IceGrid Deployment

Previous Next

If you need to connect with multiple FIX acceptors, deploying IceFIX bridges using IceGrid can dramatically simplify the management of your installation. On this page we'll configure an IceGrid deployment that connects with two FIX acceptors named `TP1` and `TP2`. The IceGrid node and registry run on host `192.168.2.10`, and the data and configuration files reside in `C:\Fix`.

On this page:

Configuring the IceGrid services

First configure the IceGrid registry:

config.registry

```
IceGrid.InstanceName=FIXGrid
IceGrid.Registry.Client.Endpoints=tcp -h 192.168.2.10 -p 4061 -t 10000
IceGrid.Registry.Server.Endpoints=tcp -h 192.168.2.10 -t 10000
IceGrid.Registry.Internal.Endpoints=tcp  -h 192.168.2.10 -t 10000
IceGrid.Registry.Data=C:\Fix\db\registry
IceGrid.Registry.PermissionsVerifier=FIXGrid/NullPermissionsVerifier
IceGrid.Registry.AdminPermissionsVerifier=FIXGrid/NullPermissionsVerifier
IceGrid.Registry.SSLPermissionsVerifier=FIXGrid/NullSSLPermissionsVerifier
IceGrid.Registry.AdminSSLPermissionsVerifier=FIXGrid/NullSSLPermissionsVerifier
```

Next configure an IceGrid node named `Node`:

config.node

```
Ice.Default.Locator=FIXGrid/Locator:tcp -h 192.168.2.10 -p 4061 -t 10000
IceGrid.Node.Name=Node
IceGrid.Node.Endpoints=tcp -h 192.168.2.10 -t 10000
IceGrid.Node.Data=C:\Fix\db\node
```

Back to Top ^

Creating the IceGrid deployment

Now we'll create an IceGrid deployment and store it in the file `C:\Fix\fixgrid.xml`. We need to deploy one IceFIX bridge for each FIX acceptor, and we'll assume that QuickFIX configuration files for each of the bridges have already been created as `C:\Fix\config.tp1` and `C:\Fix\config.tp2`.

The following XML file defines an IceGrid template for an IceFIX bridge:

fixgrid.xml

```
<icegrid>
  <application name="IceFIX">
    <service-template id="IceFIX">
      <parameter name="instance-name" default="${application}.IceFIX"/>
      <parameter name="bridge-endpoints" default="default"/>
      <parameter name="fix-config"/>
      <parameter name="db-home" default=""/>

      <service name="${instance-name}" entry="IceFIXService,10:create">

        <dbenv name="${service}" home="${db-home}"/>

        <adapter name="${service}.Bridge"
                 id="${instance-name}.Bridge"
                 endpoints="${bridge-endpoints}">
          <object identity="${instance-name}/Bridge" type="::IceFIX::Bridge"/>
        </adapter>

        <properties>
          <property name="${service}.InstanceName" value="${instance-name}"/>
          <property name="${service}.FIXConfig" value="${fix-config}"/>
        </properties>
        <target name="debug">
          <properties>
            <property name="${service}.Trace.Bridge" value="1"/>
            <property name="${service}.Trace.Incoming" value="1"/>
            <property name="${service}.Trace.Outgoing" value="1"/>
            <property name="${service}.Trace.Event" value="1"/>
          </properties>
        </target>
      </service>
    </service-template>

    <server-template id="IceFIX">
      <parameter name="instance-name" default="${application}.IceFIX"/>
      <parameter name="bridge-endpoints" default="default"/>
      <parameter name="fix-config"/>
      <parameter name="db-home" default=""/>

      <icebox id="${instance-name}" exe="icebox" activation="on-demand">
        <service-instance template="IceFIX"
                          instance-name="${instance-name}"
                          bridge-endpoints="${bridge-endpoints}"
                          fix-config="${fix-config}"
                          db-home="${db-home}"/>
      </icebox>
    </server-template>
```

The service instance template takes four arguments:

- `instance-name` is a unique name assigned to each bridge
- `bridge-endpoints` specifies the endpoints for the bridge
- `fix-config` defines the path name of the QuickFIX configuration file
- `db-home` specifies the directory that contains the bridge's database files

The `fix-config` argument is mandatory, whereas the remaining arguments are optional (although typically you would always supply a value for `instance-name`).

Next, we need to deploy two instances of the bridge on the IceGrid node named `Node` using this template; one instance for `TP1`, and another instance for `TP2`:

fixgrid.xml (cont'd)

```
    <node name="Node">
      <server-instance
         template="IceFIX"
         instance-name="TP1"
         fix-config="C:\Fix\config.tp1"
         db-home="C:\Fix\db-tp1"/>
      <server-instance
         template="IceFIX"
         instance-name="TP2"
         fix-config="C:\Fix\config.tp2"
         db-home="C:\Fix\db-tp2"/>
    </node>
  </application>
</icegrid>
```

Note that the database files reside in `C:\Fix\db-tp1` and `C:\Fix\db-tp2`. As for the standalone deployment, these directories must be created in advance. If we had not supplied a value for `db-home` in the `server-instance` descriptors, the database files for each bridge would reside in the IceGrid deployment directory instead, and therefore it would not be necessary to create a database directory in advance.

To simplify the task of using `icegridadmin` and `icefixadmin`, let's create a configuration file named `config.admin` that contains a proxy for the IceGrid locator:

config.admin

```
Ice.Default.Locator=FIXGrid/Locator:tcp -h 192.168.2.10 -p 4061 -t 10000
```

We can add two more properties to `config.admin` to specify a user name and password for `icegridadmin` to use when logging into the IceGrid registry:

config.admin (cont'd)

```
IceGridAdmin.Username=foo
IceGridAdmin.Password=bar
```

Note however that the IceGrid registry does not validate this account information because we configured it to use the `NullPermissionsVerifier`.

Back to Top ^

Starting the IceGrid services

Before starting IceGrid for the first time, you must create several directories:

```
> mkdir C:\Fix\db
> mkdir C:\Fix\db\node
> mkdir C:\Fix\db\registry
> mkdir C:\Fix\db-tp1
> mkdir C:\Fix\db-tp2
```

Now we're ready to start the registry:

```
> icegridregistry --Ice.Config=C:\Fix\config.registry
```

In another window, start the IceGrid node:

```
> icegridnode --Ice.Config=C:\Fix\config.node
```

Back to Top ^

Deploying the IceFIX bridges

The following command deploys the IceFIX bridges as defined in `fixgrid.xml`:

```
> icegridadmin --Ice.Config=config.admin -e "application add C:\FIX\fixgrid.xml"
```

This command only needs to be executed once. If you subsequently change `fixgrid.xml`, you can deploy the changes using the `application update` command.

Back to Top ^

Activating the bridges

After deployment, we can activate the bridges using `icefixadmin`:

```
> icefixadmin --Ice.Config=C:\Fix\config.admin
Ice 3.4.2  Copyright 2003-2011 ZeroC, Inc.
>>> status
TP1
  status: not active
TP2
  status: not active
>>> activate
activating TP1
activating TP2
>>> status
TP1
  status: logged on
TP2
  status: logged on
>>>
```

If you'd like to configure the bridges to log diagnostic messages, use the `debug` target when deploying the application:

```
> icegridadmin --Ice.Config=config.admin -e "application add C:\FIX\fixgrid.xml debug"
```

Or, if you have already deployed the application, you can update it as follows:

```
> icegridadmin --Ice.Config=config.admin -e "application update C:\FIX\fixgrid.xml debug"
```

Back to Top ^