Writing Applications: Attach Server

This chapter describes the programming of Attach Server for EntireX Broker. It assumes you are familiar with basic Broker ACI programming. The following topics are covered:

- Implementing an Attach Server
- Implementing Servers started by an Attach Server

Implementing an Attach Server

An attach server is a server that is capable of starting another server rather than handling service requests itself. See example under Attach Manager Interface. To implement an attach server, perform the following steps:

- Step 1: Register with EntireX Broker
- Step 2: Issue a Receive with Wait
- Step 3: Start Task
- Step 4: Deregister when the Work is Done

Step 1: Register with EntireX Broker

To register with EntireX Broker, the application has to add the `ATTACH` option to the `REGISTER` call. The `SERVER-CLASS`, `SERVER-NAME` and `SERVICE` parameters must reflect the service you can dynamically start. If the attach server is able to start several services, it has to register each service with the option `ATTACH` so that EntireX Broker knows exactly which services can be started by that attach server.

For example, an attach manager can start services (C1, N1, S1), (C2, N2, S2) and (C3, N3, S3). It therefore issues the following three registrations:

```
REGISTER  SERVER-CLASS=C1,SERVER-NAME=N1,SERVICE=S1,OPTION=ATTACH
REGISTER  SERVER-CLASS=C2,SERVER-NAME=N2,SERVICE=S2,OPTION=ATTACH
REGISTER  SERVER-CLASS=C3,SERVER-NAME=N3,SERVICE=S3,OPTION=ATTACH
```

Step 2: Issue a Receive with Wait

After all startable services have been registered by the attach server, the attach server must issue an unrestricted `RECEIVE` command in order to receive notification about queued service requests. The `RECEIVE` itself must be blocked for a certain time (`WAIT`=*nnn*). The attach server must be prepared to receive a notification for one of the announced services.

To continue the example from Step 1 above, the attach server now issues the `RECEIVE` command:

```
RECEIVE SERVER-CLASS=*,SERVER-NAME=*,SERVICE=*,WAIT=10M,RECEIVE-LENGTH=150
```

EntireX Broker answers either that no messages will be available after 10 minutes (error class 0074 is used for this kind of information) or that an attach service is required (error class 0010 and error code 0022), for example:

```
SERVER-CLASS=C2,SERVER-NAME=N2,SERVICE=S2,RETURN-LENGTH=116
```

with the following structure in the receive buffer, which is shown here in C programming language notation. The structure is the same for all programming languages and must be described in accordance with the programming language you select:

```
typedef struct
{
ETB_SHORT atm_version;   /*version of structure          */
ETB_SHORT atm_NotUsed;   /* alignment                    */
ETB_LONG  atm_nAttach;   /* # of failed server lookups   */
ETB_LONG  atm_nServer;   /* # of registered replicas     */
ETB_LONG  atm_nPendConv; /* # of pending conversations   */
ETB_LONG  atm_nActvConv; /* # of active conversations    */
ETB_CHAR  atm_server_class [S_SERVER_CLASS];/*class to attach        */
ETB_CHAR  atm_server_name  [S_SERVER_NAME]; /*server name to attach  */
ETB_CHAR  atm_service      [S_SERVICE];     /*service name to attach */
} ETB_ATMCB;
```

This structure contains the information necessary to decide whether a new replica needs to be started.

atm_nAttach

Number of client requests (`SEND CONVID=NEW`) the Broker could not schedule to a server immediately. After the Attach Manager has issued a `RECEIVE`, the value is reset to 0. If the Attach Manager does not issue its `RECEIVE`, this number shows the unreceived requests.

atm_nServer

Number of registered servers (replicas) minus those servers that are only finishing existing conversations (after issuing `DEREGISTER OPTION=QUIESCE`).

atm_nPendConv

Number of pending conversations, that is, client requests that could not currently be scheduled to a server. They are a subset of the active conversations.

atm_nActvConv

Number of the active conversations requesting a particular service.

Step 3: Start Task

This step depends very much on the platform. The attach server determines how to start up the desired application. The attach server only gets the logical name of the service. The mapping from the logical name to the program, including the path, startup parameters etc., must be performed by the attach server.

Step 4: Deregister when the Work is Done

Generally, attach servers are designed to "run forever". Once they are deregistered, no more services can be started on that platform automatically. However, if the administrator decides to shut down an attach server for whatever reason, he or she must DEREGISTER all registered services. There is no special flag for the deregistration.

After the final deregister, the attach server should perform a LOGOFF call to release all allocated resources:

```
DEREGISTER  SERVER-CLASS=C1,SERVER-NAME=N1,SERVICE=S1
DEREGISTER  SERVER-CLASS=C2,SERVER-NAME=N2,SERVICE=S2
DEREGISTER  SERVER-CLASS=C3,SERVER-NAME=N3,SERVICE=S3
```

or better

```
DEREGISTER SERVER-CLASS=*,SERVER-NAME=*,SERVICE=*
```

and as the last EntireX Broker-related command:

```
LOGOFF
```

Implementing Servers started by an Attach Server

In general, every server that can be used as a standalone server can be started up automatically. However, servers started by an attach server do not usually deregister and quit when no longer busy. They are not scalable, i.e. the number of replicas increases if not enough power is available, but the number does not decrease when there is no more work to be done.

To get around this situation, servers need to be prepared in such a way that they are started up automatically. Note the following points:

Notes:

1. The easiest server you can implement handles only one client for one conversation. After the last EOC, you can DEREGISTER or, preferably, LOGOFF the application and exit.
2. If you write an application that is automatically controlled by an attach server, try to implement the startup and the first RECEIVE as soon as possible. In other words, perform the necessary initialization after the conversation request is received.
3. Receive only the first call with the option NEW. Receive all subsequent calls with receive functions that are restricted to the established conversation (either with the option OLD, or with explicit restriction to the established conversation).
4. If you want to implement a server that does not exit after the first conversation, observe point 3 above. After the conversation has finished, set up the next RECEIVE with the option NEW. With this mechanism, the number of servers started in parallel corresponds to the number of clients trying to access the service simultaneously. This feature adapts the number of servers for high load peaks.
5. If you want to reduce the number of servers when they are no longer needed, set a proper RECEIVE timeout if you want to accept a new conversation, and finish your server if you actually receive a timeout. Both mechanisms give you the chance to react to load changes in both directions (increasing load and decreasing load).
6. Starting up a server for only one conversation is a simple server scheme, but you have to balance the simplicity of the application against the performance degradation for automatic startup. We

recommend you use purely automatic server startup for servers only when the conversation is expected to last a reasonable length of time.
If this is not clear, or if you want to run servers with short conversations - or even conversationless servers - you should consider using the method described under 4 and 5 above.