client has lock on this file	
send notification back to client and plugin	

Transferring file from client to server (upload)

Transfers are noted by → and ← symbols

Client Action	Server Action
Compress file and send to server →	
	Check that this client has lock on this file or path
	← return ack/error to client
return ack/error to plugin	

On synchronising server to client (download)

Transfers are noted by → and ← symbols

Client Action	Server Action
Request Sync List →	
	← Return synch list
Generate new/updated files list comparing incoming list with one generated off our existing local copy (might not exist at all….).	
Delete local files – notify other plugins	
Request new/updated files →	
	← return zipped files
save files and notify plugins as files arrive	
Remember list of downloaded files	

On synchronising client to server (upload)

Client Action	Server Action
Send Sync List →	
	Check client has lock on this path
	Generate update list
	Delete local files and notify other clients
	← request new/updated files
return zipped files→	
	save files and notify other clients
	← send sync list ack

Notification at client of file change

Has file been referenced by any open plugin?	Yes – download file No – do nothing

File Incoming

Is file open in any plugins? (should be we wouldn't have had this file arrive otherwise)	Yes – notify the interested plugins (may be one or several) that this new file is available No – do nothing

Appendix

Here is a listing of some of the test strings used in the development of the configuration server. The master of this list can be found with the server code.

It is not comprehensive since it does not show every possible error response.

This shows examples of client commands and server responses. This data can be used on a normal Telnet session to the server for testing.

```
============================
SEND: SYSTEM LIST AND LOGIN
============================
<xml>
        <systems type="request" />
        <login id="dave" password="dave" system="test" />

TO REDCEIVE:
============
<xml>
<systems type="reply">
         <system name="test" />
         <system name="testRig" />
         <system name="v4.5" />
</systems>
<login type="reply" id="dave" level="admin" />


======================
SEND: DIRECTORY LISTING
======================
<xml>
        <login id="dave" password="dave" system="test" />
        <dir type="request" system="test" path="/" recursive="false" />

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<dir type="reply" uid="" system="test" filter="">
         <item type="dir" time="2005-01-13T15:12:30" path="/" >backup</item>
         <item type="dir" time="2005-01-13T15:13:38" path="/" >config</item>
         <item type="dir" time="2005-01-10T12:01:19" path="/" >windows</item>
                  (this list system dependent of course.....)
</dir>


==================================
SEND: DIRECTORY LISTING WITH FILTER
==================================
<xml>
        <login id="dave" password="dave" system="test" />
        <dir type="request" system="test" path="/" recursive="false" filter="*.exe" />


TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
        <dir type="reply" uid="" system="test" filter="*.exe">
                <item type="file" time="2003-10-22T04:00:00" path="/" >unins000.exe</item>
                        (this list system dependent of course.....)
        </dir>


===============
SEND: LOCK PATH
===============
<xml>
        <login id="dave" password="dave" system="test" />
```

```
        <lock uid="101" type="request" system="test" >
                <file name="config/hello.txt" />
        </lock>

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<lock uid="101" system="test" type="reply" >
        <file name="config/hello.txt" type="reply" />
</lock>


=================
SEND: UNLOCK PATH
=================
<xml>
        <login id="dave" password="dave" system="test" />
        <unlock uid="101" type="request" system="test" >
                <file name="config" />
        </unlock>

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<unlock uid="101" system="test" type="reply" >
        <file name="config" type="reply"/>
</unlock>


======================
SEND: DOWNLOAD FILE(S)
======================
<xml>
        <login id="dave" password="dave" system="test" />
        <download uid="102" type="request" system="test" >
                <file name="config/hello.txt" />
        </download>

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<download uid="102" type="reply" system="test" >
        <file type="reply" name="config/hello.txt" size="989" compressedsize="294" date="2005-01-
13T15:02:57" >
789C8DD34D4FC3201807F0B3267E07E4EED6D6B73669DD41E7CD68B27AD89115D292959700D5F9EDC54D28A923EEC6F32FBF8
7863C948B1DEBC107519A0A5EC174964040782330E56D05DFEBE7AB1C2E1E2ECECBCBA7D7C77AFDB6049F426DB541C6EED760
B55ED5CB1700C370860D863F240C6D7D160680627B5A9A42C01123156C04631A13BDB5C9A0890A12388FE0EC0FCE4EC7D70E7
7C3C69FC908A6C806517513A8EC6475EB548BFA9EA82FC31C1D8398BD9B582D068E27FC90C53ADCBB0E5211CDE88E28C7C720
66F3D0B60AC98E363AE43E8B75285C87FDA74E48A77D1D9159E22541A61BFFDA9531E7674A0F1B434D6F27D9D12089693F548
2DB7DC4C9DF2AA6FC342129FDFDECD7C7459EF89B6D3A453D3914C74D51FC43CAF9E4C57D03E1594281</file>
</download>



==================
SEND: UPLOAD FILES
==================
<xml>
        <login id="dave" password="dave" system="test" />
        <upload uid="103" type="request" system="test" >
                <file name="config/hello.txt" size="989"
>789C8DD34D4FC3201807F0B3267E07E4EED6D6B73669DD41E7CD68B27AD89115D292959700D5F9EDC54D28A923EEC6F32FBF
87863C948B1DEBC107519A0A5EC174964040782330E56D05DFEBE7AB1C2E1E2ECECBCBA7D7C77AFDB6049F426DB541C6EED76
0B55ED5CB1700C370860D863F240C6D7D160680627B5A9A42C01123156C04631A13BDB5C9A0890A12388FE0EC0FCE4EC7D70E
77C3C69FC908A6C806517513A8EC6475EB548BFA9EA82FC31C1D8398BD9B582D068E27FC90C53ADCBB0E5211CDE88E28C7C72
066F3D0B60AC98E363AE43E8B75285C87FDA74E48A77D1D9159E22541A61BFFDA9531E7674A0F1B434D6F27D9D12089693F54
82DB7DC4C9DF2AA6FC342129FDFDECD7C7459EF89B6D3A453D3914C74D51FC43CAF9E4C57D03E1594281</file>
        </upload>


TO REDCEIVE:
============
```

```
<xml>
<login type="reply" id="dave" level="admin" />
<upload uid="103" type="reply" system="test">
        <file type="reply" name="config/hello.txt" />
</upload>



=======================================
SEND: GET SYNCILST (SYNC SERVER TO LOCAL
=======================================
<xml>
        <login id="dave" password="dave" system="test" />
        <synclist uid="xyz" type="request" system="test" path="windows/lib/alarms" />

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<synclist uid="xyz" type="reply" system="test" path="windows/lib/alarms" >
        <data name="ALM_BNCS_ACQ.DLL" date="2004-12-21T10:11:44" size="102400"/>
        <data name="ALM_BNCS_OP.DLL" date="2004-12-21T12:09:37" size="49152"/>
        <data name="ALM_DB_OUTPUT.DLL" date="2004-12-21T12:16:10" size="73825"/>
</synclist>

============================================
SEND: PROVIDE SYNCLIST (SYNC LOCAL TO SERVER)
============================================
<xml>
       <login id="dave" password="dave" system="test" />
<synclist uid="stuff" type="reply" system="test" path="config2" >
        <data name="hello.txt" date="2003-12-02T19:14:05" size="989" />
</synclist>

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
<download uid="stuff" type="request" system="test" >
        <file name="config2/HELLO.TXT" />
</download>

THEN SEND IN RESPONSE TO DOWNLOAD REQUEST:
============
<download uid="102" type="reply" system="test" >
        <file type="reply" name="config2/hello.txt" size="989" compressedsize="294" date="2005-01-
13T15:02:57"
>789C8DD34D4FC3201807F0B3267E07E4EED6D6B73669DD41E7CD68B27AD89115D292959700D5F9EDC54D28A923EEC6F32FBF
87863C948B1DEBC107519A0A5EC174964040782330E56D05DFEBE7AB1C2E1E2ECECBCBA7D7C77AFDB6049F426DB541C6EED76
0B55ED5CB1700C370860D863F240C6D7D160680627B5A9A42C01123156C04631A13BDB5C9A0890A12388FE0EC0FCE4EC7D70E
77C3C69FC908A6C806517513A8EC6475EB548BFA9EA82FC31C1D8398BD9B582D068E27FC90C53ADCBB0E5211CDE88E28C7C72
066F3D0B60AC98E363AE43E8B75285C87FDA74E48A77D1D9159E22541A61BFFDA9531E7674A0F1B434D6F27D9D12089693F54
82DB7DC4C9DF2AA6FC342129FDFDECD7C7459EF89B6D3A453D3914C74D51FC43CAF9E4C57D03E1594281</file>
</download>


====================================================
SEND: GET LIST OF LOCKS ON THIS SERVER FOR THIS USER
====================================================
<xml>
       <login id="dave" password="dave" system="test" />
       <locklist uid="xyz" system="test" />

TO REDCEIVE:
============
<xml>
<login type="reply" id="dave" level="admin" />
        <locklist type="reply" uid="xyz" system="test" >
                <lock name="config2" />
        </locklist>
```