Version 4.2

# Distrix User GUIDE

Published: 20/08/2014

Please contact us at: Toll Free: +1.855.657.7275

**General**: info@distrix.com

**Products**: sales@distrix.com

**Support**: support@distrix.com

**Head Office: T: 604.736.6675 | F: 604.648.9[illegible]**

1880 West First Ave, Suite 200, Vancouver, BC, V6J [illegible] Canada

## License Attributions

# Contents

About Distrix Gateway Routers ........ 7

Concept Overview ........ 9

Getting Started ........ 11

- System Requirements ........ 12
- How to Install Distrix ........ 13
- Uninstall Distrix ........ 19
- How To Login ........ 21
- Licenses ........ 22
- User Profile Setup ........ 23
- Change Password ........ 25
- Automate TAP & Bridge Utilities ........ 27
- Application Tunnel Connector ........ 28
- FIPS-140 Security Module ........ 33

Distrix User Interface ........ 36

- Overview ........ 38

Configure ........ 41

- Distrix Nodes ........ 43
  - **Apply Licenses** ........ 43
  - **Security Certificate Introduction** ........ 47
  - **Setup Users** ........ 68

Provision ........ 71

- Introduction to Tunnels ........ 72
- Create Tunnels ........ 74
- Tunnel List ........ 86
- Application Tunnel Connector ........ 87

Monitor ........ 92

- Monitor Network Topology ........ 94
- Monitor a Node List ........ 97
- Detail Node Status ........ 98
- Tunnel Connections ........ 100
- Tunnel Instance Status ........ 101
- Node Connection List ........ 103
- Link Connection List ........ 104
- Link Status ........ 106

Trouble-shooting 110

Logging 112
License Expired 113
Linux Reverse Path Filtering 114
Multiple Links Not Forming 115
Nodes Not Connecting 116
UDP or TCP Tunnel Not Working 117

Appendix A 118

File Libraries 120
File Libraries For Windows 125

Appendix B 133

Configuration Files Location 135

HTTP Server Configuration 136

Communication Configuration 137

Security Configuration 139

Link Modules Configuration 141

Link Modules List 141

Tunnels Configuration 142

App Tunnel Connector 144

Hardening Ubuntu Linux 153
Hardening Windows 160
Glossary 163

System Installation Requirements

How To Install

How To Uninstall

How To Login

How To Apply Licenses

How To Set-up User Profiles

How To Change Password

# Getting Started

# System Requirements

These are the system requirements necessary for Distrix 4.2 deployment and operation. The basic requirements are listed along with our recommendations to achieve optimal performance. The platform compatibilities are also listed.

**NOTE**: These requirements are separate from the requirements of the underlying operating system. It is possible to run Distrix 4.2 on hardware not meeting these requirements but it is not recommended, particularly for production environments.

### Basic System Requirements

| | Memory | Processor | Disk Space | Mi[illegible]s Web Server |
|---|---|---|---|---|
| Minimum | 64 MB | Pentium 4 - 1 GHz (or equivalent) | 30 MB | 27 MB |
| Recommended | 512 MB | Pentium 4 [illegible] GHz | 30 MB | 27 MB |

### Platform Compatibility

| Hardware Architectures | Operating Systems | Communication Protocols | Third-party Encryption Libraries |
|---|---|---|---|
| Intel 32/AMD (x86)<br>Intel 64/AMD64 (x86-64) | Windows 7 32- and 64-bit<br>Windows Server 2008 32- and 64-bit<br>Ubuntu Linux 12.04 32- and 64-bit<br>RedHat Enterprise Linux 6.2 64-bit | UDP<br>TCP | DTLS (OpenSSL) |

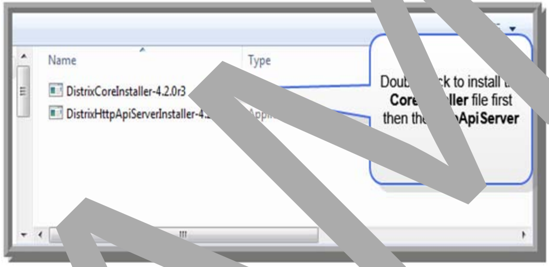
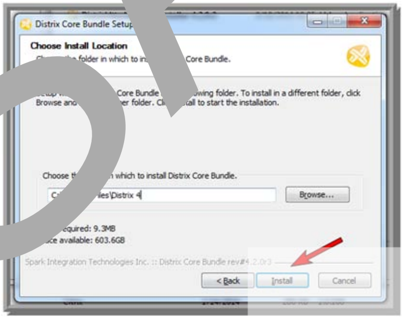
# How to Install Distrix

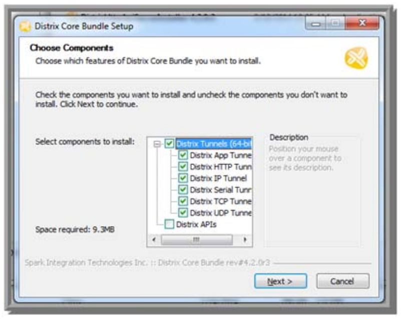
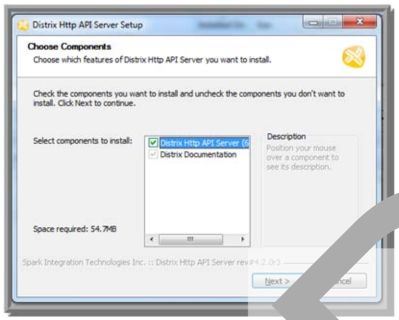
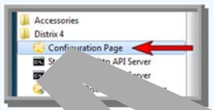
The procedure to install Distrix begins with the download of the installation package from the dowloads section of our website. During the installation process there is the option to install and run Distrix as a service or to run manually for more control as required. (See Appendix B for a detailed list of all the installed file and their exact location.) If you plan to use Ethernet tunnels (Linux Ubuntu only) be sure to read the no about the download of the dependent files.

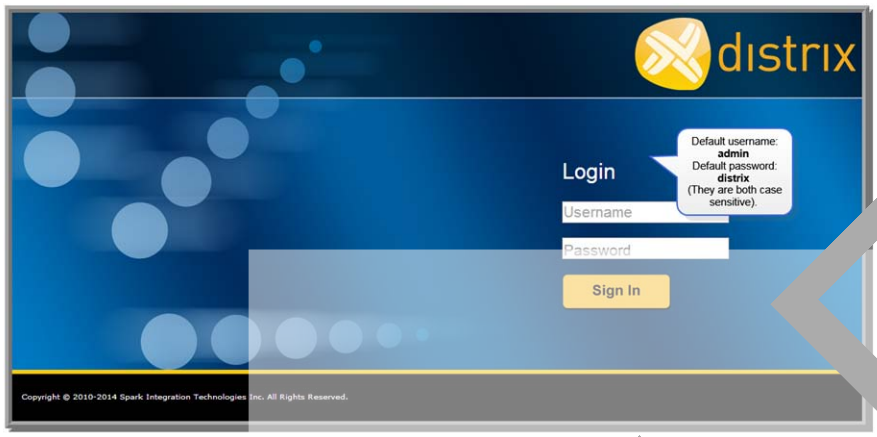
## Installation Procedures

### Windows

The installation procedure is as follows:

| | |
|---|---|
| 1<br><br>**Download** | Log-in to your Distrix account and select from the softw download section:<br><br>▪ DistrixInstallers-4.2.zip<br><br>The installer files are applicable for both 32- a Windows |
| 2<br><br>**Unpack Archive** | When unpacked, the archive contains th e (.exe) files ouble-clic files to start the installation process.<br><br>1. **DistrixCoreInstaller #_Win.e**<br>2. **DistrixHttpApiServ aller_# exe**<br><br><br>Name · Type<br>DistrixCoreInstaller-4.2.0r3<br>DistrixHttpApiServerInstaller-4.<br>Doub ck to install **Core ller** file first then the **ApiServer**<br><br>For eac pplicati the lt folder displays (C://program files/) or browse to a location and ck **stall.**<br><br><br>Distrix Core Bundle Setup<br>**Choose Install Location**<br>folder in which to in Core Bundle.<br>Core Bundle wing folder. To install in a different folder, click Browse and er folder. Cl tall to start the installation.<br>Choose th which to install Distrix Core Bundle.<br>C les\Distrix 4 · Browse...<br>equired: 9.3MB<br>ce available: 603.6GB<br>Spark Integration Technologies Inc. :: Distrix Core Bundle rev#4.2.0r3<br>< Back · Install · Cancel<br><br>The required files are selected by default. Click **Next** to begin and then **Finish** the installation. |

| | |
|---|---|
| |  Distrix Core Bundle Setup<br>**Choose Components**<br>Choose which features of Distrix Core Bundle you want to install.<br>Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.<br>Select components to install: ✓ Distrix Tunnels (64-bi; ✓ Distrix App Tunne; ✓ Distrix HTTP Tunn; ✓ Distrix IP Tunnel; ✓ Distrix Serial Tunr; ✓ Distrix TCP Tunne; ✓ Distrix UDP Tunne; ☐ Distrix APIs<br>Description: Position your mouse over a component to see its description.<br>Space required: 9.3MB<br>Spark Integration Technologies Inc. :: Distrix Core Bundle rev#4.2.0r3<br> <br> Distrix Http API Server Setup<br>**Choose Components**<br>Choose which features of Distrix Http API Server you want to install.<br>Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.<br>Select components to install: ✓ Distrix Http API Server (6; ✓ Distrix Documentation<br>Description: Position your mouse over a component to see its description.<br>Space required: 54.7MB<br>Spark Integration Technologies Inc. :: Distrix Http API Server rev#4.2.0r3<br> |
| 4<br><br>**Browse to Login** | Open the **Start Me**[illegible] click the[illegible]trix configuration page.<br> Accessories<br>Distrix 4<br>Configuration Page<br>API Server<br>Your brow[illegible] automa[illegible] open to https://<FQDN>:4000 (Where FQDN is the fully qualified dom[illegible] name of the Windows machine where Distrix is installed.) Or manually [illegible] the address[illegible] your browser. |
| 5<br><br>**[illegible]nter Pass-w[illegible]d** | When[illegible]rix login page opens in your browser enter these default login credentials:<br>**U**[illegible]**r**: admin<br>**[illegible]assword**: distrix |

| | |
|---|---|
| |   distrix<br>Login<br>Default username: admin<br>Default password: distrix<br>(They are both case sensitive).<br>Username<br>Password<br>Sign In<br>Copyright © 2010-2014 Spark Integration Technologies Inc. All Rights Reserved. The application opens to the **Monitor** > **Topology** scree[illegible]d displays a default nod[illegible] named for the local machine. |

**NOTE**: Windows configuration files save to more than on[illegible]cation; usu[illegible]y in folder[illegible]n from the non-administrative user. The administrator either confi[illegible]s all files or [illegible]s the permissions for each configuration file and/or the configuration file directory, to [illegible] confi[illegible]ation changes by other users.

## Ubuntu Linux

The installation procedure is as follows:

| | |
|---|---|
| **1**<br><br>**Pre-paration** | Prepare a di[illegible]tory [illegible]ich to extrac[illegible] Distrix package files. |
| **2**<br><br>**Dow[illegible] load** | Log-in to your Dis[illegible] account and select from the software download section:<br><br>**Ubunt[illegible]**[illegible]ixPac[illegible]ges-<platform>.tar.gz |
| **3**<br>**Unpack Archive** | When unp[illegible]ed, the archive contains these packages:<br>[illegible]istrix_4.#.#r#-#_arch.deb<br>distrix-core_4.#.#r#-#_arch.deb<br>distrix-dev_4.#.#r#-#_arch.deb<br>distrix-http_4.#.#r#-#_arch.deb<br>distrix-tunnels_4.#.#r#-#_arch.deb |

# User Profile Setup

The administrator ensures that users, permissions and groups are set-up and managed for the appropriate tasks. To facilitate the user profile set-up within the Distrix 4.2 management console application, Distrix provides a basic username and password authentication plugin or you may use a third-party plugin of [illegible]hoice (see Authentication Plugins). For those opting to use an LDAP server, Distrix provides a configur[illegible]e read-only plugin to enable a convenient single sign-on capability. Passwords can only be changed t[illegible]ugh the user interface if the plugin being used permits it (e.g. if ldapauth is being used, the password cha[illegible]e doesn't w[illegible]).

The plugins and configuration files are as follows:

### htpasswdauth

This plugin authenticates users against an htpasswd file. It should be passed the filesystem path to [illegible]on-figuration file as an argument (using the userAuthArgs HTTP server setting[illegible]he configuration file sho[illegible]on-tain the following parameters:

| | |
|---|---|
| `path` | The filesystem path to the htpasswd file for which t[illegible]ate aga[illegible] |
| `groups` | A dictionary containing usernames as keys a[illegible] list of secu[illegible] groups tha[illegible]ply to the user as values. |
| `duration` | The duration in seconds for which ne[illegible]kens sho[illegible]. |

#### Configuration File

```
{
    "groups": {
        "userA": ["Configura[illegible]n", "[illegible]ing"],
        "userB": ["Security"]
    },
    "path": "./et[illegible]users.ht[illegible]",
    "duration": 36[illegible]
}
```

### ldapa[illegible]h

The Distrix user interface w[illegible]ks with an LDAP directory as follows:

| | |
|---|---|
| **Read Only** | Authenticates users against an LDAP server but **doesn't modify** LDAP entries. |
| **Maps Users Only** | Maps users against permissions using its configuration file but **doesn't pull group/permission information** from LDAP. |
| **Add User** | An LDAP user cannot be added through the Distrix interface unless they already exist in the LDAP database. |

| | |
|---|---|
| | In the Distrix "Add User" entry form leave the password field blank, as it's ignored. Adding the user just adds an entry into the groups dictionary of the configuration file which associates the specified permissions for that user. |
| **View User List** | The Distrix user list only displays users with entries in the plugin configuration and not all of the users in the LDAP server. |
| **Delete User** | Deleting a user from the Distrix list removes their as[illegible]iated entry from the plugins configuration file and does not delete them fr[illegible] the LDAP s[illegible]er. |

This plugin authenticates users against an LDAP database. The file system path sh[illegible] be pass[illegible] to a configuration file as an argument (use the userAuthArgs HTTP ser[illegible]er setting). The configu[illegible]n [illegible] should contain the following parameters:

| | |
|---|---|
| `server` | The IP address or hostname of the L[illegible]server. D[illegible]ult: ldap.exa[illegible]e.com |
| `port` | The port number of the LDAP s[illegible]er. Defau[illegible]389. |
| `baseDn` | The base path containing us[illegible] Default: o[illegible]people,dc=example,dc=com. |
| `uidPattern` | A string which is pre[illegible]ded to th[illegible]n when binding to the ldap server. Should contain "%s", which is replac[illegible]h the use[illegible]me. Default: uid=%s |
| `groups` | A dictionary containing [illegible]ern[illegible]es as key[illegible]nd a list of security groups that apply to the user as [illegible]alues. |
| `duration` | The dura[illegible] in [illegible]nds fo[illegible]hich new tokens should last. |

### Configurat[illegible]

```
{
    "ser[illegible]r":"[illegible]dap.exa[illegible]mple.com",
    "base[illegible]"ou=people,dc=example,dc=com",
    "uidPa[illegible]ern":"uid=%s",
    "[illegible]roups[illegible] {
         "userA": ["Monitoring"],
         "userB": ["Configuration", "Security"]
    [illegible]
    [illegible]duration": 3600
}
```

# Application Tunnel Connector

Distrix's application tunnel connector enables building client applications that connect directly to a local Distrix gateway without needing to manage a socket connection.

Internally, App tunnel clients use a TCP socket to initially discover the local Distrix gateway and then use a faster inter-process communication (IPC) mechanism to transfer data. The TCP port that the Distrix gateway listens on can be controlled by setting the “port” value in the AppTunnel.cfg configuration file located at the following:

**Linux**

```
/opt/distrix/etc/AppTunnel.cfg
```

**Windows**

```
C:\Program Files\Distrix 4\etc\AppTunnel.cfg
```

**NOTE**: App tunnel clients can only connect to a Distrix gateway on the same machine. For security reasons, app tunnel clients must be running as the same user as the Distrix gateway (or as root/administrator) in order to connect.

## Guidelines

| | If... | Then... |
|---|---|---|
| **Installation** | allowing App tunnel client connections | the Distrix gateway must be running the App.tunnel plugin which is included with the core Distrix 4.1 installer. |
| | developing new App tunnel clients | the Distrix developer package must be installed to provides the necessary header files and libraries to build App tunnel client applic-ations. |
| **Building a Client** | To use the App tunnel functionality the client application must include<br>▪ the App tunnel header file, and<br>▪ link in the client library.<br>Both the header file and the client library are included in the Distrix developer’s (dev) package.<br>They are installed at the following locations: | |

```
    DX_Stream l_stream = DX_broadcast_open(&l_config, 0);
    if(!l_stream)
    {

        printf("Failed to create broadcast stream\n");
        return -1;

    }

     char *l_data = "broadcast test";
    for(;;)
    {

        int l_sent = DX_send(l_stream, l_data, strlen(l_data),
        0); printf("sent %d\n", l_sent);
        if(l_sent < 0)
        {

            return 1;

        }
        DX_msleep(50);

    }

    return 0;

}
```

### Response Example

```
#include "dx_apptunnel.h"
#include "Distrix/Abstractions/[illegible]/Time.h"

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int main(int argc, char **argv)
{

    unsigned int l_port = 0;
    if (argc > 1)
    {

        l_port = atoi(argv[1]);

    }
```

```
     DX_connect(l_port);

     DX_TunnelConfig l_config;
     memset(&l_config, 0, sizeof(l_config));
     l_config.name = "apptest_broadcast_recv";
     l_config.id = "broadcast";
     l_config.metadata = "hello";

      DX_Stream l_stream = DX_broadcast_open(&l_config, 0);
     if(!l_stream)
     {

          printf("Failed to create broadcast stream\n");
          return -1;

     }

     char l_buffer[256];
     unsigned l_recvCount = 0;
     uint64_t l_startTime = DX_[illegible]TimeMs();
     while(l_recvCount < 10 && (D[illegible]etTi[illegible]s() - l_startTime) < 20000)
     {

          int l_recv = DX_r[illegible]e[illegible](l_str[illegible] l_buffer, sizeof(l_buf-
          fer)-1, 2000);
          printf("[illegible]eceived %d[illegible]", l[illegible]ecv);
          if(l_r[illegible]0)
          {

               l_buffe[illegible]l_recv][illegible]'\0';
               printf(" [illegible]s\n", l_buffer);
               [illegible]recvCoun[illegible]

          [illegible]
          e[illegible]e[illegible](l_recv == -2)

          {

               [illegible]rintf("stream deleted\n");
               return -1;

          [illegible]

     [illegible]

     return (l_recvCount >= 10 ? 0 : 1);

}
```

[How To Use Interface]

[Configure]

[Provision]

[Monitor]

# User Interface

## Security Certificates Introduction

A Distrix network uses public-key encryption for its security, which is implemented through the use of certificates. Public-key encryption (also called asymmetric encryption) involves using a pair of keys (a public and a private key) in association with an entity requiring either electronic authentication of its identity or the ability to sign or encrypt data. While each public key is published, the corresponding private key is kept secret and hidden from public view, as any data encrypted with the public key can be decrypted only with the private key.

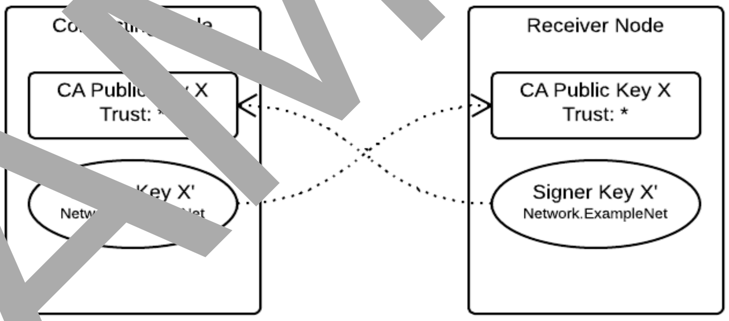
### Important Facts

- Any algorithm supported by OpenSSL can be employed to secure a Distrix network.
- The Distrix method generates Elliptic Curve Cryptography (ECC)-based CA and certificates.
- ECC is a newer algorithm which uses a much shorter key than the RSA solution but offers the same level of protection.
- The Distrix method is a much simpler procedure in both generation and distribution compared to using OpenSSL.

### How it Works

A **Certificate Authority** is the master certificate (a file) with both a private and public key; and is used to generate all other certificates. The public key of the CA cert validates the CAs digital signature on the Signer certificate being presented in a trust request. In other words, if matched and authenticated, it treats the certificate as a "letter of introduction" from that CA, determining that the Signer certificate is valid and then proceeds with the request.

#### Example



The administrator sets-up security for the network by applying certificates to the nodes: each one must have a certificate and different types can either identify networks or capabilities. In order for a connection between Distrix nodes to be made, bi-directional trust must be established between the nodes. The node establishing the connection sends its Signer certificate to the receiving node which then validates the Signer certificate against the CA certificate resident on that node. However, bidirectional trust is required for a connection to be established so once the connecting node's Signer certificate has been validated, the receiving node provides its Signer certificate and evidence for similar validation.

## Flexible Security Models

Distrix provides flexibility in how you plan and set-up your network security; ranging from very a simple (less secure) mode to a multi-layered and more robust approach. The Distrix installation package includes a default Certificate Authority Certificate, the master certificate (or file) with both a private and public key, which can be used “out-of-the-box” to implement basic security.

This gives you time to decide whether a

- "Basic Security Mode" on page 54 or
- "Full Security Mode" on page 57 (enhanced)

is required, as you deploy your network structure: determining common names, permissions and tasks. The advanced (Full) security setting requires Signer Certificates on a per-task basis, and allows the granularity of corresponding CA Certificates to any level.

## Distrix Default Certificates

A Distrix default CA cert (default.ecc.pub) and a default Signer cert (defaultnetwork.ecc) is installed on nodes as you set-up your network in a basic security mode. To enhance security, the network administrator replaces the default certificates with certificates generated with one of these methods:

- "Distrix Method To Generate Certificates" on page 57
- "OpenSSL Method To Generate Certificates" on page 59

( Reading Resource: **Public-key Cryptography**)

## CA Certificates

A Distrix default CA cert (default.ecc.pub) and a default Signer cert (defaultnetwork.ecc) are installed on nodes as you set-up your network in a basic security mode. Though this basic "out-of-the-box" set-up is convenient, for enhanced security the network administrator should replace these default certificates and generate their own using either:

- "Distrix Method To Generate Certificates" on page 57
- "OpenSSL Method To Generate Certificates" on page 59

The Certificate Authority (CA) cert has a private and a public key and both are used to generate Signer certificates. The public key of the CA cert must be distributed to all nodes. It's recommended to use the same CA cert to create the Signer certs, as a same name configuration keeps the match-up and verification process simple.

## Install a CA Certificate

1. Go to **Configure** > **Distrix Nodes**. Next to the listed **Node Name**, on the **Actions** column click **Security** to go to **Configure** > **Certificates**.
2. On the **CA Certificates** tab go to **Add CA cert** and **Choose File** to browse for the public key of your CA cert located in a specified directory on your system. Once selected, the key displays in the **File** field.
   - **Name**–There is the option to type a new name otherwise it defaults to the file name.
   - **Trust**–For most purposes leave it set to the default (*) which applies to all.

---

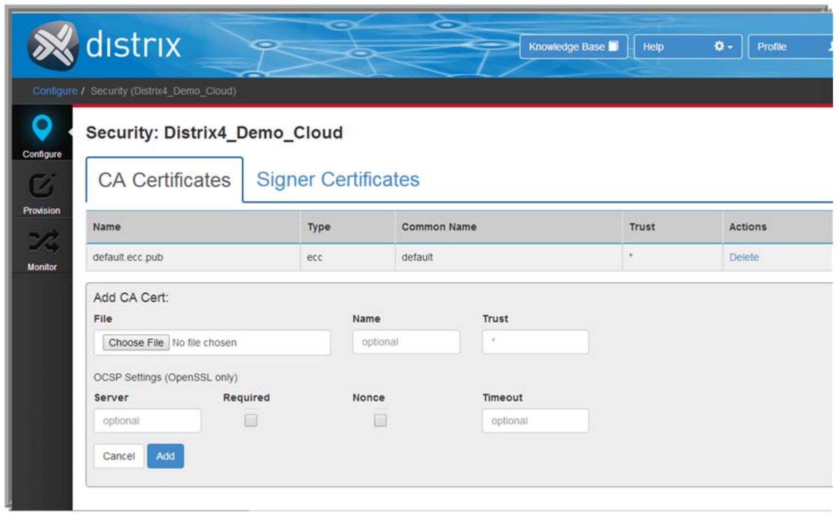
"Online Certificate Status Protocol (OCSP)" on page 62 (OpenSSL only) - Using these security settings type the following:

| | |
|---|---|
| `url` | This is the URL of the OCSP responder. |
| `required` | A boolean: if false, the certificate is accepted if none of the listed servers can be contacted. Default is true. (If true, behavior is as above.) |
| `nonce` | This indicates whether or not the OCSP request should use nonce (a number or string used only once). Optional: if omitted it defaults to false. (The parameter has changed from "useNonce" to "nonce".) |
| `timeout` | This is the timeout, in milliseconds, on waiting for an OCSP response. (Defaults to 1000 ms. = 1 second.) |

---

3. Click **Add**. The CA certificate displays in the list as being applied to the node, along with these details:
   - **Name**–The new name otherwise it defaults to the file name.
   - **Type**–The type of certificate, whether OpenSSL or Elliptic Curve Certificate (ECC).
   - **Common Name**–The common name of the generated CA cert.
   - **Trust**–Usually set to the default (*), which applies to all.

NOTE: The Distrix default CA cert can be left in the list or deleted.



Security: Distrix4_Demo_Cloud

CA Certificates | Signer Certificates

| Name | Type | Common Name | Trust | Actions |
|---|---|---|---|---|
| default.ecc.pub | ecc | default | * | Delete |

Add CA Cert:

File: Choose File No file chosen — Name: optional — Trust: *

OCSP Settings (OpenSSL only)

Server: optional — Required: ☐ — Nonce: ☐ — Timeout: optional

Cancel | Add

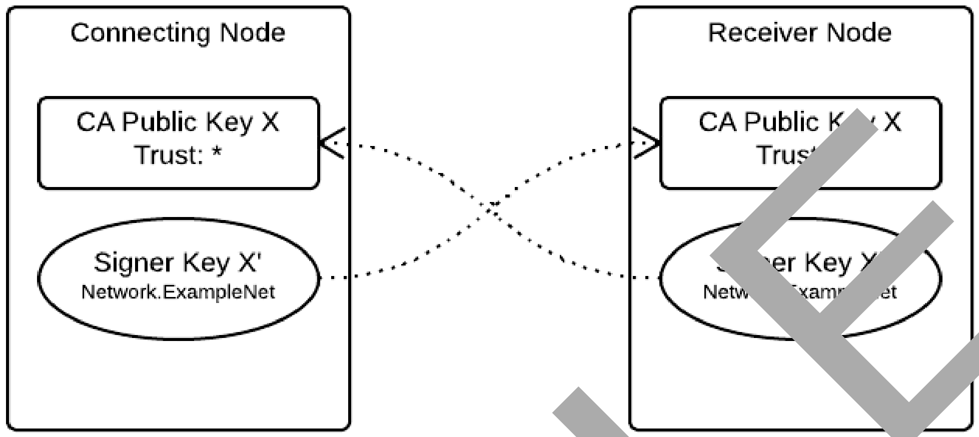
## Signer Certificates

Signer certificates are used for encryption and can be used f ore gran r control to distinguish and separate different permissions (i.e. groups, companies, tions) wi ful security mode. A Certificate Authority (CA) cert must be available and when generated by x or Ope , has two separate components: the public key usually with .pub suffix, and the private ke (S Key Exa ). Both keys are needed to create a Signer cert but the private key is not distributed, only t pub key of the A certificate is present on each node. The Distrix installation package includes a Signer ert (di ved as defaultnetwork.ecc and has a common name of Network.Distrix) and in curity mo the sa certificate is used for everything, whether configuring nodes, assigning co ction rovi ning tunnels.

**NOTE**: Signer certs g ted by the Distrix thod include both the private and public key in a single file. Decide in advance hich ficate genera method to use before creating Signer certs as the methods and types can e combin

### How it Wor

In order a connection betwee istrix nodes to be made, bi-directional trust must be established between the no s. (Commu can t place in a Distrix network without trust established between every node and e other e; as lo s there is trust between each pair of "adjacent" nodes on the route.) The node establis e connection nds its Signer certificate to the receiving node which then validates the Signer certificate against the CA tificate resident on that node. However, bidirectional trust is required for a connection to be establish so once the connecting node's Signer certificate has been validated, the receiving node provides its ner certificate and evidence for similar validation.



### What is a Common Name?

The common name (CN) is intrinsic to the ce   cate so that w  n a certifica   s received, the common name is received along with it. The common name be   with a d   cription of how it is being used (i.e. Network, Monitoring, etc.) and ends with a netw   name (i.   Mo   ring.Distrix). In Full security mode (and in Basic) there might be a need for different Sig   ts under   rent common names if the trust value specifies a particular pattern on the corresponding CA   or   mple, if   something other than "*" then the CN of the Signer certificate must match it (i.e. if th   ust   e of the   were "*.Distrix", the Signer cert's CN would have to be "<something>  Distrix").

#### Advanced Settings

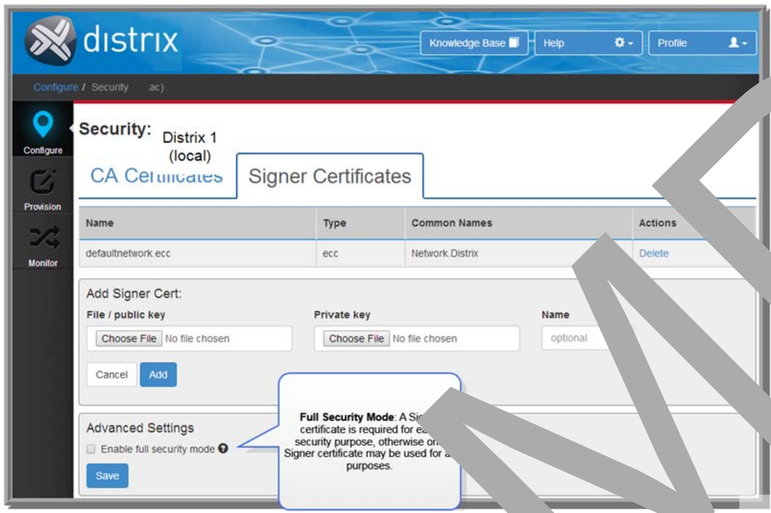
**Full Securi**   –If a full secur   ode is required, then multiple Signer certs are required for each security purpose.   eck t   to enable.   k **Save**. Otherwise leave unchecked.

### ll a Signe   ertificate

1. Go to **C   figure** > **Distrix Nodes**. Next to the listed **Node Name**, on the **Actions** column click **curity**   go to **Configure** > **Certificates**.
2. On   **Signer Certificates** tab go to **Add Signer Cert** and **Choose File** to browse for the public key   nd private key of your Signer cert located in a specified directory on your system. Once selecte   e keys display in the **File** fields
   - **Name**–There is the option to type a new name otherwise it defaults to the file names.
3. Click **Add**. The Signer certificate displays in the list as being applied to the node, along with these details:
   - **Name**–The new name otherwise it defaults to the file name.
   - **Type**–The type of certificate, whether OpenSSL or Elliptic Curve Certificate (ECC).
   - **Common Name**–The common name of the generated Signer cert. When using Full Security mode, the common name is used for two purposes:

  - Comparison against the trust field of the associated CA cert.
  - Association between actions requiring Signer certs (monitoring, security, provisioning, networking, and configuration) and their respective certificates.
- **Trust**–Usually set to the default (*), which applies to all.

**NOTE**: Signer certs generated by the Distrix method include both the private and public key in a single file. Upload that single file to the **File/public key** field.



| Name | Type | Common Names | Actions |
|---|---|---|---|
| defaultnetwork.ecc | ecc | Network.Distrix | Delete |

Add Signer Cert:

File / public key: Choose File No file chosen

Private key: Choose File No file chosen

Name: optional

Cancel Add

Advanced Settings

Enable full security mode

Save

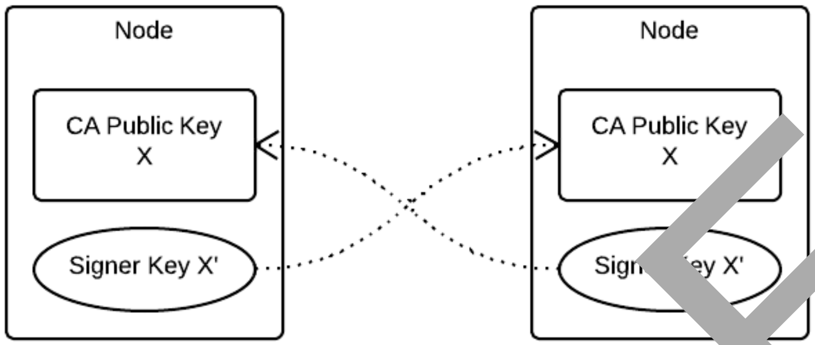
**Full Security Mode**: A Si[illegible] certificate is required for e[illegible] security purpose, otherwise o[illegible] Signer certificate may be used for a[illegible] purposes.

## FIPS-140 Security Module

The 140 series of **Federal Information Processing Standards** (FIPS) are government security standards that specify requirements for cryptography modules for both software and hardware components. The Distrix FIPS security module (FIPS.security) plugin contains an embedded FIPS 140-2 validated cryptographic module (the OpenSSL FIPS Object Module v2.0.5). When the FIPS.security module is installed the only cryptographic implementations Distrix uses are those provided by the FIPS Object Module. The installer places the FIPS.security module into the Distrix "lib" directory and Distrix automatically loads it on start-up and prevents the use of any other security modules.

When the FIPS.security module is installed, Distrix calls the FIPS_mode_set() function on startup.

| If... | Then... |
|---|---|
| the FIPS_mode_set() function call | the string “*** IN FIPS MODE ***” is printed to the Distrix log. |

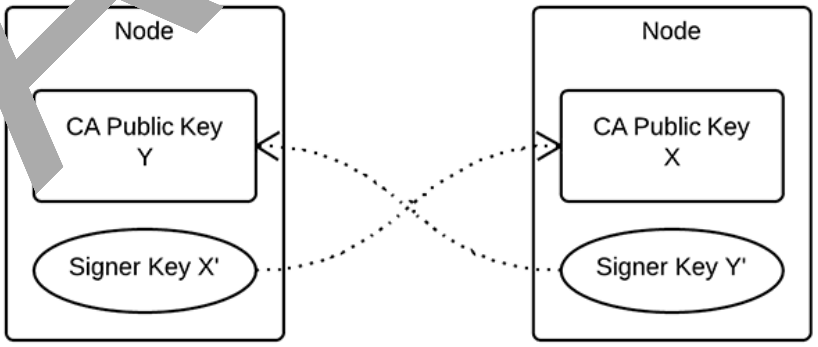


Every node in the network is identically configured (as far as security is concerned). Every node has the same CA public key, and every node has an identical signer certificate, generated from that CA. As every node has the same CA, the same signer certificate will work in establishing trust for every node.

The default configuration established when Distrix nodes are installed is similar but not identical: each signer certificate is generated at install, using the installed CA. Since all of the CAs are identical, the signer certificates are all sufficient to establish trust (in basic mode) with any other freshly-installed node.

### Reciprocal Configuration

This configuration bridges two separate logical groups, and demonstrates how bidirectional trust is established in Distrix node-to-node communication. (NOT recommended for production environments.)

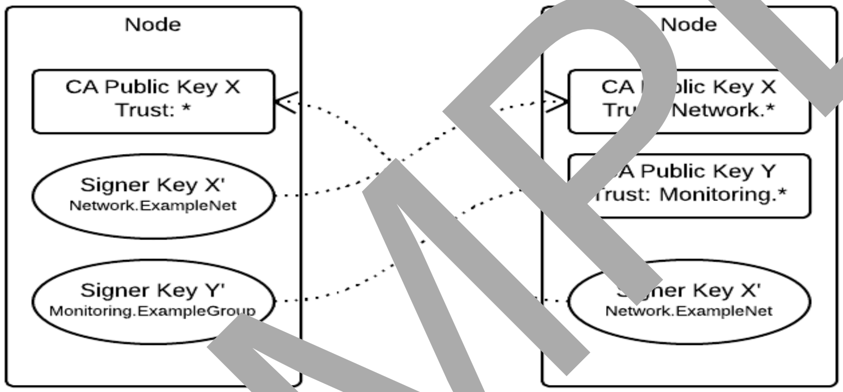


If the node on the left is establishing communication with the node on the right (and possesses only the appropriate X' signer certificate), it will send the certificate and associated evidence to the node on the right. The node on the right checks the certificate against its CA, determines that X' was derived from X, and sends its own signer certificate (again, assuming that this is the only signer certificate it has) back for the reciprocal operation.

At that point, communications are established. If X' and Y' were not the only signer certificates available to the left and right nodes, respectively, then the choice of common name when generating the certificate becomes important. A certificate intended to establish a connection between Distrix nodes should have a common name in the format "<usage>.<networkname>", where the suffix is the network name shared by the two nodes.

## Segregated CAs

In this configuration, different Signer certs (and CAs) are employed for different tasks asymmetrically.



Even in Basic mode, Signer certificates are chosen based on their common name, and then arbitrarily if the necessary common name is absent. In this case, the node on the left needs different signer certificates to establish communications and allow monitoring from the API respectively, because the trust settings for the node on the right's CAs only accept specific common names.

As the Basic mode is being employed, and the node on the left is configured to trust any signer certificate signed by CA "X", Signer key "X" works for all requests of the node on the right.

## Full Security Mode

### Use Multiple CA Certs & Signer Certs

The advantage to using Full Security mode with multiple CA certificates and Signer certificates is the enhanced security that can be obtained by segregating the permissions required for actions on each node. The disadvantage is that it takes more pre-planning and adds complexity in multiple layers and tiered authorizations which must be rigorously tracked. As a first step in this advanced mode, consider keeping the security simple with a single CA cert but create multiple Signer certs that are applied to every node for each security purpose. The trust is set to a wildcard (*) or for more granular control you could specify a trust with a wildcard and add a company or product name, for example *.Distrix. Then if required you could partition by networks, for example, the common name may be in the format "<usage>.<networkname>" where the suffix is the network name shared by the two nodes.The common name is the important value.

**RECOMMENDATION**: The security granularity can be increased by using a different Signer cert for every single task on every node and distributing the Signer certs as appropriate to the other nodes that are meant to be connected. The common types of certificates to manage authorized tasks on a network are: network (or communications), configuration, provisioning, security, monitoring, and tunnels. Every node must have a Signer cert for each function (i.e. monitoring.*, provisioning.*).

**NOTE**: There may be the rare occasion when the Distrix default CA cert is either deleted or ignored and there isn't any security setup and trust is assumed on each node. For example, a quick test network is needed and taking time to setup certificates on the nodes isn't warranted. Operating a network in this mode is NOT recommended.

## Distrix Method to Generate Certificates

Certificates are used to secure communications throughout and in the administration of a Distrix network. The Distrix installation package includes a default CA certificate (default.ecc.pub) and a default Signer cert (defaultnetwork.ecc) which is used “out-of-the-box” to enable the set-up of your network in a basic security mode. To enhance security, the network administrator replaces these default certificates and can either use Distrix's method to generate an Elliptic Curve Certificate (ECC) or use an [OpenSSL method].

**NOTE**: Decide in advance which certificate generation method to use before creating Signer certs as the methods and types can't be combined.

### Generate a CA Certificate

The Distrix executable file includes a function to generate CAs employing ECC. Use the command line and enter the parameter `create-ca` as follows:

#### Syntax

```
opt/distrix/bin/distrix create-ca <filename> <common name> <expiry (days from now)>
```

### Example

```
opt/distrix/bin/distrix create-ca exampleca.ecc ExampleNetwork 730
```

### Parameters

| | |
|---|---|
| `filename` | The filename of the generated CA cert. Two files, filename (the private key), and filename.pub (the public key) are produced. |
| `common name` | The common name of the generated CA cert though it's not relevant to the cert for its use in Distrix. |
| `expiry` | A whole number representing the number of days before the certificate expires. |

## Generate a Signer Certificate from a CA Certificate

Once a CA cert is available, Signer certs can be derived from it. Signer certs generated by the Distrix executable include both the private and public key in a single file, so only that file is required. Go to the command line and use the parameter `create-cert` as follows:

### Syntax

```
opt/distrix/bin/distrix create-cert <filename> <ca private key> <common name> <expiry
(days from now)>
```

### Example

```
opt/distrix/bin/distrix create-cert examplesigner.ecc exampleca.ecc Network.Distrix
365
```

### Parameters

| | |
|---|---|
| `filename` | The filename of the generated cert. One file, a combined public and private key will be generated with this filename. |
| `ca private key` | The path of the private key of the CA from which this cert is to be derived and with which trust is to be established. (This should be the file previously generated that doesn't have the .pub suffix.) |
| `common name` | The common name of the generated Signer cert. When using Full Security mode, the common name is used for two purposes: ▪ Comparison against the trust field of the associated CA cert. ▪ Association between actions requiring Signer certs (monitoring, security, provisioning, networking, and configuration) and their respective certificates. |
| `expiry` | A whole number representing the number of days before the certificate expires. The value should be less than its parent CA cert. |

### OpenSSL Method To Generate Certificates

Distrix provides the option to use OpenSSL, X.509 certificates for trust and encryption. Distrix supports the use of "Online Certificate Status Protocol (OCSP)" on page 62 ) This section describes their method of generation, which involves a number of steps. Before proceeding with the generation, it is important that a separate directory is made; there are multiple files and a hierarchy when creating the Signer certs. A CA certificate needs to be available or created first in order to create subsequent Signer certs.

(See OpenSSL for information on the command line tool and configuration file.)

## Generate a CA Cert

### Syntax

```
openssl req -x509 -nodes -days <days until expiry> -subj <subject string> -newkey
<alg:bits> -keyout <path-to-keyfile> -out <path-to-cert>
```

### Example

```
openssl req -x509 -nodes -days 365 -subj /C=CA/ST=British Columbi-
a/L=Vancouver/CN=GeneriCo.com/ -newkey rsa:2048 -keyout newca/cacert.key -out new-
ca/cacert.pem
```

### Parameters

| | |
|---|---|
| `days until expiry` | Similar to Distrix generated certificates, this is a whole number representing the number of days before the certificate expires. |
| `subject string` | A list of name attributes of the format `'/type1-1=value1/type2=value2/.../'` |
| `alg:bits` | Specification of the encryption algorithm and key size (in bits) of the new key (e.g. rsa:1024). RSA is a popular algorithm used with OpenSSL. Popular key lengths are 1024, 2048, and 4096 bits (longer keys are more secure but slightly more expensive, computationally). |
| `path-to-keyfile` | The path, relative or absolute, to which the private key of the new CA cert should be written. |
| `path-to-cert` | The path, relative or absolute, to which the public key of the new CA cert should be written. |

## Generate a Signer Certificate from a CA Certificate

This is a multi-step process so both the key and CA certificate should be in a common folder that is subordinate to the folder in which the other certificates are generated.

1. Before the CA cert can be employed, these items must be added to the folder:
    - An empty database file.
    - A serial file containing: 01.
    - An OpenSSL configuration file.

### Configuration File Example

```
[ ca ]
default_ca                    = CA_default          # The default ca section
[ CA_default ]                = newcerts            # path to new certificates
new_certs_dir

database                      = cafiles/index.txt   # path to an (initially)
                                                    empty file

default_md                    = md5                 # md to use

serial                        = cafiles/serial      # path to the serial file

policy                        = policy_any

default_days                  = 365

[ policy_any ]
   countryName                    = supplied

   stateOrProvinceName            = optional

   organizationName               = optional

   organizationalUnitName         = optional

   commonName                     = supplied

   emailAddress                   = optional
```

2. Generate the key that forms the foundation of the Signer cert.

### Syntax

```
openssl genrsa -out <keyfile> <bits>
```

### Parameters

| | |
|---|---|
| keyfile | The path to the generated key. |
| bits | The length of the key in bits. |

### Example

```
openssl genrsa -out Network/Network.key 2048
```

3. Request a new certificate from the CA cert.

### Syntax

```
openssl req -new -nodes -subj <subject string> -key <path-to-key> -out <path-to-cert>
```

### Parameters

| | |
|---|---|
| `subject string` | A list of name attributes of the format `'/type1-1=value1/type2=value2/.../'`<br>The most important attribute here is the common name (CN) attribute which (as in Distrix generated certificates) is used for CA trust validation and for per-action validation in Full security mode. |
| `path-to-keyfile` | The path to the generated key (in step 2). |
| `path-to-cert` | The path to the generated cert. |

### Example

```
openssl req -new -nodes -subj '/C=CA/ST=British Columbia/L=Vancouver/CN=Network/' -key
Network/Network.key -out Network/Network.csr
```

4. The Signer cert must be formally certified by the CA cert.

### Syntax

```
openssl ca -batch -cert <path-to-ca-cert> -keyfile <path-to-ca-key> -in <path-to-cert>
-config <config-path> -out <path-to-certified>
```

### Parameters

| | |
|---|---|
| `path-to-ca-cert` | The path to the CA certificate. |
| `path-to-ca-key` | The path to the CA key. |
| `path-to-cert` | The path to the requested certificate. |
| `path-to-ca-key` | The path to the CA configuration file (in step 1). |
| `path-to-certified` | The output path for the fully certified certificate. |

### Example

```
openssl ca -batch -cert newca/cacert.pem -keyfile newca/cacert.key -in Net-
work/Network.csr -config newca/config.cnf -out Network/Network.pem
```

These keys are uploaded in **Configure** > **Distrix Nodes** > **Security** > **Add Signer Cert**

| | |
|---|---|
| `Network/Network.key` | Private Key |
| `Network/Network.pem` | Public Key |

## Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs in a public key infrastructure (PKI). Since [illegible] OCSP response contains less information than a typical CRL (certificate revocation list), OCSP can us[illegible]etworks and client resources more efficiently.

**NOTE**: OCSP is supported for OpenSSL generated certificates but not for Distrix gene[illegible]d cer-tificates.

A list of OCSP responders can be specified in Security.cfg to validate certi[illegible]ates presented by conne[illegible] nodes. If multiple entries are provided, Distrix attempts a check with each se[illegible] At least one server mu[illegible] return an "ok" response for the connection to be accepted and any "revoked" res[illegible]se causes the c[illegible]nection to be rejected.

There are four components to an OCSP entry:

| | |
|---|---|
| `url` | This is the URL of the OCSP [illegible]onder. |
| `required` | A boolean: if false th[illegible]ificate is a[illegible]pted if none of the listed servers can be contacted. Default[illegible]o [illegible] (If true, [illegible]avior is as above). |
| `nonce` | This indicates whether o[illegible]ot th[illegible]CSP request should use nonce (a number or bit st[illegible]sed only onc[illegible] Optio[illegible] if omitted it defaults to false. (The para-meter ha[illegible]han[illegible] from "u[illegible]Nonce" [illegible]o "nonce".) |
| `timeout` | This is the tim[illegible]t, in milli[illegible]ds, on waiting for an OCSP response. (Defaults to 100[illegible] = 1 second.) |

OCSP settings are a comp[illegible]ent of a c[illegible] entry in Security.cfg, rather than a separate entry, as in the fol-lowing example:

```
caCert[illegible]pe: "opens[illegible]
fil[illegible]"ca.pem",
tr[illegible]t: "*",
oc[illegible]: {req[illegible]d: fa[illegible]e, u[illegible]l: "example.com", nonce: true, timeout: 1000}]
```

### Targets

Targets are the destinations to which connections are made. They are indicated by an array of strings spe-cifying the remote target nodes to which they connect. Link modules without any targets configured may accept incoming connection attempts depending on other properties but won't attempt to initiate connections themselves. Each entry in the array may be

- a string (just the target destination)
- an object containing a "target" string
- an optional "network" string, and
- an optional "config" object containing configuration specific to the link that is created for the target.

The following shows the complete syntax for a target specification along with som[illegible]tring examples.

### Syntax

```
"<target IP address>[:port]/[interface]/[interface a[illegible]ess]"
```

### String Example

"10.10.0.20"

"10.10.0.20:5000"

"10.10.0.20:5000/eth1"

"10.10.0.20:5000/eth1/10.10.0.50"

> This last string example [illegible]ds as: "C[illegible] to the Distrix node at 10.10.0.20, on port 50, using the the int[illegible] address [illegible]0.10.0.50 on eth1".

"10.10.0.20//10.10.0.50"

## Add Targets

The selected node is displayed [illegible]ame and th[illegible] default network.

1. C[illegible] **Target** and ty[illegible] in the **Target** name (or IP address).
2. Ch[illegible]se the **Ty**[illegible]DP or TCP). Click the drop-down arrow to select from a list.
3. The[illegible]s an [illegible]ion to ty[illegible] in the **Network** name if required for specific identification (<name>.<network [illegible]).

**NOTE:** Leave th[illegible]arget name without the optional network or group name and it acts as a wildcard ([illegible] be a[illegible]ied to any target (e.g. distrix.com applies to multiple targets or use distrix.networkone.c[illegible] to apply specifically to networkone.)

**NOTE:** It's not possible to add two targets with the same IP address but you can add a target with a hostname that resolves to the same IP as an existing target. If a hostname and a matching IP are provided as targets (or multiple hostnames targeting the same IP addresses), only one will connect. Since the association between a hostname and IP may change, this does not necessarily mean that this configuration is invalid.

Tunnels Introduction

Create Tunnels | Endpoints

View Tunnel List

# Provision

# Create Tunnels

Tunnels are established by first identifying a device and the required connections (i.e. endpoints) and then determining the network interface type needed to transport the data (e.g. UDP). The data exchange is bidirectional: specifying which ports are listening, receiving and acknowledging the receipt and return of data. The tunnel acts like a cable connecting two networks and is oblivious to the content of the data being transported. Policies can be added to the tunnel endpoints for data analysis or more granular control. Tunnels are configured using the Distrix user interface or the tunnel.cfg file. (See Appendix "Tunnels Configuration" on page 142) (Also see Tunnel Instance Configuration in the Control API Guide) Access a detailed list of attributes and configuration parameters using the drop-down text box next to each tunnel type.

### Create a Tunnel

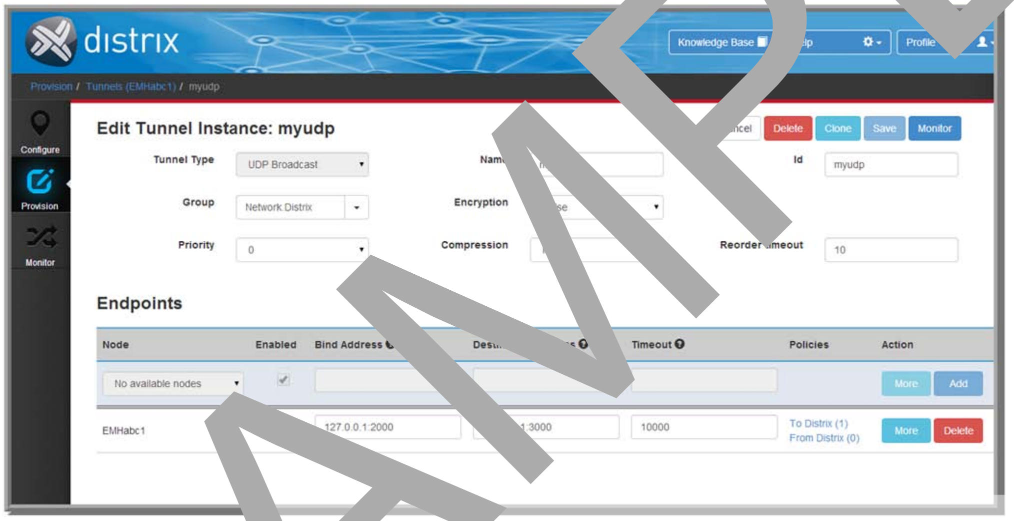
1. Determine the tunnel **Type** to be created. Click the drop-down arrow to select from a list.
   - Ethernet*, HTTP, TCP, UDP Point to Point, UDP Broadcast [illegible], Serial
     *Distrix on a Windows platform does not support Ethernet Tunnels.
2. Type the following tunnel details:
   - **Name**–This human-readable name identifies the tunnel and is used only as a reference.
   - **Group**–Click the drop-down arrow to select from a list. Or type the name of the security group to be applied to this tunnel. This specifies which certificate is being employed for communication by this instance.
   - **Encryption**–Click the drop-down arrow to select from a list. Set to True if encryption is being used, otherwise set to False. (An encrypted tunnel instance does not connect to an unencrypted interface.)
   - **ID**–This unique string identifies the tunnel. When ID's match the endpoints can be joined.
   - **Priority**–Click the drop-down arrow to select from a list. The order in which traffic backlogs are sent: the priority range is from -31 (highest priority) to 31 (lowest). If congestion isn't expected on the tunnel, leave the default value of zero. A lower value, in the negative, is actually a higher priority, so for example a value of -5 is sent before +5.
   - **Reorder Timeout**–Type the timeout number in milliseconds to wait for missing packets from Distrix before sending them to the destination or 0 (default value) to not reorder any packets. This only applies to packets with the "reorder" attribute (e.g. TCP packets sent over an Ethernet tunnel).
   - **Compression**–Click the drop-down arrow to select from a list. Set to True if compression is being used, otherwise set to False.
3. Click **Create**. A message displays that the tunnel was created successfully. The screen opens to enable editing of the new tunnel.

**TIP**: Use the **Clone** button to simplify building multiple tunnels that have similar attributes.

## Create Endpoints

When the tunnel has been created add the endpoints, the specific logical points where external data can enter or exit the Distrix network. There may be many endpoints or none at all on each Distrix node or there may be many endpoints on a given Distrix node for each supported tunnel type.

1. Select a node. Click the drop-down arrow to select from a list.Check the box if the e point is enabled.
2. Next to the selected node, type in the attribute values required by the tunnel t The endp attribute fields display according to the tunnel type that has been selected. Clic **re** to cess additional attribute fields.
3. Click **Add**. The new end-point displays in the list by Node name and attributes.
4. Click **Save**. This only saves the properties defined for that spec endpoint. Click **Save** a p of the page to save all edits. A dialog box displays to confirm the **e** if navigating away fro the page and there are unsaved changes.



Edit Tunnel Instance: myudp

Tunnel Type: UDP Broadcast
Nam
Id: myudp
Group: Network Distrix
Encryption: se
Priority: 0
Compression
Reorder meout: 10

Endpoints

| Node | Enabled | Bind Address | Dest | Timeout | Policies | Action |
|---|---|---|---|---|---|---|
| No available nodes | ✓ | | | | | More Add |
| EMHabc1 | | 127.0.0.1:2000 | 1:3000 | 10000 | To Distrix (1) From Distrix (0) | More Delete |

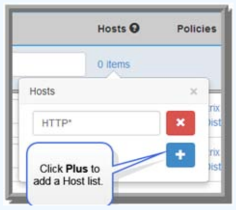
## Tunnel ypes & Endpoint Attri tes

There e several typ hic an be tunnelled through Distrix 4.2, their description, attributes and parameters ows:

### HTTP

The HTTP tunnel can only be used in **point-to-point** mode. If the `bindAddr` option is set, the node listens for incoming HTTP connections on the specified address. When a request is received, the destination host (from the Host: header) is examined. The node looks at all other nodes that are members of the tunnel, and selects the first one which contains an entry in the hosts list that matches the destination host. A tunnel connection is formed between these two nodes, and the destination node attempts to open an HTTP connection to the destination host. If the connection is established, data flows bi-directionally over the tunnel.

### Parameters

| | |
|---|---|
| `bindAddr` | An IP address (and optional port) on which to listen for HTTP connections. |
| `timeout` | A length of time in milliseconds to be used as the timeout when forming an outgoing HTTP connection. |
| `hosts` | A list of HTTP hosts for which this node can proxy requests. May contain wildcards ("*").<br><br>Displays host names or if none the link displays as "0 items". Click on host names for the dialog box to Add or Delete.<br><br> Hosts<br>Policies<br>0 items<br>Hosts<br>HTTP*<br>Click **Plus** to add a Host list. |

## UDP

In **Broadcast mode**, both the `bindAddr` and `destAddr` settings must be supplied. Each Distrix node which is a member of the tunnel listens on `bindAddr` and sends data received from other nodes to `destAddr`.

In **Point to point mode**, none of the options are strictly required. Any node with the `bindAddr` option set listens on that address for new UDP packets. A UDP packet from a unique source results in a tunnel connection being created.

Options and the resulting actions are described as follows:

| If... | Then... |
|---|---|
| the `remoteDest` option is set on that node | it is associated with the tunnel. |
| the connection is setup | the first available node, which is also a member of the tunnel, is used for the other end of the connection. |
| this node has the `destAddr` option set | UDP packets are forwarded to this address.<br><br>NOTE: This over-rides a `remoteDest` setting. |
| it is not, and the `remoteDest` option was set at the other end of the tunnel | `remoteDest` is used as the destination address. |
| neither are set | the tunnel connection is not established. |
| the connection is established | both sides of the connection sends data bi-directionally. |

| If... | Then... |
|---|---|
| either does not receive any data in either direction within the timeout period | the connection is closed. |

**Parameters**

| | |
|---|---|
| `bindAddr` | An IP address (and optional port) on which to listen for UDP data. |
| `destAddr` | The host and port to which traffic emerging from this tunnel at thi[illegible]nd is sent (overrides 'Remote Destination' from the other side). |
| `remoteDest` | The host and port to which traffic emerging from this tunnel on the [illegible]s sent (overridden by 'Destination Address' on the other side). |
| `timeout` | A length of time in milliseconds in which to shut[illegible]n the virtual connection [illegible] data is received. Only applies to point to point tunn[illegible] |

## TCP

The TCP tunnel can only be used in point-to-point mode.

| If... | [illegible]en... |
|---|---|
| the `bindAddr` option is set | [illegible]e [illegible]e listens [illegible]ncoming TCP connections on the [illegible]ecific [illegible]ddress. |
| the `remoteDest` option is set on tha[illegible] | it is [illegible]ssocia[illegible] with the connection. |
| a TCP connection is received on this nod[illegible] | [illegible]s the first other node that's a member of the tunne[illegible]nd sets up a tunnel connection to it. |

This node forms a connectio[illegible]o `dest[illegible]r` (i[illegible]s set on the node), or `remoteDest` (if it was set on the first node), or reject the connectio[illegible]n[illegible] the TCP connection is established, data flows bi-directionally between the two no[illegible], u[illegible]l one of the [illegible] connections is closed; at which point the tunnel is closed.

**Param[illegible]ers**

| | |
|---|---|
| `bin[illegible]dr` | [illegible]P ad[illegible]ss (a[illegible]d optional port) on which to listen for incoming connections. |
| `destAdd[illegible]` | The host [illegible]d port to which traffic emerging from this tunnel at this end is sent (overrides 'Remot[illegible]estination' from the other side). |
| `remoteDest` | [illegible]e host and port to which traffic emerging from this tunnel on the far end is sent (over-ridden by 'Destination Address' on the other side). |
| `timeout` | A length of time in milliseconds to be used as the timeout when forming an outgoing TCP connection. |

### Ethernet

(Currently only available for Linux.) In an Ethernet tunnel entire frames are encapsulated and passed through the Distrix network; it essentially acts as an extended Ethernet cable with routing capabilities. For each network tunnelled, a Tap device and a Linux software bridge must be configured. The latter bridges between the TAP device and the interface on the network being tunnelled. In order to preserve Ethernet packet headers, Distrix connects to the TAP device, which, with the bridge in place, allows effective read/write from the physical Ethernet interface.

> **NOTE**: There are some Ethernet tunnel dependencies which (on Ubuntu Linux) require the installation of utilities packages (`bridge-utils` and `uml-utilities`) to enable Ethernet network bridging. Both packages are installed using apt from the command line as follows: `sudo apt-get install bridge-utils uml-utilities`.

(See "Automate TAP & Bridge Utilities" on page 27 to configure automation of the interface set-up.)

The Ethernet tunnel can only be used in **Point to point mode**. If a tunnel instance is acting as a switch, it will accept connections from other Ethernet tunnel instances (switch and non-switch) and route the traffic it receives from its interface to the appropriate other tunnel(s) using a MAC table.

At least one part of the tunnel needs to be configured as a "switch". In Distrix 4.2 set "switch" to "true" in the tunnel.config file. (Previous versions: Distrix 4.1 has "switch" default to "true".) Multiple switches can be connected together and multiple non-switches can connect to a single switch.

> **NOTE**: When running in switch mode, traffic received over Distrix from another tunnel is only sent out over the tunnelled Ethernet interface and will NOT be sent to any other tunnel endpoint.

TCP traffic carried over an Ethernet tunnel is marked as being reordered. This means that if the reorder timeout configuration for the tunnel instance is non-zero, TCP packets that arrive at the other end of the tunnel out of order will be queued for the reorder timeout (or until the missing packets are received over the tunnel).

This can help TCP performance in situations where packets are frequently reordered, for example, if you are sending data over multiple links with rapidly varying latencies (such as wireless links). However, in a wired or single link scenario, reordering typically causes TCP traffic to be slower.

#### Parameters

| | |
|---|---|
| `device` | The ethernet interface to tunnel data to/from. |
| `switch` | Whether or not this end-point should act as a "switch" for multiple incoming tunnels. |

#### Configuration

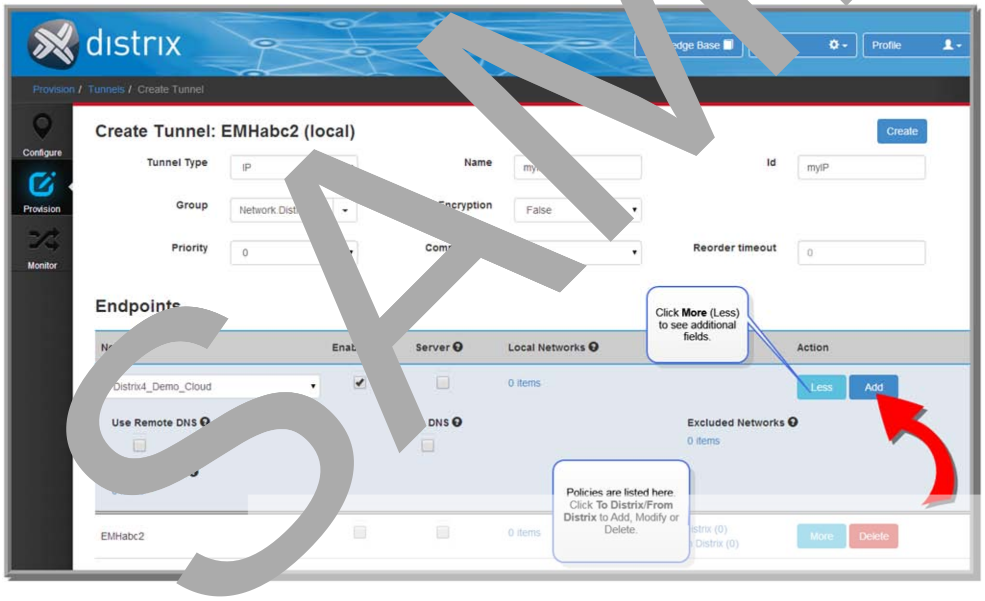
```
{
   "device": "en0",
   "switch": false
}
```

## IP

The IP tunnel connector allows Distrix 4.2 to tunnel IP traffic between different IP networks (IPv4 and IPv6). Acting at the IP rather than Ethernet layer, it is similar to a traditional virtual private network (VPN). Once the IP tunnel is setup, IP traffic for the tunnelled remote networks must arrive at the machine hosting the Distrix node in order for Distrix to tunnel the traffic. Distrix automatically configures the machine on which it's running so the routes are set up correctly. Any other local network machines must be configured to use the Distrix node as the gateway for the remote networks. To achieve this use one of the following options:

- Run the Distrix node on the regular network gateway.
- Configure local machines to specify the Distrix node as the gateway for tunnelled remote networks.
- Configure the regular network gateway to route traffic destined for the tunnelled remote networks to the local Distrix node.
- Setup a Network Address Translation (NAT) methodology on the local Distrix node between it and the tunnelled remote network. This method allows machines on the remote network to access machines on the local network but not vice versa.

**NOTE**: If using Windows, then DIstrix must be run as an administrator to use the IP tunnel connector.


distrix
Knowledge Base
Profile
Provision / Tunnels / Create Tunnel
Configure
Provision
Monitor
Create
Click More (Less) to see additional fields.
Less
Add
Policies are listed here. Click To Distrix/From Distrix to Add, Modify or Delete.
More
Delete

Create Tunnel: EMHabc2 (local)

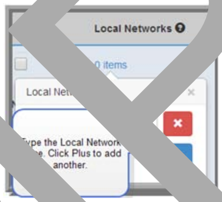
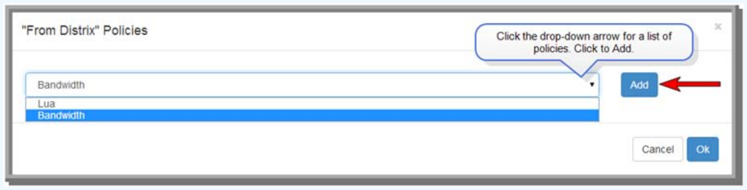
| | | | | | |
|---|---|---|---|---|---|
| Tunnel Type | IP | Name | my | Id | myIP |
| Group | Network Dist | Encryption | False | | |
| Priority | 0 | Comp | | Reorder timeout | 0 |

Endpoints

| Name | Enabled | Server | Local Networks | | Action |
|---|---|---|---|---|---|
| Distrix4_Demo_Cloud | ✓ | | 0 items | | Less Add |
| Use Remote DNS | | DNS | | Excluded Networks 0 items | |
| EMHabc2 | | | 0 items | Distrix (0) Distrix (0) | More Delete |

Selecting the tunnel type automatically displays the endpoint attribute fields for that particular type. (Click **More** to see additional fields.*) The following list of tunnel attributes have been mapped to the config file (API) for your information only and are as follows:

### Parameters

| Configuration File | UI | Description |
|---|---|---|
| `server` | **Server**<br><br>Select and check the box if `true`. | Determine whether this endpoint accepts incoming tunnel connections. |
| `sharedNetworks` | **Local Networks** | A list of the local IP networks th[illegible]hould be tu[illegible]lled to the remote side of the tunnel. This ma[illegible]clude [illegible]al IPv4 and [illegible]v6 networks.<br><br>Networks in this list are available to the co[illegible]uter r[illegible]ing the remote tunnel end[illegible]nt (and other computers [illegible] remote network, if the remote t[illegible]l endpoint is acting as a gateway).<br><br>Displays t[illegible]es or if [illegible], the link d[illegible]ays as "0 items". Click [illegible]names [illegible] dialog [illegible]to Ad[illegible]r Delete.<br><br> Local Networks<br>0 items<br>Local Net<br>Type the Local Network [illegible]e. Click Plus to add another. |
| `tunnelPolicy_toDistrix`<br><br>`tunnelPolicy_fromDistr[illegible]` | [illegible]licies | [illegible]se po[illegible]s for granular control and analysis of data traffic t[illegible]ough tu[illegible]els. (See API Guide Tunnel Policies.) Applied p[illegible]ies are listed here and if there are none then it's indicated by [illegible] click the text link to open the dialog box to select and apply policies.<br><br>■ **To Distrix**–The policies that apply to data entering the tunnel from an outside source via the instance.<br><br>■ **From Distrix**–The policies that apply to data leaving the tunnel. Both can be a JSON object specifying a "type" attribute and any policy-specific configuration or an array of strings designating policy types.<br><br>Click the arrow in the drop down box to select the policies to add.<br><br> "From Distrix" Policies<br>Click the drop-down arrow for a list of policies. Click to Add.<br>Bandwidth<br>Lua<br>Bandwidth<br>Add<br>Cancel<br>Ok |
| `useDNS` | **Use Remote DNS*** | Determine whether this side of the tunnel uses the remote DNS server (if available). |

### Configuration Example

```
{

  "server": true,
  "useDNS": false,
  "pushDNS" false,
  "sharedNetworks": [],
  "excludedNetworks": [],
  "remap": [

       {

            name: "IpTunnel",
            orig: "192.168.0.0/24"
            mapped: "10.0.1.0/24"

       }

  ]

}
```

## Serial RS232

The RS232 Serial tunnel can only be us[illegible] in [illegible]**int to po**[illegible]**ode**. If the bindAddr option is set, the node listens for incoming RS232 connections [illegible] the [illegible]cified ad[illegible]ss. When a request is received, the destination host (from the Host: head[illegible]) is examined. [illegible]e node[illegible]oks at all other nodes that are members of the tunnel, and selects the first one [illegible]tains an e[illegible]ry in the[illegible]osts list that matches the destination host. A tunnel connection is formed betwe[illegible]hese [illegible]ode[illegible] and the destination node attempts to open a Serial connection to the destination host. [illegible]he connect[illegible]s established, data flows bidirectionally over the tunnel.

| Configuration File | UI | Description |
|---|---|---|
| | | greater than 9 require the \\.\ prefix, while those with lower numbers may have it omitted, as in COM2.<br><br>This is a required value. |
| `baudrate` | **Baud Rate** | An integer which determi[illegible] the spe[illegible] of the serial connection and is dependent on [illegible] required by [illegible] application or device connected [illegible] he serial p[illegible] The default is `115200` Bd.<br><br>**NOTE**: The [illegible] erating system requires that the "[illegible] te" be [illegible] of a set of [illegible] rete values and the va[illegible] s suppo[illegible] by a [illegible] en device are usually a subse[illegible] f these va[illegible]<br><br>[illegible] r ref[illegible] ence on supported values go to the following webs[illegible]<br><br>Wind[illegible]<br><br>[illegible] inux |
| `databits` | **[illegible] ta Bits** | The number of data bits per serial frame. This is determined by the properties of the device being connected, though in a majority of cases the default value of `8` bits is correct. Permitted values are in the range 5 to 9. |
| `stopbits` | **Stop [illegible] its*** | The number of stop bits per serial frame (0, 1, or 2). This parameter is also determined by the requirements of the device being connected. The default value is `1` bit. |
| `parity` | **Parity***<br><br>Click the drop-down arrow to select from a list. | The parity setting is determined by the properties of the attached device and may be one of three values: "`even`", "`odd`", or "`none`." The default is "`none`." |
| `ha[illegible] areFlowControl` | **Hw flow control***<br><br>Check the box if `true`. | This boolean value determines whether or not the serial port's RTS (ready to send) and CTS (clear to send) lines are used to manage the data flow.<br><br>When turned on, the serial port hardware at both ends of the cable, start and stop the transfer as needed to prevent the overflow of hardware serial data buffers. |

# Application Tunnel Connector

Distrix's application tunnel connector enables building client applications that connect directly to a local Distrix gateway without needing to manage a socket connection.

Internally, App tunnel clients use a TCP socket to initially discover the local Distrix gateway and th[illegible]use a faster inter-process communication (IPC) mechanism to transfer data. The TCP port that the Di[illegible]x gateway listens on can be controlled by setting the “port” value in the AppTunnel.cfg configuration file [illegible]ated at the f[illegible]lowing:

**Linux**

```
/opt/distrix/etc/AppTunnel.cfg
```

**Windows**

```
C:\Program Files\Distrix 4\etc\AppTunnel.cfg
```

> **NOTE**: App tunnel clients can only connect to a Distrix g[illegible]way on the [illegible]me machi[illegible] security reasons, app tunnel clients must be running as the sa[illegible]ser as the Di[illegible]x gateway (o[illegible] as root/administrator) in order to connect.

## Guidelines

| | | |
|---|---|---|
| **Installation** | **If...** | **Then...** |
| | [illegible]g App tunn[illegible] client c[illegible]ections | the Distrix gateway must be running the App.tunnel plugin which is included with the core Distrix 4.1 installer. |
| | c[illegible]ng new App tunnel clients | the Distrix developer package must be installed to provides the necessary header files and libraries to build App tunnel client applications. |
| **Building a Client** | To use the App tunnel functionality the client application must include<br>▪ the App tunnel header file, and<br>▪ link in the client library.<br>Both the header file and the client library are included in the Distrix developer’s (dev) package.<br>They are installed at the following locations: | |

| | |
|---|---|
| | **Header File**<br>**Linux**<br>`/opt/distrix/include/dx_apptunnel.h`<br>**Windows**<br>`C:\Program Files\Distrix 4\inclu[illegible]dx_apptunnel.h`<br>**Client Library**<br>**Linux**<br>`/opt/distrix/lib/libdx_apptunnel.[illegible]`<br>**Windows**<br>`C:\Program Files\Di[illegible]ix 4\lib\dx_apptunnel.lib` |
| **System Libraries** | Some ad[illegible]ional system[illegible]braries are r[illegible]red to build an App tunnel:<br>**[illegible]**<br>li[illegible]hr[illegible] librt, an[illegible]m (-lpthread -lrt -lm).<br>**Wi[illegible]ows**<br>winm[illegible]lib, ws2_[illegible]2.lib, mswsock.lib, advapi32.lib, and iphlpapi.lib |

**Linux**–use th[illegible]Makefile[illegible]ild App [illegible]unnel Connector examples.

**Makefile**

```
[illegible]FIX?= /opt/[illegible]strix
INCDIR?= $(PREF[illegible])/include
LIBDI[illegible]REFI[illegible]/lib
C[illegible]gcc CF[illegible]GS+= -Wall -Wextra -Wno-unused-parameter -Werror -fPIC -I$(INCDIR)
[illegible]DFLAGS+= -[illegible](LIBDIR)
LDADD+= -l[illegible]_apptunnel -lpthread -lrt -lm

SR[illegible]brecv.c bsend.c
OBJS= brecv.o bsend.o

all: brecv bsend

brecv: brecv.o
```

```
      $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $+ $(LDADD)

bsend: bsend.o

      $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $+ $(LDADD)

%.o: %.c

      $(CC) $(CPPFLAGS) $(CFLAGS) -o $@ -c $<

clean:
rm -f brecv bsend $(OBJS)
.PHONY: all clean
```

### Request Example

```
#include "dx_apptunnel.h"
#include "Distrix/Abstractions/Time/Time.h"

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

static void signalHandler(int p_sig)
{

      exit(0);

}

int main(int argc, char **argv)
{

      signal(SIGINT, &signalHandler);
      signal(SIGTERM, &signalHandler);

      unsigned short l_port = 0; if (argc > 1)


            l_port = atoi(argv[1]);

      }

      DX_con[illegible]l_port);
      DX_TunnelConfig l_config;
      memset(&l_config, 0, sizeof(l_config));
      l_config.name = "apptest_broadcast";
      l_config.id = "broadcast";
      l_config.metadata = "hello";
```

```
     DX_Stream l_stream = DX_broadcast_open(&l_config, 0);
     if(!l_stream)
     {

          printf("Failed to create broadcast stream\n");
          return -1;

     }

      char *l_data = "broadcast test";
     for(;;)
     {

          int l_sent = DX_send(l_stream, l_data, strlen(l_data),
          0); printf("sent %d\n", l_sent);
          if(l_sent < 0)
          {

               return 1;

          }
          DX_msleep(50);

     }

     return 0;

}
```

### Response Example

```
#include "[illegible]_ap[illegible]el.h"
#include "Distrix/Abstractions/Time/Time.h"

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int main(int argc, char **argv)
{

     unsigned short l_port = 0;
     if (argc > 1)
     {

          l_port = atoi(argv[1]);

     }
```

```
    DX_connect(l_port);

    DX_TunnelConfig l_config;
    memset(&l_config, 0, sizeof(l_config));
    l_config.name = "apptest_broadcast_recv";
    l_config.id = "broadcast";
    l_config.metadata = "hello";

     DX_Stream l_stream = DX_broadcast_open(&l_config, 0);
    if(!l_stream)
    {
        printf("Failed to create broadcast stream\n");
        return -1;
    }
    char l_buffer[256];
    unsigned l_recvCount = 0;
    uint64_t l_startTime = DX_getTimeMs([illegible]
    while(l_recvCount < 10 && (DX_getTimeM[illegible] - l[illegible]tartTime) < 20000)
    {
        int l_recv = DX_receive(l_s[illegible]re[illegible] l_bu[illegible], sizeof(l_buf-
        fer)-1, 2000);
        printf("received %d\n", l_rec[illegible];
        if(l_recv >= 0)
        {
            l_buffer[l_recv] [illegible] '\0';
            pri[illegible]" > %s\n", l[illegible]ffer);
            ++l[illegible]ecv[illegible];
        }
        else if(l_r[illegible] = -2)
        {
            [illegible]f("s[illegible]ream deleted\n");
            retu[illegible] -1;
        }
    }
    return (l_recvCount >= 10 ? 0 : 1);
}
```

Network Topology

"Monitor a Node List" on page 97

Node Status Detail

Tunnel Connections

Tunnel Instance Status

Node Connection List

Link Connection List

Link Status

# Monitor

# Monitor Network Topology

The administrator, as part of the network configuration, establishes the "norms" or conditions that can be expected for the type of network set-up. Once this baseline is formed, then other users can monitor the nodes, connections, tunnels and links for any changes. The configuration files ("Configuration Files Location" on page 135) and those found in the Control API Guide, provide descriptions of the parameters, attributes and values. There are several functions that can be monitored for each network:

- Node status.
- Point-to-point connections between nodes.
- End-to-end tunnel stream status and capacity utilization.

This section enables a graphical view of your Distrix networks, nodes, tunnels and links and for each, the ability to drill down to monitor and configure. The details are displayed as follows:

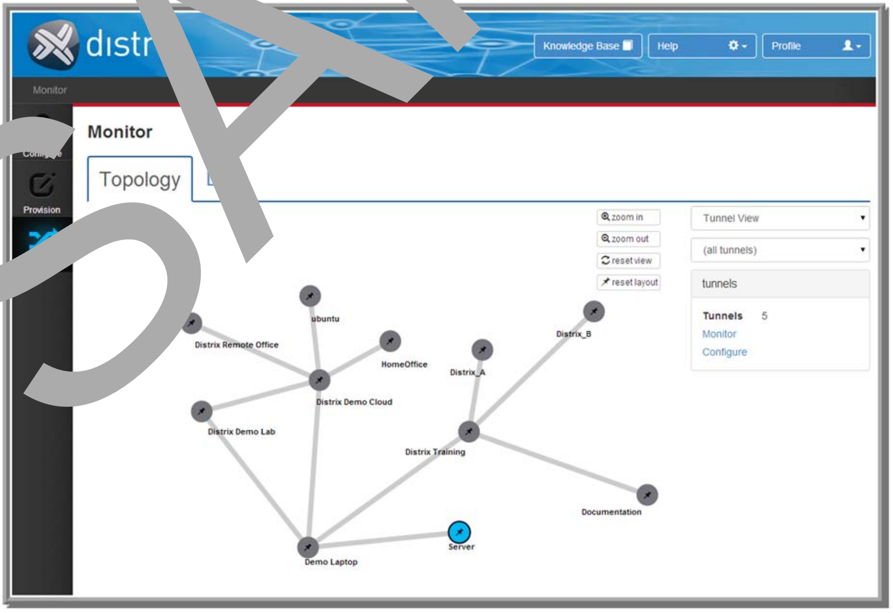
- **Networks**–all Distrix nodes and their connections.
- **Tunnels**–the selected node and the number and name of tunnels along with nodes linked by tunnel endpoints to this node.

Click the drop down arrow to select one of two view modes as follows:

**Network View**

| Item | Action | Description |
|---|---|---|
| **Display** | **Default** | The Network View is the default view that displays when the page loads. The system automatically determines the position of all nodes (each with its associated label) and displays the network by default in the cleanest layout as possible. |
| | **Zoom In/Out** | Use the mouse scroll wheel or click +/- to the right of the displayed network to zoom. Click and drag in the white-space to pan. Click the circulating arrows beneath the zoom buttons to reset the pan and zoom. |
| **Nodes** | **Selection** | By default the "local" node is selected and is highlighted in blue. Click on any node to select it; the selection is indicated by changing the node from gray to blue.<br><br>When a node is selected the name, network and version are displayed. Use the hyperlinks to drill down to monitor the node, associated tunnels or connections.<br><br>Click the drop down arrow to select **Tunnel View**.<br><br>Click in the white-space to deselect ALL nodes. If ALL nodes are deselected the Tunnel View is not access- |

| Item | Action | Description | |
|---|---|---|---|
| | | there is another node in the net-work which has an endpoint for that tunnel, | a dark blue line is drawn between them. This may be on top of a connection line, or it may be indepen ent if there aren't dir-ect connections h ween nodes. |
| | | there is a single tunnel shared between the nodes, | the line is n. |
| | | there is more tha one tunnel, | the line is thicker. |
| **Tunnels** | **Selection** | Cl on the line o specific tu to display the following:<br>▪ An on to filter the view between<br>  ▪ all tunnels, or<br>  ▪ all tunnels of a specific connector type.<br>▪ The work name, connection, number of links and link ID. Use the hyperlinks to drill down to monitor and configure. | |



| Filename | Filepath | Description | Installed w/Package | Uninstalled w/Package |
|---|---|---|---|---|
| | | vider. | | |
| **App.tunnel** | /opt/distrix/lib/App.tunnel | Distrix App tunnel support provider. Used when internodal App calls are made. | Yes | Yes |

### distrix-httpapiserver

| Filename | Filepath | Description | Installed w/Package | Uninstalled w/Package |
|---|---|---|---|---|
| **Distrix REST API Server** | /opt/distrix/bin/dxhttp | Must run so that the REST API (and web interface) is available. | Yes | Yes |
| **Distrix htpasswd Authorization Plugin** | /opt/distrix/bin/htpasswdauth | Use htpasswd to protect the Distrix node web interface. | Yes | Yes |
| **Distrix ldapauth Authorization Plugin** | /opt/distrix/bin/ldapauth | Use ldapauth to protect the Distrix node web interface. | Yes | Yes |
| **Default htpasswd** | /opt/distrix/etc/defaultpasswds.htpassd | Used by Distrix' htpasswd plugin to protect the Distrix node web interface. Replaced by administrator. | Yes | Must use "--purge" |
| **Configuration File** | /opt/distrix/etc/dxhttp.cfg | Dictates REST API server and web interface behavior. Specify the use of an authentication plugin here. | Yes | Must use "--purge" |

Hardening Ubuntu Linux

Configuration Files Location

HTTP Server Configuration

Communication Configuration

Security Configuration

Link Modules Configuration

Link Modules List

Tunnels Configuration

Log Levels Log Outputs

Logging Configuration

# Appendix B

# Configuration Files Location

The configuration files can be modified to adjust node behaviour and to configure and connect to other Distrix 4.2 nodes.

All configuration files are:

| | |
|---|---|
| In JSON format. | The contents of each file is a JSON object containing one or more name value pairs. |
| Stored by default in a directory called "etc" | Linux: Found in filepath: /opt/distrix/etc |
| "etc" is located relative to the Distrix installation directory. | Linux: the -d option may be passed to the Distrix process to change this directory. |

**NOTE**: The Distrix process must be restarted before any changes made to the configuration files take effect.

### Systems Location

| Linux | Windows |
|---|---|
| `/opt/distrix/etc/` | `\ProgramData\Distrix 4\etc (where \ProgramData is typically C:\ProgramData)` |

### Configuration Files

| | |
|---|---|
| "Communication Configuration" on page 137 | Configuration of communication between Distrix nodes; both how the configured node connects to others and how other nodes can connect to it. |
| "Logging Configuration" on page [illegible] | Logging configuration for the Distrix node. |
| "Security Configuration" on page [illegible] | Encryption and grouping configuration for securing communication between Distrix nodes. |
| "Tunnels Configuration" on page 142 | Configuration of tunnels from the specified node. |

## HTTP Server Configuration

The Distrix HTTP API server is an application which contains an HTTP server and communicates with the local Distrix process to provide the HTTP API. The server reads its configuration from a file named dxhttp.cfg, which is in the etc directory, at ../etc relative to the bin directory. The configuration file contains a JSON object. The following parameters may be specified:

### Parameters

| | |
|---|---|
| `apiAddress` | A string containing the IP address and port that the Distrix process liste[illegible]g on for API commands. Default: a local socket or (on Windows) a local pi[illegible] |
| `webListenAddress` | A string containing an IP address and port on which to listen for HTTP con- nections. Default: localhost:4000 |
| `certPath` | A path to an SSL certificate. If specified, th[illegible]erver wil[illegible] HTTPS inste[illegible] of HTTP. |
| `keyPath` | A path to the corresponding key fi[illegible]or the SSL ce[illegible]ficate. Requ[illegible]d if certPath is specified. Default: <empty> |
| `staticPath` | A local filesystem path fr[illegible]hich to s[illegible]tatic files (i.e. HTML, Javascript, images, etc..). Default: <e[illegible] |
| `staticUrl` | The base URL from which to [illegible]erve [illegible]tic files. [illegible]ault: /static |
| `cors` | A child obje[illegible]ining the fo[illegible]wing pa[illegible]eters:<br>■ `ena`[illegible]—If t[illegible]COR[illegible] support is enabled, allowing Javascript reques[illegible]o the A[illegible] domains other than that of the API itself. Default: [illegible]e<br>`llowedHos`[illegible]A list of domain names from which requests are accep- [illegible]f not speci[illegible]d, all domains are accepted. Default: <empty> |
| `tokenFile` | The [illegible]esys[illegible] path o[illegible]e file to use for authentication token storage. Default: ../etc[illegible]ns.cfg |
| `userA`[illegible]`nPlugin` | The ap[illegible]cation to use for user authentication. If none is specified, user authen- [illegible]tion [illegible]isabled. See authentication. Default: <empty> |
| `userA`[illegible] | An [illegible]ptional list of arguments to pass to the user authentication plugin. Default: <[illegible]pty> |

### Configuration

```
{

    "apiAddress": "[::1]:2259",
    "webListenAddress": "localhost:4000",
    "staticUrl": "/static",
    "staticPath": "./static/",
    "cors": {

          "enabled": true,
          "allowedHosts": null },

    "keyPath": "", "certPath": "",
    "userAuthPlugin": "bin/htpasswdauth",
    "userAuthArgs": ["conf/htpasswd.conf"],
    "tokenFile": "conf/tokens.conf"

}
```

## Communication Configuration

The file Communication.cfg is used for configuring Distrix communication links between Distrix nodes. There are a number of top-level configuration parameters in this object and a "modules" object which can contain an entry for each available link module.

```
{
  "name": "Distrix-abc.local",
  "network": "Distrix",
  "maxQueueSize": 10485760,
  "modules": {
       "UDP": {
            "accept": true,
            "acceptPatterns": ["*"],
            "failoverOrder": 0,
            "spillOrder": 0,
            "heartbeatInterval": 500, "maximumSegmentSize": 1472,
            "useChecksum": true,
            "bandwidthEstimate": 0,
            "listenPort": 24444, "minConnectRetryInterva[illegible]: 50,
            "maxConnectRetryInterval": 2000, "r[illegible]eThrea[illegible]": 1,
            "requireHelloCookie": true,
            "enabled": true,
            "targets": [{
                        "target":"192.[illegible]8.40.1[illegible]et-
                        work": null,
                        "config": null
                  }],
            "linkSpecific"[illegible]
                  "127.0.0.4": [illegible]
                        "failoverOr[illegible]": 2
                  }
            }
       }
       [illegible]TCP": {
            "[illegible]: t[illegible],
            [illegible]accept[illegible]terns": ["*"],
            "failov[illegible]Order": 0,
            "spill[illegible]der": 0,
            "h[illegible]tbeatInterval": 500, "maximumSegmentSize": 2920,
            [illegible]useChecksum": false,
            "bandwidthEstimate": 0,
            "listenPort": 25444,
            "connectTimeout": 2000,
            "handshakeTimeout": 5000,
            "enabled": true,
            "targets": [{
```

```
                      "target": "192.168.40.135", "net-
                      work": null,
                      "config": {
                      }

                 }],

            "linkSpecific": {
            }

       }

  }

}
```

## Security Configuration

The file Security.cfg is used to configure the certificates Distrix uses for authentication, authorization and encryption. It contains one list of "caCerts" and one list of "signerCerts".

```
{
  "security": "full",
  "apiGroupSuffix": "",
  "caCerts": [{
            "type": "ecc",
            "name": "ca.ecc.pub",
            "trust": ["*"],
            "file": "ca.ecc.pub"
        }, {
            "type": "openssl",
            "name": "CAs.pem",
            "trust": ["*"],
            "file": "CAs.pem"
        }],
  "signerCerts": [{
            "type": "ecc",
            "name": "cert.ecc",
            "purpose": ["Network.Distrix"], "file": "cert.ecc"
        }, {
            "type": "ecc",
            "name": "tunnels.ecc",
            "purpose": ["Tunnels.Distrix"], "file": "tunnels.ecc"
        }, {
            "type": "ecc",
            "name": "monitoring.ecc",
            "purpose": ["Monitoring"],
            "file": "monitoring.ecc"
        }, {
            "type": "ecc",
            "name": "configuration.ecc", "purpose": ["Con-
            figuration"],
            "file": "configuration.ecc"
        }, {
            "type": "ecc",
            "name": "security.ecc",
            "purpose": ["Security"],
            "file": "security.ecc"
        }, {
            "type": "ecc",
            "name": "provisioning.ecc", "purpose": ["Pro-
            visioning"],
            "file": "provisioning.ecc"
        }]
}
```

## Link Modules Configuration

Distrix 4.2 includes TCP and UDP Link Modules which can be specifically configured (as in the example), with braces, and preceded by the link module's name and a colon.

### Example

```
  "modules": {
      "UDP": {
                "enabled": true,
                ...
            },
      "TCP": {
                "enabled": true,
                ...
            },
      },
```

## Link Modules List

Use this portion of the file to specify configurations for Link Modules that require differentiation from those with default values. All configuration settings must be inside the top-level modules parameter.

**NOTE**: By default, any link module present in the /opt/distrix/LinkModule folder is loaded and run with their default values.

There are two ways a Link Module can be disabled or explicitly enabled:

- Specify false (or true) as the entirety of the link module configuration, avoiding the use of the enabled option altogether.
- Specify enabled: false (or true) within the body of the specific module's configuration.

### Disabled

```
"modules": {


               "UDP": false,
               "TCP": true,

}
```

### Enabled

```
   "modules": {
       "TCP": {

                "enabled": false,
                ...

             },

     }
```

## Tunnels Configuration

The file Tunnels.cfg is used for configuring Distrix tunnels. The JSON object in this file consists of a single name, "instances", whose value is a list of objects describing the tunnel instances. (See **Tunnel Instance Configuration** in the API Control Guide.)

> **NOTE**: When configuring a tunnel using the configuration files, Tunnels.cfg must be modified on all nodes which are to be tunnel endpoints in order to establish the tunnel. If using the Distrix user interface (UI) to configure tunnels, all potential endpoints can be selected and configured at once.

### Example

```
{
    "instances": [{
              "type": "UDP",
              "uuid": "1D4142A8-B6B856B8-3ED35CDC",
              "name": "UDPBroadcast",
              "id": "UDPBroadcast",
              "group": "Tunnels.Distrix",
              "linkType": "broadcast", "reorderTimeout[illegible]10,
              "priority": 0,
              "encrypted": true,
              "destAddr": "127.0.0.1:3000"[illegible]
              "bindAddr": "127.0.0.1:2000"
         }, {
              "type": "UDP",
              "uuid": "091C4214-[illegible]4E92F3-33[illegible]7FC8",
              "name": "Other UD[illegible]roadcast[illegible]
              "id": "myudp"[illegible]
              "group": "T[illegible]ls.Dis[illegible],
              "linkType": [illegible]dcast", [illegible]eorderTimeout": 10,
              "priority": 0,
              "encrypted": t[illegible]e,
              "d[illegible]dr": "12[illegible]0.0.1[illegible]001", "bindAddr":
              "12[illegible].[illegible]000"
         }],
              "type": "U[illegible]
              [illegible]d": "CEDB[illegible]B-C9F8E306-2A2BD9C7",
              "nam[illegible]UdpP2P",
              "p[illegible]rs": [[illegible]istrixabc.local","ubuntu"],
              [illegible]d": "UdpP2P",
              "group": "Tunnels.Distrix",
              [illegible]linkType": "p2p",
              [illegible]eorderTimeout": 100,
              "priority": 5,
              "encrypted": false,
              "bindAddr": "127.0.0.1:2001",
              "remoteDest": "127.0.0.1:3001", "timeout": 10000
         }]
}
```

## App Tunnel Connector

Distrix's application tunnel connector enables building client applications that connect directly to a local Distrix gateway without needing to manage a socket connection.

Internally, App tunnel clients use a TCP socket to initially discover the local Distrix gateway and then use a faster inter-process communication (IPC) mechanism to transfer data. The TCP port that the Distrix gateway listens on can be controlled by setting the "port" value in the AppTunnel.cfg configuration file located at the following:

**Linux**

```
/opt/distrix/etc/AppTunnel.cfg
```

**Windows**

```
C:\Program Files\Distrix 4\etc\AppTunnel.cfg
```

> **NOTE**: App tunnel clients can only connect to a Distrix gateway on the same machine. For security reasons, app tunnel clients must be running as the same user as the Distrix gateway (i.e. root/administrator) in order to connect.

### Guidelines

| | | |
|---|---|---|
| **Installation** | **If...** | **Then...** |
| | allowing App tunnel client connections | the Distrix gateway must be running the App tunnel plugin which is included with the core Distrix 4.1 installer. |
| | developing new App tunnel clients | the Distrix developer package must be installed to provides the necessary header files and libraries to build App tunnel client applications. |
| **Building Clients** | To use the App tunnel functionality the client application must include<br>▪ the App tunnel header file, and<br>▪ link in the client library.<br>Both the header file and the client library are included in the Distrix developer's (dev) package. They are installed at the following locations:<br>**Header File**<br>**Linux**<br>`/opt/distrix/include/dx_apptunnel.h`<br>**Windows** | |

```
        char l_buffer[256];
        unsigned l_recvCount = 0;
        uint64_t l_startTime = DX_getTimeMs();
        while(l_recvCount < 10 && (DX_getTimeMs() - l_startTime) <
        20000)
        {

                int l_recv = DX_receive(l_stream, l_buffer,
                sizeof(l_buffer)-1, 2000);
                printf("received %d\n", l_recv);
                if(l_recv >= 0)
                {

                        l_buffer[l_recv] = '\0';
                        printf(" > %s\n", l_buffer)[illegible]
                        ++l_recvCount;

                }
                else if(l_recv == -2)[illegible]
                {

                        printf("stream c[illegible]ted\[illegible];
                        return [illegible]

                }

        }
        return [illegible]vCount >= [illegible] ? 0 [illegible]);
}
```

## Log Leve[illegible]

Distrix [illegible]2 facilitates troubleshoo[illegible]g of its operations by providing a large amount of output which can take a variet[illegible]f forms an[illegible]urabl[illegible]hrough `Logging.cfg`, found in

```
/[illegible]istrix/Conf[illegible]/Logging.cfg.
```

As with the other config[illegible]ion files, attributes are denoted by their name, followed by a colon and then their setting (as outli[illegible]ow). If any attribute is missing, its default value is employed. If the entire file is missing, default values are assumed for all attributes.

There are seven logging levels, in order of increasing verbosity (each level logs all preceding levels). If not otherwise specified, the log level defaults to "info".

| Levels | Logs |
|---|---|
| `"off"` | Doesn't log at all, regardless of any other settings. |
| `"fatal"` | Only those errors that cause catastrophic failure. |
| `"error"` | Serious errors. |
| `"warning"` | Potential issues which haven't progressed to outright errors. |
| `"info"` | Information about the normal operation of Distrix. |
| `"debug"` | Detailed information about the normal operation of Distrix. |
| `"trace"` | Information about every operation performed by Distrix. |

### Syntax

```
level: loglevel,
```

### Example

```
level: "info",
```

## Log Outputs

Distrix 4.2 can log to any combination of the following outputs (or none, if all are disabled).

### Attributes

| Name | Description | Default Values |
|---|---|---|
| `printf` | If enabled, [[[Undefined variable VersionNumbers.Version1]]] outputs any logging to standard output though not if it is running as a service.; true or false. | true |
| `syslog` | If enabled [[[Undefined variable VersionNumbers.Version1]]] logs to a syslog server when a target (syslog server hostname or IP address) is specified. | false |
| `file` | If enabled, [[[Undefined variable VersionNumbers.Version1]]] logs to one or more logfiles. The following values must be specified:<br>■ `name`: The file path as a quoted string.<br>■ `maxSize`: The maximum size (in bytes) of any one log file.<br>■ `maxFiles`: The maximum number of rollover files; a numerical suffix is appended. (e.g. output.log.2) | disabled |

When the maximum number of files is reached, the oldest log file is overwritten. Log roll-over can be induced before `maxSize` is reached by sending `SIGHUP` to the Gateway process.

### Syntax

```
outputs: {
            option: value,
            ...
}
```

### Example

```
outputs: {
           printf: true,
           syslog: "localhost",
           file: {name: "/var/log/distrix/output[illegible]og", m[illegible]ize: 10000[illegible]
           maxFiles: 2}
}
```

## Logging Configuration

The file Logging.cfg controls the log output of the Dis[illegible] pr[illegible]ss, whic[illegible]ay be used for debugging or troubleshooting purposes. The top level entries contro[illegible]e ge[illegible]l output. [illegible] printf is true, the output is printed to stdout. If syslog is set to an IP addr[illegible]or hostname, [illegible]g outp[illegible] sent in syslog format over UDP. The api child object controls the log output rela[illegible] [illegible]calls, so[illegible]at it ca[illegible]e redirected to a separate file (for example) and used for auditing purposes.

The configuration file is fou[illegible]d at the following [illegible]tion:

### Linux

```
  /opt/distrix/etc/Log[illegible]ng.cf[illegible]
```

### Windows

```
  C:\[illegible]ogramData\Distrix 4[illegible]tc\Logging.cfg (a typical install)
```

## Example

```
   "level": "debug",
   "outputs" : {

        "printf": true,
        syslog: "localhost", // hostname of syslog server file: {name: "out-
        put.log", maxSize: 10000, maxFiles: 2} //path is relative to the gateway
        support directory (/opt/Distrix on linux)
        // maxSize is in bytes, maxFiles indicates rollover files - appends numer-
        ical suffix (eg. output.log.1) // Can cause log rollover by sending
        SIGHUP to the Gateway process

   },
   api: {

        level: "info",
        outputs: {

             printf: true, syslog: "localhost", // hostname of syslog
             server
             file: {name: "api.log", maxSize: 10000, maxFiles: 2} //path is
             relative to the gateway support directory (/opt/Distrix on
             linux)
             // maxSize is in bytes, maxFiles indicates rollover files -
             appends numerical suffix (eg. output.log.1)
             // Can cause log rollover by sending SIGHUP to the Gateway pro-
             cess

        }
}
```

# Glossary

# Glossary

## A

### Application Programming Interface (API)

Specifies how some software components should interact with each othe[illegible]n API comes[illegible]n the form of a library that includes specifications for routines, data structures, object c[illegible]ses, and var[illegible]es. In some other cases, notably for SOAP and REST services, an API comes as ju[illegible] specifica[illegible] of remote calls exposed to the API consumers. An API differs from an application binary i[illegible]face[illegible]BI) in that an [illegible] is source code based while an ABI is a binary interface. For instance POSIX is [illegible]PI, while the L[illegible]x Standard Base is an ABI.

### Asynchronous

Not occurring at the same time; operating without the use of [illegible] time intervals.

### Asynchronous Transfer Mode (ATM)

A standard defined in the 1980's and [illegible]igned to unify [illegible]ecommunicat[illegible]n and computer networks. It was designed for a network that must handl[illegible]h traditi[illegible] high-throughput data traffic (e.g., file transfers), and real-time, low-latency cont[illegible]t such as [illegible]ce a[illegible] video. ATM is a core protocol used over the SONET/SDH backbone of the [illegible]c switche[illegible]phone network (PSTN) and Integrated Services Digital Network (ISDN), but its use is d[illegible]g in favou[illegible]all IP. ATM provides functionality that is similar to both circuit switching and packet swi[illegible]ing [illegible]tworks: A[illegible] uses asynchronous time-division multiplexing, and encodes data into small, fixed-siz[illegible] pa[illegible]s (ISO-O[illegible]mes) called cells. This differs from approaches such as the Internet Protocol or Et[illegible]net t[illegible]se variable sized packets and frames. ATM uses a connection-oriented [illegible]el in which a v[illegible]ual circ[illegible] must be established between two endpoints before the actual data exch[illegible]. These [illegible]tual circ[illegible]ts may be "permanent", i.e. dedicated connections that are usually precon[illegible]red b[illegible] servi[illegible] provider, or "switched", i.e. set up on a per-call basis using signalling and disconne[illegible] when t[illegible] terminated.

## B

### Broadcast [illegible]nnel

In a bro[illegible]unnel, data sent over the Distrix network from one endpoint is delivered to all other endpoints in [illegible] tunnel that are configured to receive data. Note: Data is only delivered to the members of the tunnel an[illegible] not broadcast throughout the Distrix network. Broadcast tunnel endpoints may be configured [illegible]d dat[illegible]eceive data, or both. A broadcast tunnel endpoint that is configured to send data consumes [illegible]rably [illegible]ore network resources, even if no data is being sent over it.

### Buffer

A[illegible]uffer routine or storage medium used in telecommunications compensates for a difference in rate of [illegible]low of data, or time of occurrence of events, when transferring data from one device to another. A buffer is primarily used for input, output, and sometimes very temporary storage of data that is either en route between other media or data that may be modified in a non-sequential manner before it is written (or read) in a sequential manner. A "Disk Cache" or "File Cache" keeps statistics on the data contained within it and commits data within a time-out period in write-back modes. A buffer does none of this.

## L

### Latency

In a packet-switched network, latency is the delay between the sender and the receiver decodin[illegible]it; mainly a function of the signal's travel and processing time at nodes the information traverse[illegible] a packet-switched network it is measured from the source sending a packet to the destination recei[illegible] it, or from source to destination plus the one-way latency from the destination back to the source. [illegible]ency is compounded by traffic congestion, some network protocols and electromagnetic interfere[illegible].

### LDAP

The Lightweight Directory Access Protocol (LDAP) is an application protocol for accessing a[illegible]aintaining distributed directory information services over an Internet Protocol (IP) network. The LDA[illegible] us[illegible] to provide a "single sign-on" where one password for a user is share[illegible]tween many services, su[illegible] applying a company login code to web pages (so that staff log in only o[illegible]to company computers, and then are automatically logged into the company intranet). A client starts a[illegible]AP session by co[illegible]necting to an LDAP server, called a Directory System Agent (DSA), on [illegible]CP or U[illegible]enerally po[illegible]89).

### Least Required Access

Least permission or least required access are securi[illegible]als where[illegible]u only get to know what you need to know and only when you need to know it.

### Link (Distrix)

A single point-to-point transport path between t[illegible] Dist[illegible]odes. Cu[illegible]ntly we support UDP and TCP links. For example, a single TCP connection betw[illegible]en tw[illegible]istrix nodes is a link.

### Link Module

A plugin that provides the ability to cre[illegible] links over [illegible]ven protocol. Distrix has a TCP link module and a UDP link module

### Long Polling

The browser makes a[illegible]asyn[illegible]onous request of the server, which may wait for data to be available before res[illegible]g. The respo[illegible]an contain encoded data (typically XML or JSON) or Javascript to be e[illegible]cuted by the client. [illegible]e end of the processing of the response, the browser creates and sends [illegible]nother XHR, to await th[illegible]ext event. Thus the browser always keeps a request outstanding with the server, to be a[illegible]ered as[illegible]ch event occurs.

## M

### MAC table

Layer 2 switching uses the Media Access Control Address (MAC address) from the host's network interface cards (NICs) to decide where to forward frames. Layer 2 switching is hardware based which means switches use application-specific integrated circuit (ASICs) to build and maintain filter tables (also known as MAC address tables or CAM tables). One way to think of a layer 2 switch is as a multiport bridge. Layer 2 switching is highly efficient because there is no modification to the data packet, only to the frame encapsulation of the packet, and only when the data packet is passing through dissimilar media (such as from Ethernet to FDDI).