Monitoring your Environment

Scenario: "I want to monitor my environment and check that all components (broker, RPC servers) are up and running."

EntireX offers a script-based solution to check if all brokers and services of a defined environment are active.

- Defining your Environment
- Monitoring your Environment
- Error Handling

Defining your Environment

To define the environment to be monitored

- Select option 7 from the *EntireX Command-line Script Menu*, "Define your Environment".

  Or:
  Enter command `edit_user_specific_environment_definition.bat` to specify the environment to be monitored (defined by broker and list of services).

  This opens a text editor (for example Notepad) with a sample definition of an environment that you can customize. You can enter values for the following parameters:

Parameter	Value	Description	Note
ENVIRONMENT	<Env_Name>	Logical name of the environment	
ERROREXIT	<Exit_Name>	Batch file to be called if a component of the environment is not active.	See *Error Handling*.
BROKER	<Broker_Name> <Broker_ID>	Logical name and ID of broker used for the etbinfo calls.	
	<Broker_Name> <Broker_ID> <UserId> <Password>	Additional user ID and password if the broker is running with EntireX Security.	
SERVICE	<Service_Name> <Class> <Server> <Service>	Logical service name, class, server, service to be monitored.	Checks if the specified service is registered at the broker.
RPCSERVICE	<RPC_Service_Name> <Class> <Server> <Service>	Logical RPC service name, class, server, service to be monitored.	Valid only for RPC servers and issues an RPC ping command to the specified service.

Notes:

1. The file may contain a list of environments.
2. Each environment can consist of list of brokers, and for each broker a list of services can be defined.
3. Blanks in the logical names are not supported.

The file you define here is used for the following scripts:

monitor_environment.bat	See *Monitoring your Environment*.
process_environment_file.bat	This batch file processes the environment definition file and calls check_environment.bat. This batch file is called by monitor_environment.bat.
check_environment.bat	This batch file is called by process_environment_file.bat with the parameters of one line of the environment definition file. The batch file checks the parameters and either: • sets environment variables for subsequent calls • calls etbinfo to check if the broker/service is running

Examples

This environment has one broker:

```
ENVIRONMENT myProductionServers
ERROREXIT handle_error.bat
BROKER myProductionBroker localhost:1971
RPCSERVICE myRPCServer RPC SRV1 CALLNAT
```

This environment has multiple brokers:

```
ENVIRONMENT myMFServers
ERROREXIT handle_error.bat
BROKER myMFBroker ibm2:3930
SERVICE myACIServer ACLASS ASERVER ASERVICE
BROKER myMFBroker2 ibm2:3940
SERVICE myACIServer2 ACLASS ASERVER ASERVICE
RPCSERVICE myRPCServer2 RPC SRV2 CALLNAT
```

Monitoring your Environment

To monitor your environment

- Select option 8 from the *EntireX Command-line Script Menu* "Monitor your Environment".

  Or:
  Enter a command as shown below:

  ```
  monitor_environment.bat

  monitor_environment.bat <Time>

  monitor_environment.bat <Time> <EnvDefFile>
  ```

  where `<Time>` is the interval between checks in seconds (default 60)

  `<EnvDefFile>` is the file containing the definition of the environment (default *MyEnvironment.cfg*).

  Example:

  ```
  monitor_environment.bat 30 myEnvironmentDefinitionFile.txt
  ```

The following checks are performed:

- That the service is registered at the broker.
- That the server can be called. This is done with an RPC ping command.

A user exit specified in the environment definition file (see *Defining your Environment*) is called if a specified broker or service is not active. See *Error Handling* below.

Error Handling

A sample batch file `handle_error.bat` is provided to handle the situation where a component of a defined environment (see *Defining your Environment*) is not available. The environment definition file specifies the name of the error exit to be called. You can use this file as a template for your own exit to customize your error handling.

```
@echo off
@rem the following environment variables are set when the bat file is called
@rem environment variable %OBJECT% Error Object. possible values: BROKER or SERVICE
@rem the following environment variables are set for OBJECT SERVICE and OBJECT BROKER
@rem environment variable %ENV% logical name of environment
@rem environment variable %BNAME% logical name of Broker
@rem environment variable %BID% Broker ID
@rem the following environment variables are only set for OBJECT SERVICE
@rem environment variable %SNAME% logical service name
@rem environment variable %CLASS% Class
@rem environment variable %SERVER% Server
@rem environment variable %SERVICE% Service

echo Example User exit to handle errors: handle_error.bat
echo Error during check of Environment %ENV%
echo Broker %BNAME% (%BID%)

@rem check error object
@rem %OBJECT% == BROKER - Error Situation: defined Broker cannot be called
if %OBJECT%.==BROKER. goto Broker
@rem %OBJECT% == SERVICE - Error Situation: defined Service not registered
if %OBJECT%.==SERVICE. goto Service
echo Unknown Error Object %OBJECT%
goto end

:Broker
@rem the Broker (logical Name BNAME, Broker ID BID) is not running.
@rem add your code here to handle this situation

echo FATAL ERROR
echo Environment %ENV%
echo Broker %BNAME% ( %BID%) not active
goto end

:Service
@rem the Service (logical Name SNAME ,  CLASS /  SERVER / SERVICE ) on
@rem Broker (logical Name BNAME, Broker ID BID) is not running.
@rem add your code here to handle this situation

echo FATAL ERROR
echo Environment %ENV%
echo Service %SNAME% (%CLASS% / %SERVER% / %SERVICE% ) at Broker %BNAME% ( %BID%)  not registered
goto end


:end
@rem remove the pause so that monitoring of the environment can continue without a break
pause
```