CyFlex® Connection Server Setup

Version 6

February 20, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	5/18/2016	Initial publication
2	8/23/2018	Format with SGS brand
3	4/7/2020	Retrofit to new template Removed mention of ASSET and associated content
4	12/7/2021	Revised *Section 1 Overview* on page 1 and *Section 2.3 Verifying connsrvr_specs File Changes* on page 6 to add hypertext linked cross-references to applicable usage help topics on cyflex.com.
5	6/13/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
6	2/20/2024	Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1 Overview* on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

1 OVERVIEW 1

2 SETTING UP THE CONFIGURATION SERVER 4

2.1 MODIFYING THE SPECIFICATIONS FILE 4

2.2 MODIFYING THE GO SCRIPT 6

2.3 VERIFYING CONNSRVR_SPECS FILE CHANGES 6

2.4 TESTING THE CONNECTION SERVER 9

List of Figures

FIGURE 1: CLIENT - CONNSRVR DATA FLOW 2
FIGURE 2 CSDUMP OUTPUT EXAMPLE 8

LIST OF TABLES

TABLE 1: CONNECTION SERVER ENVIRONMENTS 1
TABLE 2: NODE IDENTIFICATION COMMANDS 5

1 Overview

The connection server enables computers running software on different operating systems (Linux/Windows) to share information. This includes:

- Moving data between any CyFlex systems on the network
- Sharing measurement values between tasks running on different test cells
- Remotely controlling AVL Indicom software on a Windows computer

  Note:

  This requires a CyFlex machine and the controlled Windows machine running the Indicom service application.

- Allowing Web application services to retrieve `eLog` data (electronic log book entries) and CyFlex history data

  Note:

  Web services use the `web_sm_server` (`LV_inter`) application for this.

- Working with additional software as needed since other service applications can be developed

Note:

The terms *test cell*, *node* and *computer* are used interchangeably in this document to mean a test cell computer.

The `connsrvr` connection server application can operate with current and legacy systems, sending and receiving information over a network. Test cells running Linux can communicate with cells using Windows and vice versa.

This communication works using the User Datagram Protocol (UDP). The connection server arranges messages into UDP packets and sends them to other nodes also running the connection server program.

Table 1: Connection Server Environments

Connection Server Application	Test Cell Automation Software	Operating System (OS)
`connsrvr`	CyFlex	Fedora 8 or Scientific Linux 6.0 (or higher)
Windows Connection Server (WCS)	Indicom service (TRP Labs), AVL Indicom software	MS Windows XP or Windows 7

Refer to cyflex.com usage help for `connsrvr` command syntax.

The connection server application running on CyFlex/Linux systems relies on and works in cooperation with client and server applications. Additionally, the connection server specification file must be configured to identify the other node(s).

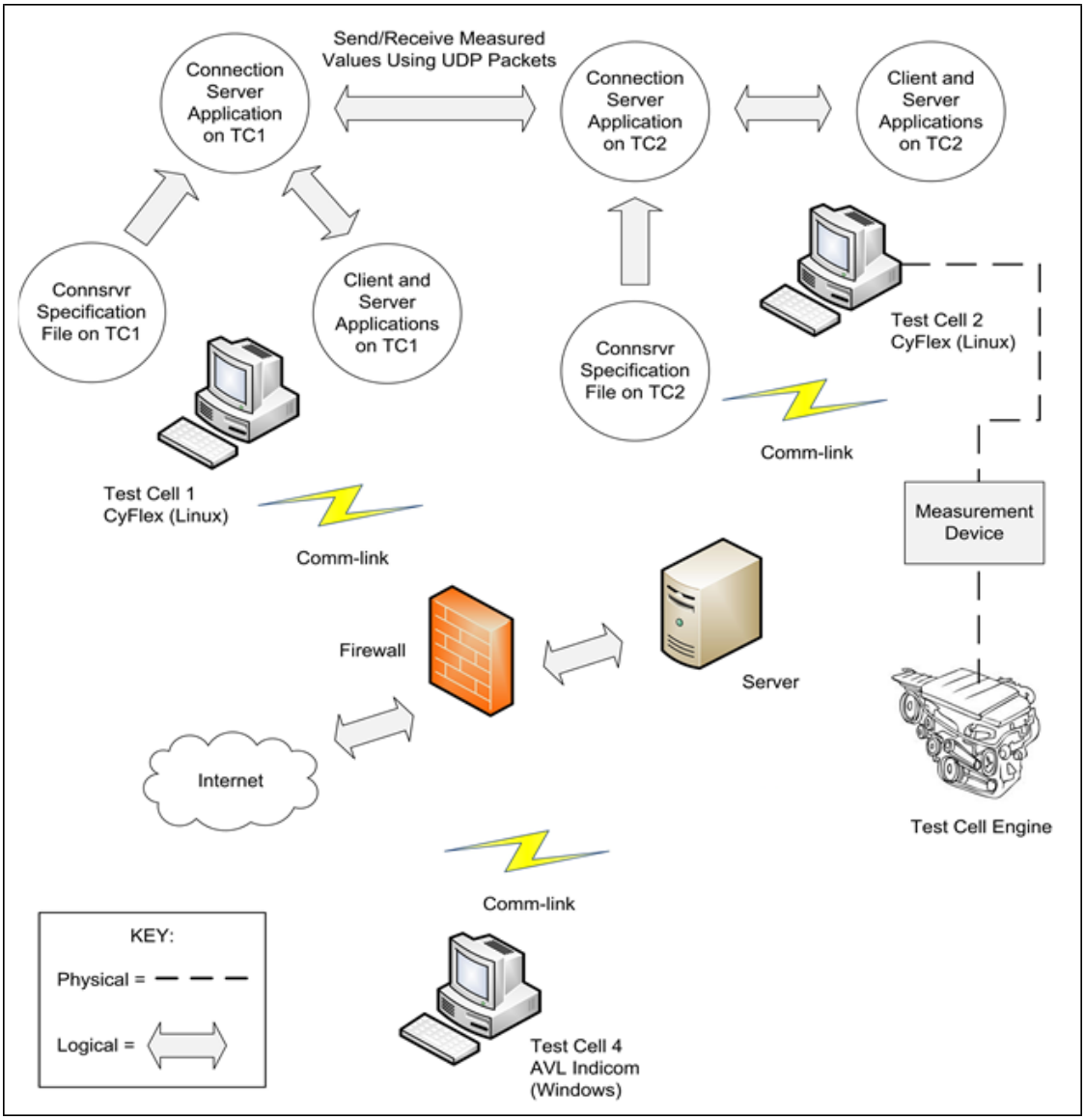
CyFlex/Linux systems require the following to communicate:

- Nodes must be connected over a network.
- Connection server application must be running on each computer.
- Service applications controlling the tasks must work with `connsrvr`.

A test cell node must be configured with the service application(s) it needs to request or send data. For example, test cell 1 runs a task that requires values from a particular measurement run on test cell 2.

Using a service application (client) and `connsrvr`, test cell 1 requests the values from test cell 2. TC2 is running a service application (server) and `connsrvr`, allowing it to send the values requested. Each service application provides a different set of services that the client or server needs.

Figure 1: Client - connsrvr Data Flow



Server applications that operate with `connsrvr`:

- `small_sm`; refer to cyflex.com usage help for `small_sm` for supplemental information.
- `web_sm_server`; refer to cyflex.com usage help for `web_sm_server` for supplemental information.
- `push_server`; refer to cyflex.com usage help for `push_server` for supplemental information.
- `startIndicom` (Windows machines for connection to AVL software); refer to cyflex.com usage help for `startIndicom` for supplemental information.

Client (requesting service) applications that work with `connsrvr`:

- `node_linkN`; refer to cyflex.com usage help for `node_linkN` for supplemental information.
- `snode_link`; refer to cyflex.com usage help for `snode_link` for supplemental information.
- `push_link`; refer to cyflex.com usage help for `push_link` for supplemental information.
- `getAstStat13`; refer to cyflex.com usage help for `getAstStat13` for supplemental information.

Notes:

The `connsrvr` program and all of the service applications listed above, except for `indicom`, are copied to the node with a CyFlex installation.

Transmitting files from one node to another can also be done manually from the command line using FTP/SFTP tools, or automatically using tools such as `tranMove`.

A sub-network is an identifiable separate part of an organization's network. Typically, a subnet represents all of the nodes at one geographic location, in one building, or on the same Local Area Network (LAN).

2 Setting Up the Configuration Server

Execute the following tasks to configure test cell nodes for use with the connection server:

1. Configure the `connsrvr_specs.nnn` file on each computer. Refer to *Section 2.1 Modifying the Specifications File* below.
2. Add `connsrvr` to the `Go` script file on each test cell. Refer to *Section 2.2 Modifying the Go Script* on page 6.
3. Ensure the test cell system is in a safe state and the engine is not running and then run the `Go` script on each computer.
4. Verify the changes made to the `connsrvr_specs` file on each computer. Refer to *Section 2.3 Verifying connsrvr_specs File Changes* on page 6.

2.1 Modifying the Specifications File

The following steps use test cell 1 (TC1) and test cell 2 (TC2) as example test cells.

Execute the following steps to modify `connsrvr_specs.nnn` on TC1.

1. Enter the following at a terminal window:

   `cd /specs`

   `edit connsrvr_specs.nnn` where *nnn* is the test cell number.

   Example `connsrvr_specs` file:

```
# Connection Server specification file
VERSION
1
# List remote nodes that provide a service to this node through the
# connection server
# Examples of the server are sm_server, small_sm, and indicom.
# Examples of the programs requesting this service are node_link,
# snode_link, and getAstStat.
# The Remote Server Port is always '-' unless there is a conflict with
# the default port 1503.
# HOSTNAMES or IP ADDRESS
SERVERS
# Hostname  Remote Server Port
CTC-TC1     -
$
# List up to 5 IP address masks that the connection server will accept
# as ad-hoc requests from
# remote connection servers. These are remote connection servers that
# need a service, such as # sm_server or indicom.
# DYNAMIC_SERVER RANGE
DYNAMIC_SERVERS
143.222.0.0
$
SERVER_TIMEOUT
300
$
```

2. Locate the hostname in the `connsrvr_specs` file. This identifies the node for the connection server application. The file may list the IP address instead of the hostname but not both, which causes a failed connection.
3. Ensure the network will recognize the hostname listed in the `connsrvr_specs` file:
   a. Open a terminal window.
   b. Enter: `nslookup`. *Table 2* lists node identification commands.

Table 2: Node Identification Commands

Node Identification	Linux Command	Comment
Hostname	`nslookup`	Node server lookup ; lists the hostname of the node
	`nslookup CTC-TC1`	Displays the IP address and FQDN (if the hostname of `CTC-TC1` is valid)
	`nslookup 143.222.168.101` The IP address is an example only.	This retrieves the hostname for the IP address, and is called "`reverse nslookup`".
Fully Qualified Domain Name	`Hostname fqdn`	Fully Qualified Domain Name (FQDN) for the hostname
IP address	`hostname ip-address`	-------

Important:

If the node server lookup does not show the hostname listed in the `connsrvr_specs` file, use the node's FQDN or IP address. A fully qualified domain name specifies the node's exact location in the tree hierarchy of the Domain Name System (DNS).

Example:

The hostname of a test cell 1 is `CTC-TC1`.

The FQDN is the hostname, plus the company's domain. Typically, the domain ends in ".com".

If an IP address is used instead of the hostname, the IP address should be one that is permanently assigned to the node. If it is changed, and the `connsrvr_specs` file is not updated for the changed address, the connection will fail.

Notes:

Ad-hoc requests are requests from nodes to retrieve data, but the nodes are not specified under HOSTNAMES or SERVERS in the `connsrvr_specs` file.

An IP address has two components: the network address and the host address. A subnet mask (the IP mask mentioned above) separates the IP address into the network and host addresses (<network><host>). "Sub-netting" further divides the host part of an IP address into a subnet and host address (<network><subnet><host>).

4. Repeat Step1 through Step 3 to modify `connsrvr_specs.nnn` on TC2.

2.2 Modifying the Go Script

1. Enter the following at a terminal window:
   ```
   cd /cell
   edit go.scp
   ```
2. Verify the first line of the file includes: `BUILD BASIC I/O SPECIFICATIONS`.
3. Add the line `connsrvr &` as in the following snippet example.
   ```
   echo "----------- BUILDING BASIC I/O SPECIFICATIONS -----------"
   connsrvr &
   sleep 10
   small_sm &

   web_sm_server &
   ```

Note:

`small_sm` and `web_sm_server` are service applications that work with `connsrvr` to answer requests. Depending on the test cell's configuration, the client applications required for use with `connsrvr`, `node_link`, `snode_link`, and `getAstStat`, would be included in the Go script.

2.3 Verifying connsrvr_specs File Changes

Execute the following steps to verify the file changes.

1. Enter the following at a terminal window:
   ```
   csdump
   ```
   This displays information about the connection server on test cell 1 as follows:

Service or Server	Description
Local Attached Services	List of registered services on the local node
Remote Attached Services	List of registered services from remote nodes
Connected Remote Servers	Status of remote servers; see next table
Unconnected Remote Servers	List of servers not completely connected to the node

Status of connected remote servers:

Remote Servers Status	Description
Hostname/IPADDR	The hostname or IP address of the remote server
Alive Time	Last time that an "Alive Message" was received from the machine
Listen Taskid	Thread listening to User Datagram Protocol (UDP) messages from Remote Server

Remote Servers Status	Description
SendTo Taskid	Process (application) that received the last message – `small_sm` is an example
Msg Count	Number of messages received
Packet Count	Number of UDP packets received
Error Count	Total number of errors receiving messages
Last Time	Time of the last message received
Send Taskid	Thread sending the UDP message to the remote server
Requestor Taskid	Process (application) that sent the last message – `snode_link` is an example
Msg Count	Number of messages sent
Packet Count	Number of UDP packets sent
Error Count	Total number of errors sending messages
Last Time	Time the last message sent

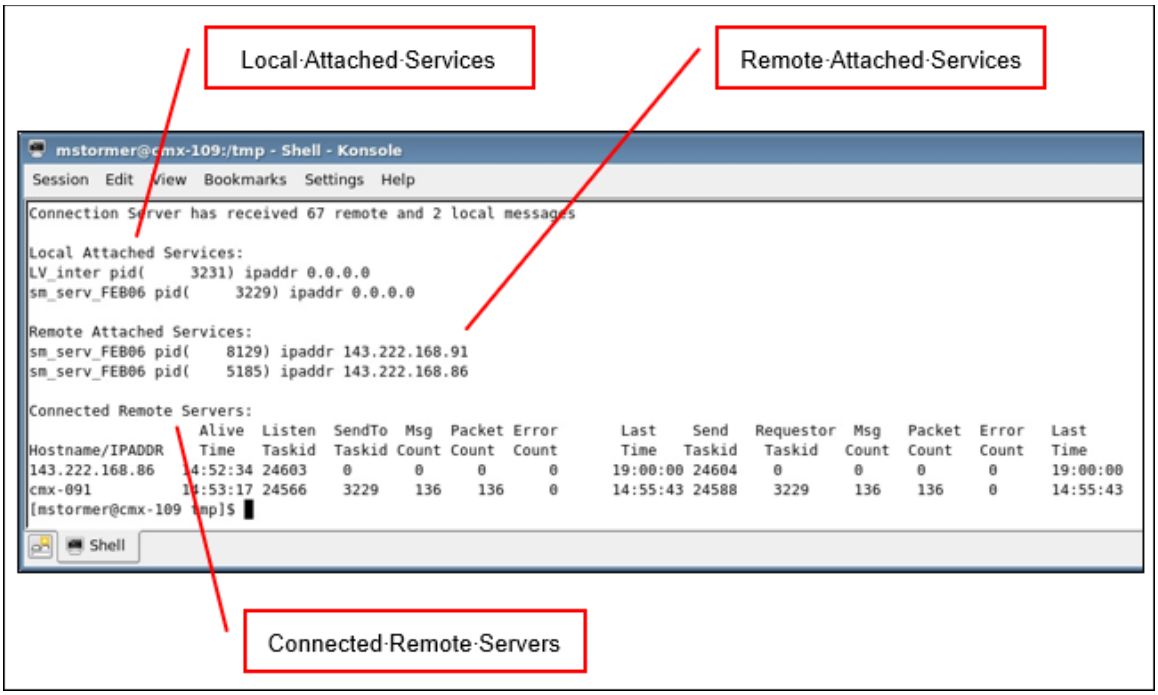
Status of remote servers not connected:

Remote Servers Status	Description
Hostname/IPADDR	The hostname or IP address of the remote server
Receive Taskid	Same as Listen Taskid
Packet Count	Number of UDP packets received
Send Taskid	Thread sending the UDP message to the remote server
Packet Count	Number of UDP packets sent

Refer to cyflex.com usage help for `csdump` for supplemental information.

2. View the `csdump` details as in the annotated output example in *Figure 2*.

Figure 2 csdump Output Example



3. Verify that the following information for TC 1 is shown under `Local Attached Services`:
   - Processes running on the local node: `sm_server` and `LV_inter`
   - Process IDs of the service applications
4. Verify whether the `connsrvr_specs` file on test cell 1 was correctly updated: The following must appear under `Remote Attached Services`:
   - Process (server application) running on the remote node (`sm_server`)
   - Process ID of the server application – `pid`(`nnn`)
   - Hostname or IP address (`ipaddr`)

   Note:

   Service names instead of the application names are displayed as shown in *Figure 2:*
   - `LV_inter` (`web_sm_server`)
   - `sm_serv_FEB06` (`small_sm`)
   - `sm_serv_MAR09` (`push._server`)
5. Confirm the connected remote server with hostname `cmx-091` is test cell 2. Enter:

   `nslookup cmx-091`

   This retrieves the IP address if one is available.

6. Verify the changes made to the `connsrvr_specs` file on test cell 2.
    a. Repeat Step 1 through Step 5.
    b. If the `connsrvr_specs` file on test cell 2 was updated correctly, the `Remote Attached Services` and `Connected Remote Servers` lists will indicate connection to test cell 1

2.4 Testing the Connection Server

Execute the following steps to test communication between the two test cell computers using `connsrvr`:

1. Open a terminal window on each test cell (node).
2. Enter: `snl_test time addr`
3. The resulting display shows:
    - Time of the test
    - Hostname or IP address of the connected remote node

    These indicate a successful connection.

Example of displayed information:

```
TC1
$ snl_test  time CTC-TC2 TC2
$ snl_test time CTC-TC1
```