Modules Example

Introduction

This document describes an ECOA® client-server example named "*Modules Example*". This example is an extension to the "*Simple Example*" (ref. [3]) which uses the same system design, but replaces the server component with a different implementation.

The client-server model (ref. [2]) is one of the most basic data, task, or workload, distribution mechanisms in computing. Clients and servers may be distributed across a network, or they may reside on the same computing system. Service oriented concepts, which form a basis behind the ECOA, naturally fit with the client-server model, the clients referencing (using) the services provided by the server. Service orientation, and therefore the ECOA, goes on a step extra, in that a component can be a client (service user) to one or more other components, whilst simultaneously being a server (service provider) to others.

This document presents the principal user generated artefacts required to create the "*Modules Example*" client-server example using the ECOA. It is assumed that the reader is conversant with the ECOA Architecture Specification (ref. [1]) and the process of defining and declaring ECOA Assemblies, ASCs (components), Modules, and deployments in XML, and then using code generation to produce Module framework (stub) code units and ECOA Container and Platform code.

Aims

This ECOA "*Modules Example*" client-server example is intended to demonstrate an alternative implementation of a server component to that presented in the "Simple Example". The system design is exactly the same as that example with the only changes being the Server component implementation (and required deployment changes as a result).

ECOA Features Exhibited

- Composition of an ECOA Assembly of multiple ECOA ASCs (components).
- Contention-free resource sharing within an ECOA Assembly.
- Use of the ECOA runtime logging API.
- Module to Module operations

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Design and Definition

Client-Server Functional Design

The “Modules Example” client-server example will simply demonstrate a basic request-response mechanism. The client will periodically perform a request, from the server and will receive a data item in return (Figure 1).

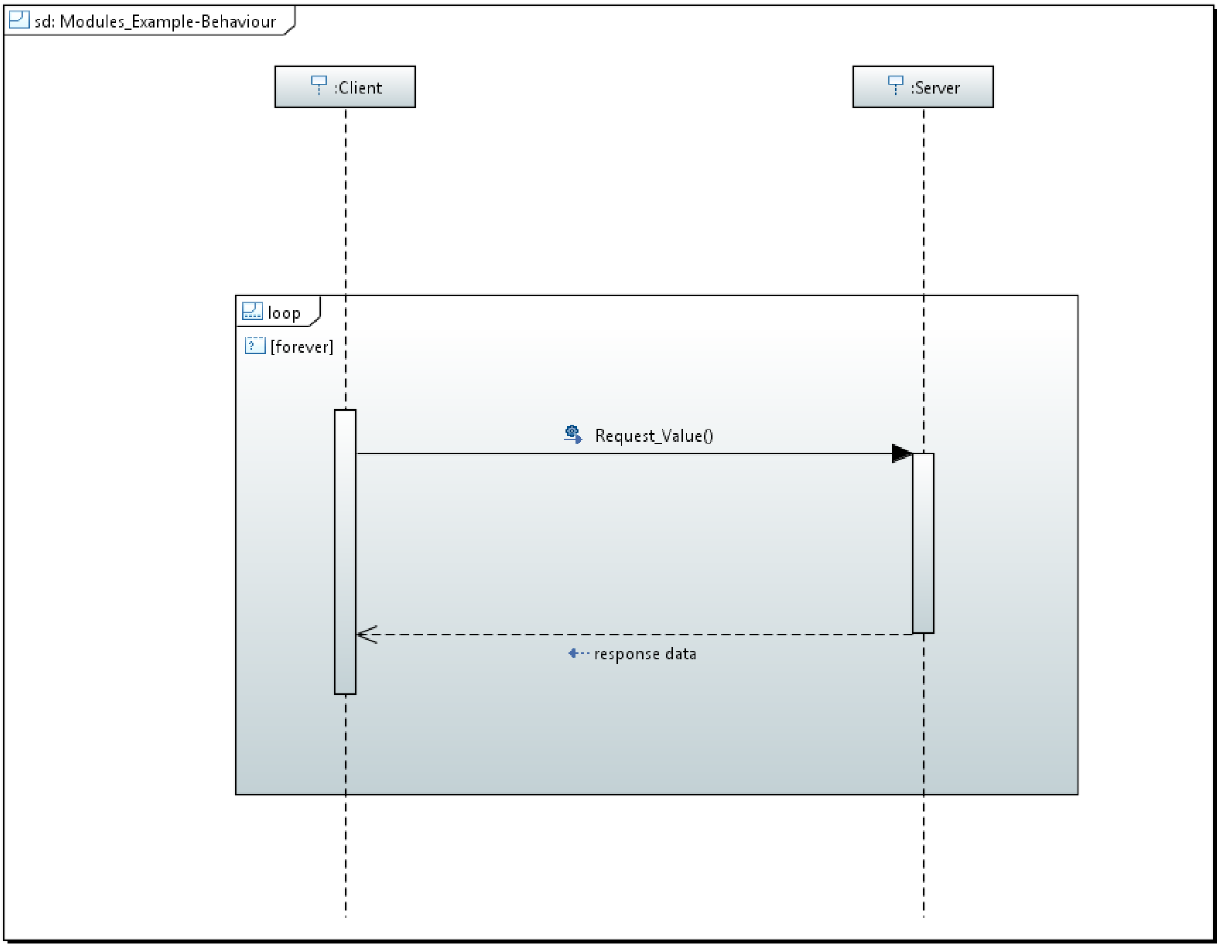


Figure 1 - ECOA "Modules Example" Client-Server Behaviour

The data content of the request will be the current absolute time and the response will be of a user defined type.

The Client will set a local variable to zero and output this to the log prior to performing the request. The result will be returned into this variable and logged.

The Client will be periodically activated at a rate of 0.5Hz (once every 2 seconds).

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an ‘as is’ basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

ECOA Assembly Design and Definition

This ECOA "*Modules Example*" client-server example ECOA Assembly comprises two ECOA ASCs named "*Client*" and "*Server*". The "*Client*" ASC type is instantiated once within the ECOA Assembly as "*Client_Inst*". The "*Server*" ASC is instantiated once within the ECOA Assembly as "*Server_Inst*" and provides the "*Provide_Value_Service*" ECOA Service, which is referenced (used) by the "*Client_Inst*" ASC (Figure 2).

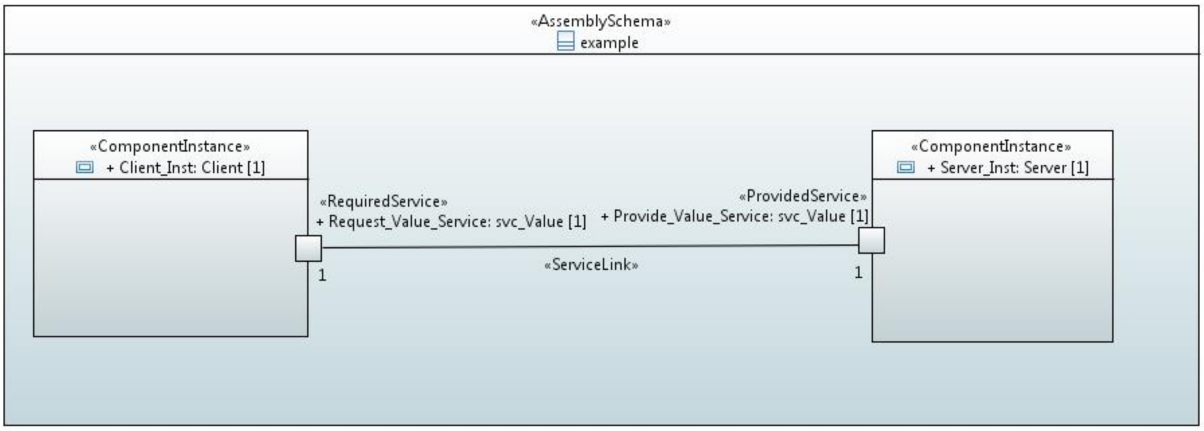

«AssemblySchema»
example
«ComponentInstance»
+ Client_Inst: Client [1]
«ComponentInstance»
+ Server_Inst: Server [1]
«RequiredService»
+ Request_Value_Service: svc_Value [1]
«ProvidedService»
+ Provide_Value_Service: svc_Value [1]
«ServiceLink»
1
1

Figure 2 - ECOA "Modules Example" Assembly Diagram

This ECOA Assembly is defined in an Initial Assembly XML file, and declared in a Final Assembly (or Implementation) XML file (which is practically identical). The Final Assembly XML for the ECOA "*Modules Example*" Assembly is as follows (file *example.impl.composite*):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<csa:composite xmlns:csa="http://docs.oasis-open.org/ns/opencsa/sca/200912"
        xmlns:ecoa-sca="http://www.ecoa.technology/sca-extension-2.0"
name="example"
        targetNamespace="http://www.ecoa.technology/dassault">

        <csa:component name="Client_Inst">
                <ecoa-sca:instance componentType="Client">
                        <ecoa-sca:implementation name="Client_Im" />
                </ecoa-sca:instance>
        </csa:component>

        <csa:component name="Server_Inst">
                <ecoa-sca:instance componentType="Server">
                        <ecoa-sca:implementation name="Server_Im" />
                </ecoa-sca:instance>
        </csa:component>

        <csa:wire source="Client_Inst/Request_Value_Service"
target="Server_Inst/Provide_Value_Service" />
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
</csa:composite>
```

The *Server* ASC type is defined in XML as follows (file *Server.componentType*):

```
<?xml version="1.0" encoding="UTF-8"?>
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ecoa-
sca="http://www.ecoa.technology/sca-extension-2.0">

        <service name="Provide_Value_Service">
                <ecoa-sca:interface syntax="svc_Value" />
        </service>

</componentType>
```

The ASC definition (the *<componentType>* XML element) declares the provision (by the ASC) of the *Provide_Value_Service* ECOA Service.

The *Client* ASC type is defined in XML as follows (file *Client.componentType*):

```
<?xml version="1.0" encoding="UTF-8"?>
<componentType xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ecoa-
sca="http://www.ecoa.technology/sca-extension-2.0">

        <reference name="Request_Value_Service">
                <ecoa-sca:interface syntax="svc_Value" />
        </reference>

</componentType>
```

This ASC definition (the *<componentType>* XML element) declares a reference (by the ASC) to the *Request_Value_Service* ECOA Service.

ECOA Service and Types Definition

The *svc_Value* Service, which is provided by the *Server* ASC and referenced by the *Client* ASC, is defined in a XML file (*svc_Value.interface.xml*):

```
<serviceDefinition xmlns="http://www.ecoa.technology/interface-2.0">

        <use library="example" />

        <operations>
                <requestresponse name="Request_Value">
                        <input name="Time" type="global_time" />
                        <output name="Value" type="example:value_type" />
                </requestresponse>
        </operations>

</serviceDefinition>
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

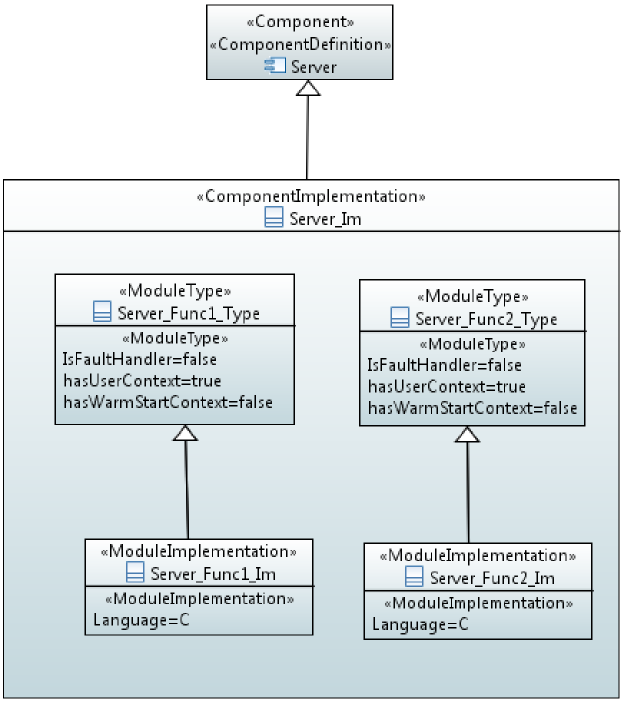
The Service comprises a single ECOA Request-Response Operation called *Request_Value* which has one input parameter (*Time* which is passed from the requesting client to the server), and one output parameter (*Value* which is the response from the server to the client). The first parameter is defined as being of type *global_time*, which is a pre-defined ECOA type. The second parameter is defined as being of type *example:value_type*, where *example* names a data types library *used* by the service definition. The data types library is, unsurprisingly, also defined in XML (file *example.types.xml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns="http://www.ecoa.technology/types-2.0">

	<types>
		<simple name="value_type" type="uint32" />
	</types>

</library>
```

The data type *example:value_type* is therefore an unsigned 32 bit integer type.

ECOA Module Design and Definition

The *Server* ASC (component) is composed of two Modules (Module Implementations *Server_Func1_Im* and *Server_Func2_Im* of Module Types *Server_Func1_Type* and *Server_Func2_Type* respectively) as illustrated in UML in Figure 3.



This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Figure 3 "Server" Module Design (as UML Composite Structure Diagram)

The *Client* ASC (component) is composed of a single ECOA Module (Module Implementations *Client_Module_Im* of Module Type *Client_Module_Type*) as illustrated in UML in Figure 4.

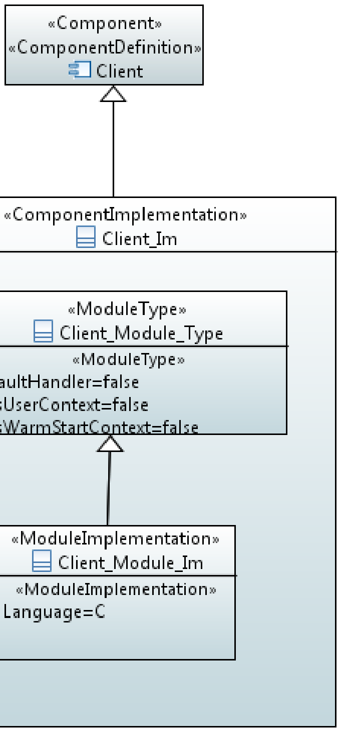


Figure 4 – "Client" Module Design (as UML Composite Structure Diagram)

Figure 5 and Figure 6 depict in UML the internal design of the *Server* ASC (component) ***providing*** the *svc_Value* ECOA Service, whilst the *Client* ASC ***references*** the Service. As always in the ECOA, the Module Implementations implement the Module Lifecycle operations defined by the ECOA.

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

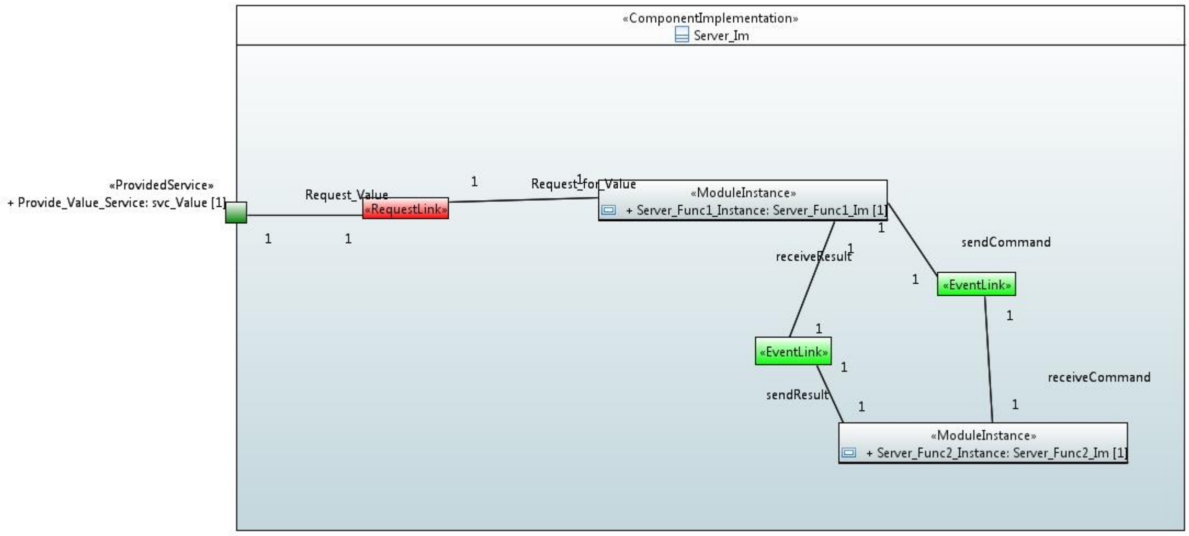


Figure 5 - "Server” Component Design (as UML Composite Structure Diagram)

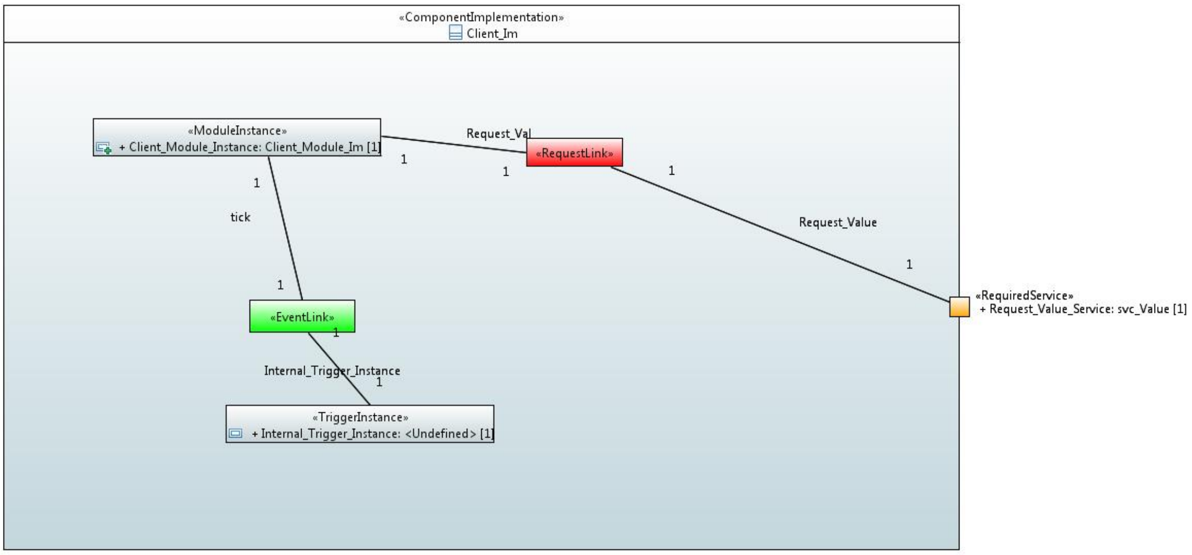

«ComponentImplementation»
Client_Im
«ModuleInstance»
+ Client_Module_Instance: Client_Module_Im [1]
Request_Val
«RequestLink»
Request_Value
tick
«EventLink»
Internal_Trigger_Instance
«TriggerInstance»
+ Internal_Trigger_Instance: <Undefined> [1]
«RequiredService»
+ Request_Value_Service: svc_Value [1]
1

Figure 6 - "Client” Component Design (as UML Composite Structure Diagram)

The Server ASC

The *Server* ASC is declared in XML as follows (file *Server_Im.impl.xml*):

```
<componentImplementation xmlns="http://www.ecoa.technology/implementation-2.0"
        componentDefinition="Server">

        <use library="example" />
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an ‘as is’ basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        <moduleType name="Server_Func1_Type" hasUserContext="true"
                hasWarmStartContext="false">
                <operations>
                        <requestReceived name="Request_for_Val">
                                <input name="time" type="global_time" />
                                <output name="val" type="example:value_type" />
                        </requestReceived>

                        <eventSent name="sendCommand">
                                <input name="new_time" type="global_time" />
                        </eventSent>

                        <eventReceived name="receiveResult">
                                <input name="result" type="uint32" />
                        </eventReceived>
                </operations>
        </moduleType>

        <moduleType name="Server_Func2_Type" hasUserContext="true"
                hasWarmStartContext="false">
                <operations>
                        <eventReceived name="receiveCommand">
                                <input name="new_time" type="global_time" />
                        </eventReceived>

                        <eventSent name="sendResult">
                                <input name="result" type="uint32" />
                        </eventSent>
                </operations>
        </moduleType>

        <moduleImplementation name="Server_Func1_Im"
                language="C" moduleType="Server_Func1_Type" />
        <moduleImplementation name="Server_Func2_Im"
                language="C" moduleType="Server_Func2_Type" />
        <moduleInstance name="Server_Func1_Instance"
                implementationName="Server_Func1_Im" relativePriority="2" />
        <moduleInstance name="Server_Func2_Instance"
                implementationName="Server_Func2_Im" relativePriority="3" />

        <requestLink>
                <clients>
                        <service operationName="Request_Value"
instanceName="Provide_Value_Service" />
                </clients>
                <server>
                        <moduleInstance operationName="Request_for_Val"
                                instanceName="Server_Func1_Instance" />
                </server>
        </requestLink>

        <eventLink>
                <senders>
                        <moduleInstance operationName="sendCommand"
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
                instanceName="Server_Func1_Instance" />
        </senders>
        <receivers>
            <moduleInstance operationName="receiveCommand"
                instanceName="Server_Func2_Instance" />
        </receivers>
    </eventLink>

    <eventLink>
        <senders>
            <moduleInstance operationName="sendResult"
                instanceName="Server_Func2_Instance" />
        </senders>
        <receivers>
            <moduleInstance operationName="receiveResult"
                instanceName="Server_Func1_Instance" />
        </receivers>
    </eventLink>

</componentImplementation>
```

That is, two Module Types (*Server_Func1_Type and Server_Func2_Type*) are declared.

Server_Func1_Type is a Module which has three operations specified:

- a *requestReceived* operation "*Request_for_Val*";
- an *eventSent* operation "*sendCommand*";
- an *eventReceived* operation "*receiveResult*".

This Module Type is implemented by a concrete Module Implementation *Server_Func1_Im* which in turn is instantiated once as the Module Instance *Server_Func1_Instance*.

Server_Func2_Type is a Module which has two operations specified:

- an *eventReceived* operation "*receiveCommand*";
- an *eventSent* operation "*sendResult*".

This Module Type is implemented by a concrete Module Implementation *Server_Func2_Im* which in turn is instantiated once as the Module Instance *Server_Func2_Instance*.

The *<requestLink>* XML logically associates the specific concrete operations of the Module Instance with the abstract Service operations. In this example, the "*Request_for_Val*" module operation (of *Server_Func1_Type*) is connected to the "*Request_Value*" service operation of the "*Provide_Value_Service*" service instance.

The two *<eventLink>* XML fragments logically associate the event send operation of *Server_Func1_Type* to the event received operation of *Server_Func2_Type* and vice-versa.

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

Two functional code units will be produced by the code generation process, implementing in code the concrete *Server_Func1_Im* and *Server_Func2_Im* classes, named "*Server_Func1_Im.c*" and "*Server_Func2_Im.c*" respectively (assuming the Module Implementation declaration has set the *Language* property to "C").

The Client ASC

The *Client* ASC is declared in XML as follows (file *Client_Im.impl.xml*):

```
<componentImplementation xmlns="http://www.ecoa.technology/implementation-2.0"
        componentDefinition="Client">

        <use library="example" />

        <moduleType name="Client_Module_Type" hasUserContext="false"
                hasWarmStartContext="false">
                <operations>
                        <eventReceived name="tick" />

                        <requestSent name="Request_Val" isSynchronous="true"
                                timeout="2">
                                <input name="time" type="global_time" />
                                <output name="val" type="example:value_type" />
                        </requestSent>
                </operations>
        </moduleType>

        <moduleImplementation name="Client_Module_Im"
                language="C" moduleType="Client_Module_Type" />
        <moduleInstance name="Client_Module_Instance"
                implementationName="Client_Module_Im" relativePriority="1" />
        <triggerInstance name="Internal_Trigger_Instance"
                relativePriority="2" />

        <requestLink>
                <clients>
                        <moduleInstance instanceName="Client_Module_Instance"
                                operationName="Request_Val" />
                </clients>
                <server>
                        <reference instanceName="Request_Value_Service"
                                operationName="Request_Value" />
                </server>
        </requestLink>

        <eventLink>
                <senders>
                        <trigger instanceName="Internal_Trigger_Instance" period="2"
/>
                </senders>
                <receivers>
                        <moduleInstance instanceName="Client_Module_Instance"
                                operationName="tick" />
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
        </receivers>
    </eventLink>

</componentImplementation>
```

That is, a Module Type (`Client_Module_Type`) is declared which has two operations:

- A "`Request_Val`" `requestSent` operation;
- The `eventReceived` operation "`tick`".

A timeout is defined for the "`Request_Val`" operation. This is to ensure that if the response is never received, the Module will not be blocked indefinitely. This scenario may occur if the request or response is lost, or if the Server Component fails to respond (in this example this may occur, as the receipt of the request causes the server to send an event and only respond once another event is received – as events are "fire and forget" this is not a robust server implementation!).

The `Internal_Trigger_Instance` Trigger Instance is introduced because the Client needs to "*periodically request a data item*" and so an ECOA periodic trigger is required. Once every period (2 seconds as set in the `<eventLink>` XML) the Trigger will fire and the Module Operation `tick` will be invoked.

This Module Type is implemented by a concrete Module Implementation `Client_Module_Im`, which in turn is instantiated once as the Module Instance `Client_Module_Instance`.

The `<requestLink>` XML logically associates the specific concrete operations of the Module Instance with the abstract Service operations. In this example, the "`Request_Val`" module operation is connected to the "`Request_Value`" service operation of the "`Request_Value_Service`" service instance.

A single functional code unit will be produced by the code generation process, implementing in code the concrete `Client_Module_Im` class, and named "`Client_Module_Im.c`" (assuming the Module Implementation declaration has set the `Language` property to "C").

ECOA Deployment Definition

The ECOA "*Modules Example*" Assembly is deployed (that is, the declared Module and Trigger Instances are allocated to a single ECOA Protection Domain, which is then allocated to a computing node) by the following XML (file `example.deployment.xml`):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<deployment xmlns="http://www.ecoa.technology/deployment-2.0"
    finalAssembly="example" logicalSystem="example">

    <protectionDomain name="Ex1">
        <executeOn computingPlatform="Example_Platform"
            computingNode="card1_bae" />
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
             <deployedModuleInstance componentName="Client_Inst"
                    moduleInstanceName="Client_Module_Instance"
modulePriority="11" />
             <deployedTriggerInstance componentName="Client_Inst"
                    triggerInstanceName="Internal_Trigger_Instance"
triggerPriority="12" />

             <deployedModuleInstance componentName="Server_Inst"
                    moduleInstanceName="Server_Func1_Instance" modulePriority="3"
/>
             <deployedModuleInstance componentName="Server_Inst"
                    moduleInstanceName="Server_Func2_Instance" modulePriority="3"
/>

      </protectionDomain>

      <platformConfiguration
             faultHandlerNotificationMaxNumber="8"
computingPlatform="Example_Platform"></platformConfiguration>

</deployment>
```

Thus in this case, a single ECOA Protection Domain is declared (*Ex1*) executing on an ECOA Computing Node, on a single ECOA Computing Platform.

Implementation

The Server ASC

The behaviour of each module of the *Server* ASC is described in detail in the following sections. Figure 7 depicts the behaviour of the component in the form of a UML sequence diagram.

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

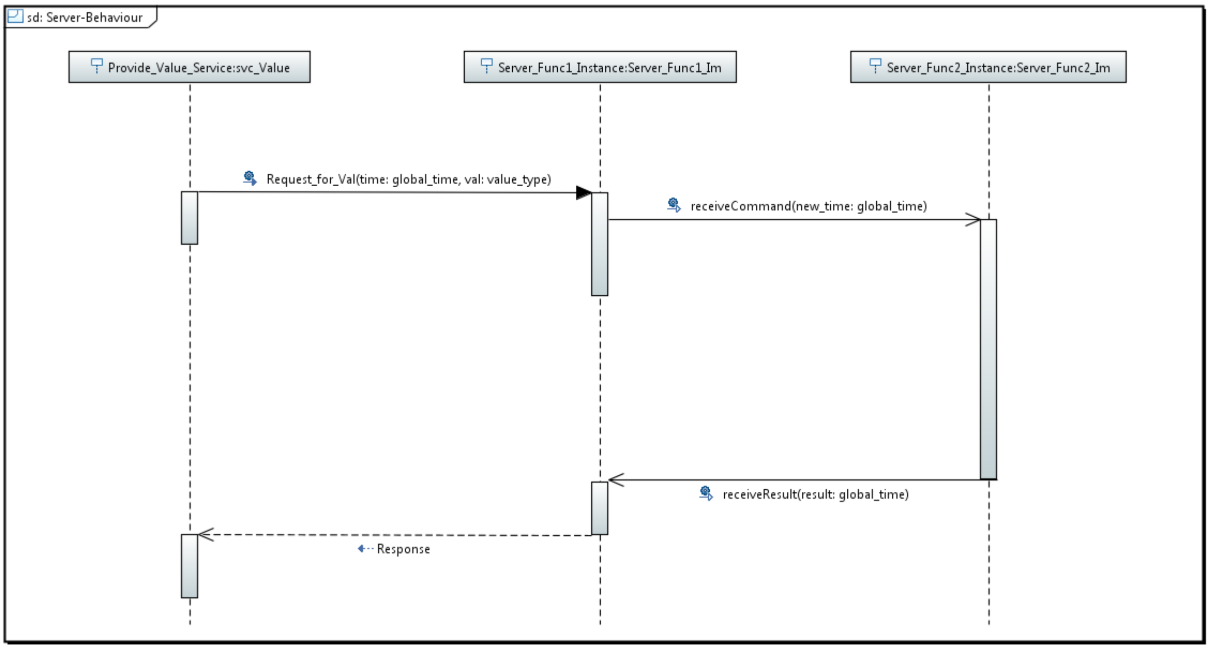

sd: Server-Behaviour
Provide_Value_Service:svc_Value
Server_Func1_Instance:Server_Func1_Im
Server_Func2_Instance:Server_Func2_Im
Request_for_Val(time: global_time, val: value_type)
receiveCommand(new_time: global_time)
receiveResult(result: global_time)
Response

Figure 7 - Server ASC Behaviour (as UML Sequence Diagram)

Server_Func1_Im Module

The "Request_Value_Service" Service request handler is implemented by the code function *Server_Func1_Im__Request_for_Val__request_received* in the (C) code unit *Server_Func1_Im.c*:

```
void Server_Func1_Im__Request_for_Val__request_received(Server_Func1_Im__context*
context, const ECOA__uint32 ID, const ECOA__global_time* time)
{
   context->user.ID = ID;

   Server_Func1_Im_container__sendCommand__send(context, time);
}
```

This function sends the "*sendCommand*" event operation (which is connected to the Server_Func2_Instance) by invoking the ECOA Container API function *Server_Func1_Im_container__sendCommand__send*. A response to the service request is not provided until the "*receiveResult*" event is received (from the Server_Func2_Instance) therefore the implementation of the server is asynchronous (note that the ID of the request is stored in the user context so it can be used when the response is sent. This implementation only supports one request being received at a time, as the ID will be overwritten if subsequent requests where received before the response is sent).

The "*receiveResult*" is implemented by the following (C) code:

```
void Server_Func1_Im__receiveResult__received(Server_Func1_Im__context *context,
const ECOA__uint32 result)
```

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
{
   ECOA__return_status return_status;

   return_status =
Server_Func1_Im_container__Request_for_Val__response_send(context, context-
>user.ID, result);
}
```

Server_Func2_Im Module

During initialisation, the module sets the user context data item to an initial value of 10. This is implemented by the code function *Server_Func2_Im__INITIALIZE__received* in the (C) code unit *Server_Func2_Im.c*:

```
void Server_Func2_Im__INITIALIZE__received(Server_Func2_Im__context *context)
{
   context->user.value = 10;
}
```

On receipt of the "*receiveCommand*" event (from *Server_Func1_Instance*) the module increments the current "*value*" and sends this back using the Container operation *Server_Func2_Im_container__sendResult__send*. This is implemented by the code function *Server_Func2_Im__receiveCommand__received*:

```
void Server_Func2_Im__receiveCommand__received(Server_Func2_Im__context *context,
const ECOA__global_time *new_time)
{
   context->user.value += 1;

   Server_Func2_Im_container__sendResult__send(context, context->user.value);
}
```

The Client ASC

All we need to do is program what to do when the *Internal_Trigger_Instance* Trigger Instance fires, i.e. to populate the *Client_SM_Im__tick__received* function stub.

```
void Client_Module_Im__tick__received(Client_Module_Im__context *context)
{
   ECOA__global_time time;
   ECOA__return_status return_status;
   example__value_type val;
   ECOA__log log;

   return_status = Client_Module_Im_container__get_absolute_system_time(context,
&time);

   val = 0;

   log.current_size = sprintf((char *) &log.data, "val before request = %d", val);
   Client_Module_Im_container__log_info(context, log);
```

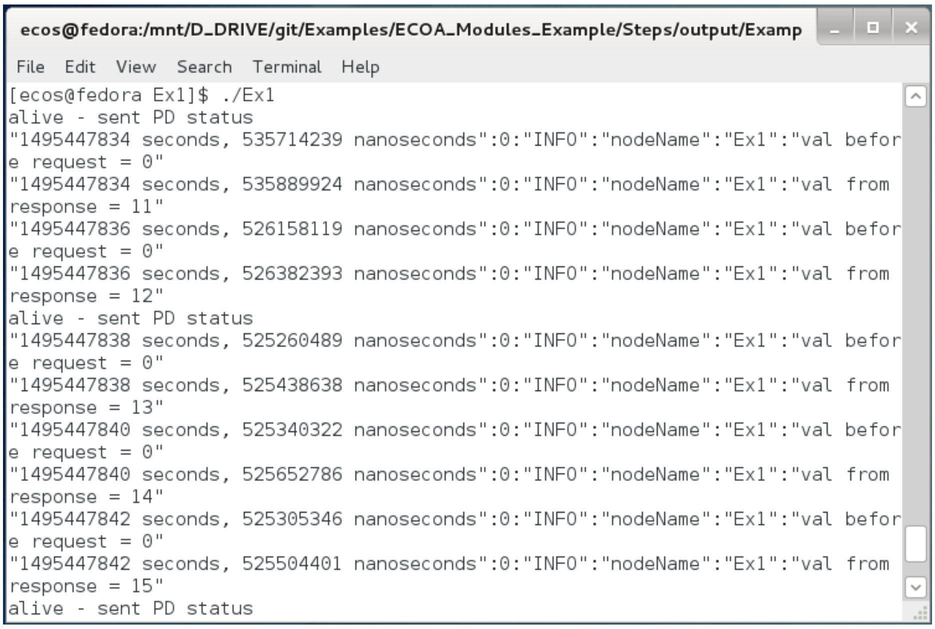
This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an 'as is' basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

```
   return_status = Client_Module_Im_container__Request_Val__request_sync(context,
&time, &val);

   log.current_size = sprintf((char *) &log.data, "val from response = %d", val);
   Client_Module_Im_container__log_info(context, log);
}
```

That is, the *val* variable is zeroed and logged prior to invoking the *Client_SM_Im_container__Request_Val__request_sync* API, and because a synchronous Request-Response call is made, the response (in variable *val*) is immediately available to log.

Program Output

When the ECOA “*Modules Example*” Assembly is built and run (in a single Node deployment), an output similar to Figure 8 should be achieved. The *Client* ASC outputs, at each iteration, both the value before sending the request message, and the value after receiving the response (note that the value is incremented with each request).


ecos@fedora:/mnt/D_DRIVE/git/Examples/ECOA_Modules_Example/Steps/output/Examp
File Edit View Search Terminal Help

```
[ecos@fedora Ex1]$ ./Ex1
alive - sent PD status
"1495447834 seconds, 535714239 nanoseconds":0:"INFO":"nodeName":"Ex1":"val befor
e request = 0"
"1495447834 seconds, 535889924 nanoseconds":0:"INFO":"nodeName":"Ex1":"val from
response = 11"
"1495447836 seconds, 526158119 nanoseconds":0:"INFO":"nodeName":"Ex1":"val befor
e request = 0"
"1495447836 seconds, 526382393 nanoseconds":0:"INFO":"nodeName":"Ex1":"val from
response = 12"
alive - sent PD status
"1495447838 seconds, 525260489 nanoseconds":0:"INFO":"nodeName":"Ex1":"val befor
e request = 0"
"1495447838 seconds, 525438638 nanoseconds":0:"INFO":"nodeName":"Ex1":"val from
response = 13"
"1495447840 seconds, 525340322 nanoseconds":0:"INFO":"nodeName":"Ex1":"val befor
e request = 0"
"1495447840 seconds, 525652786 nanoseconds":0:"INFO":"nodeName":"Ex1":"val from
response = 14"
"1495447842 seconds, 525305346 nanoseconds":0:"INFO":"nodeName":"Ex1":"val befor
e request = 0"
"1495447842 seconds, 525504401 nanoseconds":0:"INFO":"nodeName":"Ex1":"val from
response = 15"
alive - sent PD status
```

Figure 8 - ECOA "*Modules Example*" in Execution

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an ‘as is’ basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.

References

1	European Component Oriented Architecture (ECOA) Collaboration Programme: Architecture Specification (Parts 1 to 11) “ECOA” is a registered trade mark.
2	Client-server model *https://en.wikipedia.org/wiki/Client%E2%80%93server_model*
3	Simple Example. *http://www.ecoa.technology/tutorials.html*

This document is developed for and on behalf of BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd, and the copyright is owned by BAE Systems (Operations) Limited, Dassault Aviation, Bull SAS, Thales Systèmes Aéroportés, GE Aviation Systems Limited, General Dynamics United Kingdom Limited and Leonardo MW Ltd. This document is developed by BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems and is the Intellectual Property of BAE Systems (Operations) Limited, Military Air and Information, and Electronic Systems. The information set out in this document is provided solely on an ‘as is’ basis and the co-developers of this software make no warranties expressed or implied, including no warranties as to completeness, accuracy or fitness for purpose, with respect to any of the information.