Portrait
Foundation

EDGE2020 Implementation Guide

Edition 8.0

14 April 2014

Pitney Bowes
Software

Portrait Foundation

EDGE2020 Implementation Guide

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK

Portrait Software
The Smith Centre
The Fairmile
Henley-on-Thames
Oxfordshire, RG9 6AB, UK

Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778
Fax: +44 (0)1491 416601

America

Portrait Software
125 Summer Street
16th Floor
Boston, MA 02110
USA

Email: support@portraitsoftware.com
Tel: +1 617 457 5200
Fax: +1 617 457 5299

Norway

Portrait Software
Portrait Million Handshakes AS
Maridalsveien. 87
0461 Oslo
Norway

Email: support@portraitsoftware.com
Tel: +47 22 38 91 00
Fax: +47 23 40 94 99

About this document

Purpose of document

This document explains the capabilities of the EDGE-Portrait gateway and goes on to provide detailed instructions for implementers on how to make use of EDGE2020 to extend and evolve EDGE applications.

Intended audience

Members of project teams that intend to extend and evolve existing EDGE applications using Portrait Foundation.

Related documents

EDGE ETW-W Reference Manual

EDGE Client Reference Manual

Portrait Foundation Management Console online help

Portrait Foundation Installation Guide

Portrait Foundation Software Development Kit User Guide

Software release

Portrait Foundation 5.0 or later.

Contents

1 Introduction

This document describes how to create applications in which EDGE and Portrait Foundation interact using the EDGE-Portrait gateway. The EDGE-Portrait gateway allows the following types of interactions between the two systems:

Portrait calling EDGE

1. Portrait Foundation can invoke EDGE logics from within a Portrait Foundation process model. This allows Portrait Foundation to use EDGE business logic but does not allow EDGE user interface to be displayed.
2. Portrait Foundation can add call events to EDGE queues from within a Portrait Foundation process model to create a customer call-back that will be dealt with by an EDGE guide.
3. Portrait Foundation can display simple EDGE guides or subsets of guides in an HTA window.

EDGE calling Portrait

1. EDGE can invoke Portrait Foundation process models that have been exposed as a web service and do not contain any Portrait Foundation user interface. This allows EDGE to use Portrait Foundation business logic.
2. EDGE can invoke Portrait Foundation process models that do include user interface by launching a Portrait Foundation web page from an EDGE screen. This page can run Portrait Foundation process models or display Portrait Custom Controls as normal.

For each type of interaction there is a section in this document that:

- Summarises the capabilities of the EDGE-Portrait gateway
- Details the steps an implementer needs to take to make Portrait Foundation and EDGE interact
- Provides advice and guidelines on how your application should be implemented to interact in the required way.

2 Installation

2.1 Portrait Foundation

The EDGE-Portrait gateway is installed as an add-on component by the Portrait Foundation installation. You will be prompted to provide your licence key as part of the installation process. Please contact Portrait Support if you have not been provided with the licence key that is required.

There are three EDGE-Portrait gateway components that, by default, are selected for installation:

- **EDGE Interface**. This is the server component that handles all interactions initiated by Portrait to EDGE. This component is installed on each CRM server within your Portrait Foundation environment.
- **EDGE Configuration Nodes**. This component adds the ability to manipulate EDGE configuration items within the Configuration Suite.
- **.Net EDGE to Portrait Application**. This component is part of the Portrait Implementation install. It provides an application that facilitates EDGE launching Portrait Foundation windows (see section 7 for details).

You can choose not to install any of these components by deselecting the items on the relevant **Select Components** installation dialog.

2.2 EDGE

Please refer to EDGE documentation for instructions on how to install the EDGE components of the EDGE-Portrait gateway.

2.3 Pre installation tasks

EDGE2020 customers using the web enabled EDGE client must ensure that all Web Servers have IIS 6 Management Compatability (IIS Metabase) enabled and that the Microsoft Internet Explorer Web Controls v1.0 are installed. See the *Foundation Installation Guide* for more information.

2.4 Post installation tasks

2.4.1 Using different versions of EDGE

The EDGE interface installed when running the Portrait installation enables communication between Portrait and EDGE version 7.4.1. If you are using a different version of EDGE, then you need to update the EDGE interface installation. To do this:

1. Ask EDGE Support for the EDGE interface DLLs that are compatible with your version of EDGE. These DLLs comprise **EOAPI.dll**, **ECCAPI.dll** and **EDGEAPI.dll**
2. Shut down Portrait services
3. Copy **EOAPI.dll**, **ECCAPI.dll** and **EDGEAPI.dll** to the **\Common\bin** directory beneath the Portrait installation folder
4. Re-start Portrait services

2.4.2 Adding EDGE integration configurations assets

The configuration assets that accompany the EDGE-Portrait gateway are provided by the following two packages:

- EDGE Integration (Foundations). This package provides the basic EDGE integration capability. For example, it provides all the EGDE nodes that can be used in Portrait Foundation process models.
- EDGE Integration (Supporting utilities). This package provides additional configuration to support building Portrait Foundation applications that interact with EDGE. In particular, it provides support for a Portrait HTA application that emulates the EDGE GEO client.
- To add this configuration to your workspace, include the relevant package.

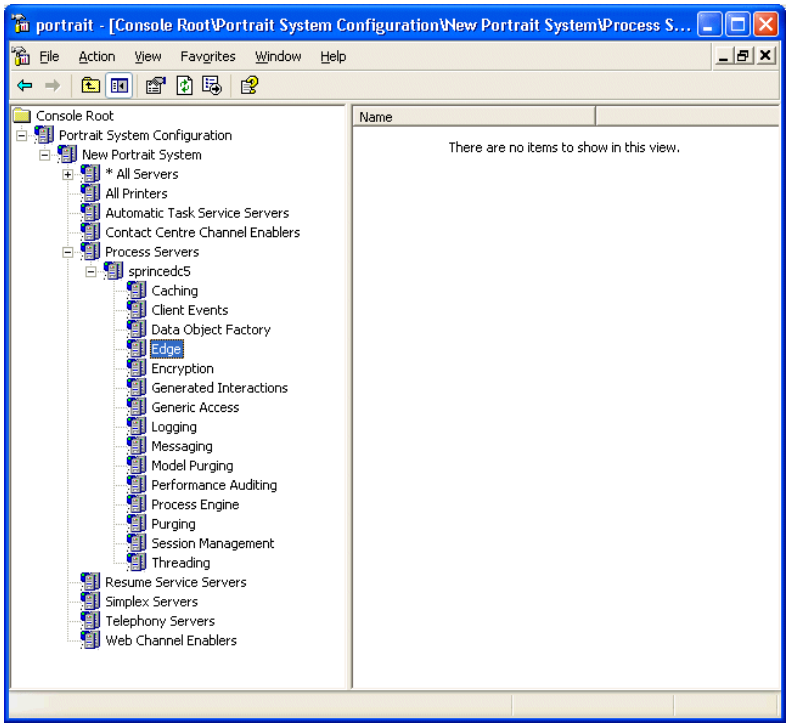
2.4.3 Configuring access to EDGE servers

The EDGE servers that Portrait Foundation is to access need to be set up within the Portrait Management Console. This is a two-step process:

- Configure the Portrait Foundation process servers that are used as gateway servers to the EDGE servers
- Configure the sessions that are used to communicate with the EDGE servers

Configuration is performed through the EDGE property pages which are accessed by right clicking the **EDGE** tree item under the relevant Portrait Foundation process server, and selecting **Properties**. The **EDGE** tree item is only available for Portrait Foundation process servers on which the EDGE gateway is installed.

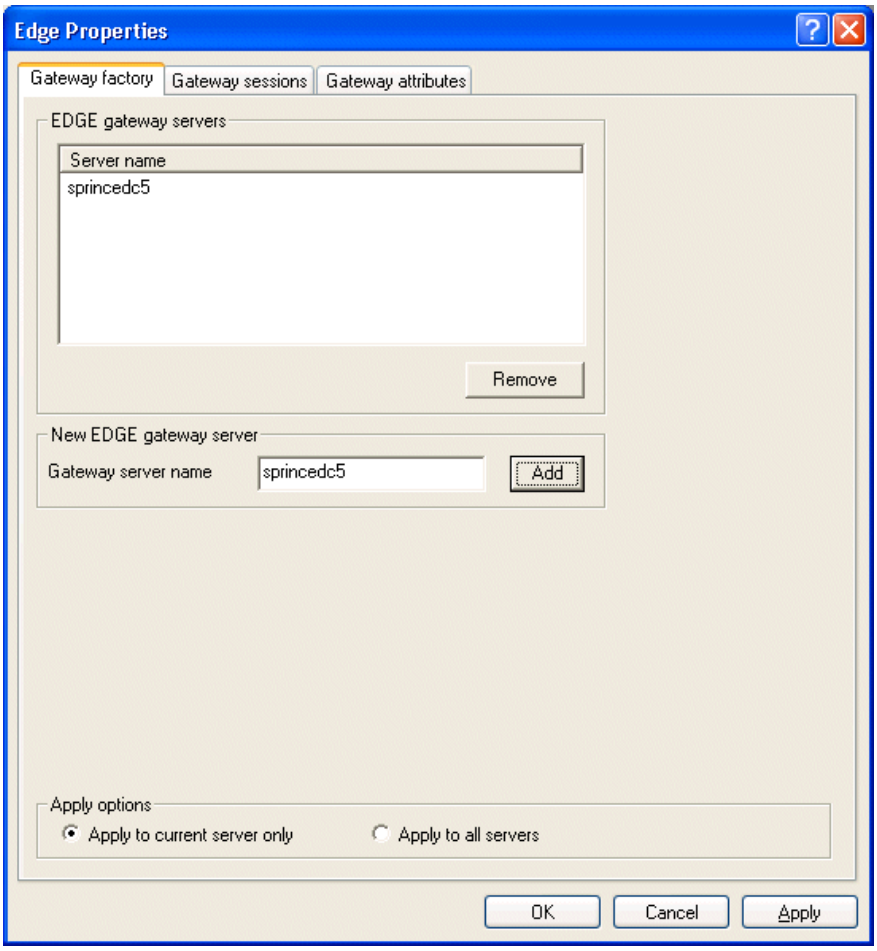
Figure 1 – EDGE node in Management Console



Managing gateway servers

EDGE gateway servers are Portrait Foundation process servers which host a service that communicates with EDGE servers. EDGE gateway servers are identified to the system through the **Gateway factory** tab.

Figure 2 – Managing gateway servers



Add a gateway server by entering the name of the Portrait Foundation process server that is acting as a gateway server in the edit field, and then click the **Add** button.

A Portrait Foundation process server will always attempt to use a local EDGE gateway if the local machine is configured as an EDGE gateway server. If the local machine is not configured as an EDGE gateway server then a round robin approach is taken on the configured gateway servers.

Multiple gateway services

The EDGE gateway maintains the state of each session user's desktop in memory. The memory used in a large system may result in the virtual address space required by the Portrait HostU.exe service becoming too large (>2Gb). To overcome this, the EDGE Gateway may be configured to run in one or more separate instances of HostU.exe. (See the Portrait Installation Guide for details of Portrait service configuration.)

Where multiple gateway services are configured, each gateway component is registered with its own PROGID:

<PROGID>AIT.AMC.Edge.Gateway.Service2</PROGID>

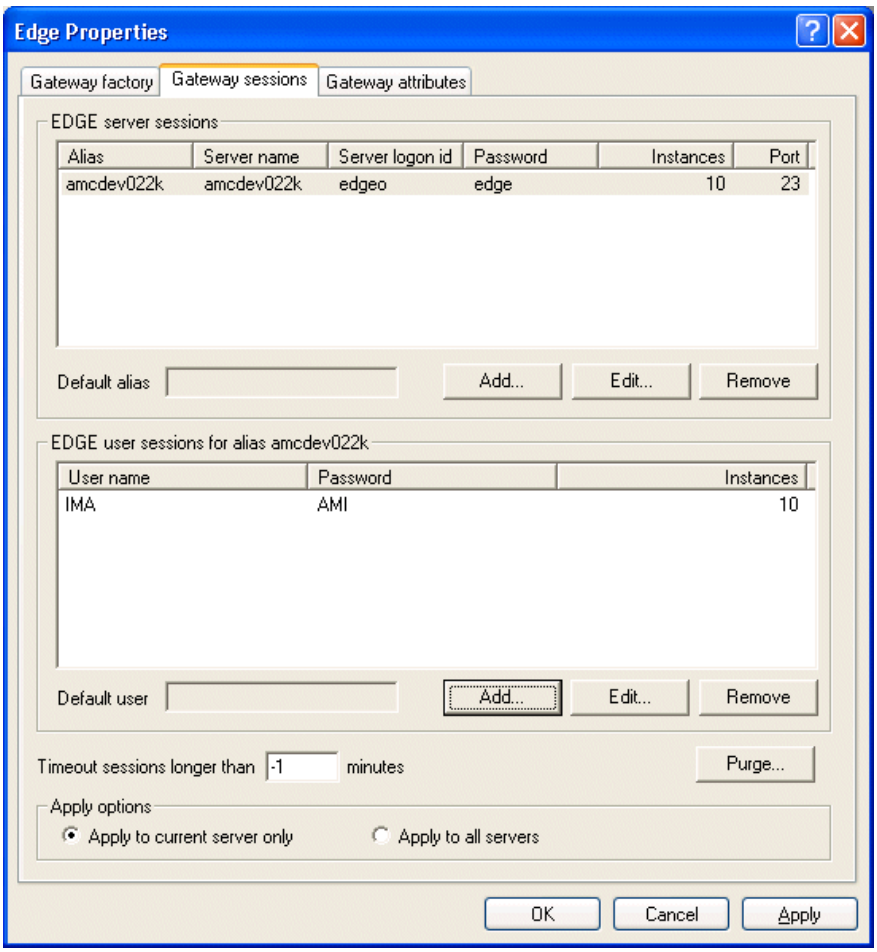
In this configuration each service should be configured in the **Gateway factory** tab. To add a gateway service, enter the name of the Portrait Foundation process server plus the final suffix from the PROGID of the gateway component for that service, e.g.:

ServerName.Service2

Managing EDGE sessions

An EDGE session holds the information required to log on to an EDGE server and processes work against the EDGE server using a given user id and password. When a gateway server is started it will create a pool of EDGE sessions using the configured connection information. EDGE sessions are configured through the **Gateway sessions** tab.

Figure 3 – Managing EDGE sessions



The **Gateway sessions** tab displays two sets of information. The **EDGE server sessions** list displays the information used to log on to an EDGE server. Each column is described in the following table:

Table 1 – EDGE server sessions

Column	Description
Alias	A unique name that identifies the set of EDGE server connection information
Server name	The name of the EDGE server
Server logon id	The id used to log on to the EDGE server
Password	The password that corresponds to the Server logon id
Instances	The number of instances of this session that will be created in the pool of EDGE sessions maintained by the gateway server that is being configured
Port	The port that will be used to communicate with the EDGE server

A default alias can be configured although it is not mandatory. The default alias is used by the gateway server if an alias is not supplied when attempting an EDGE operation.

More than one EDGE server may be configured for each alias to allow connections to be distributed across multiple EDGE servers. Connections will be allocated across servers round-robin.

Note that when adding a gateway session, the low level timeout (ECCAPI) for the EDGE server session may be also configured.

The **EDGE user sessions** list displays EDGE user connection information. The users displayed correspond to the selected EDGE server session. When a user session is created, the corresponding EDGE server session information is used to connect to the EDGE server before the user is logged on. Each column is described in the following table:

Table 2 – EDGE user sessions

Column	Description
User name	The EDGE user name
Password	The password that corresponds to the User name
Instances	The number of instances of this session that will be created in the pool of EDGE sessions maintained by the gateway server that is being configured

A default user name can be configured although it is not mandatory. The default user name is used by the gateway server if a user name is not supplied when attempting an EDGE operation.

Add, edit or remove EDGE server sessions and EDGE user sessions using the corresponding **Add**, **Edit** and **Remove** buttons.

Other options on the **Gateways sessions** tab relate to returning sessions that have been active for a given time period back to the pool of available sessions. These are described in the following table:

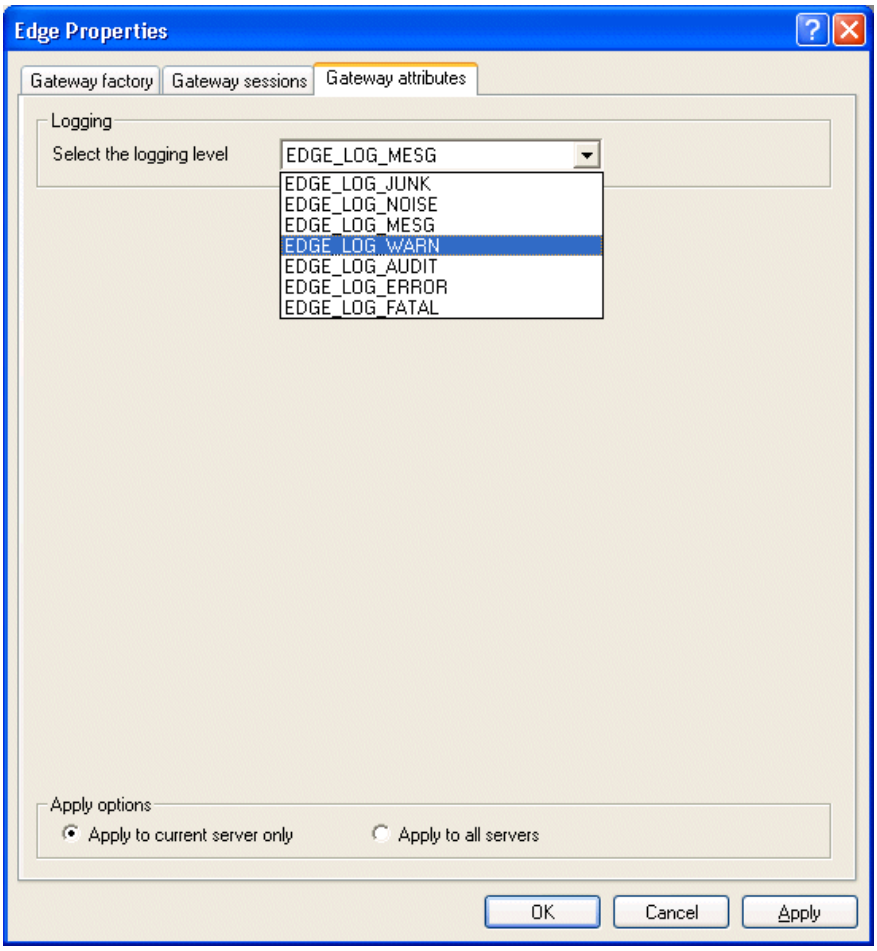
Table 3 – Removing sessions from the pool

Field	Description
Timeout sessions	Use this field to set the number of minutes a session on which there is no activity will remain available. Once this period has expired the session is returned to the pool of available sessions. Set this value to 9999 if you are working in a development environment and do not want sessions returned to the pool.
Purge	Use this option to perform an immediate timeout of sessions on which there has been no activity for the given period in minutes. Purged sessions will be made available again in the session pool.

Managing gateway attributes

The **Gateway attributes** tab allows configuration of attributes that define the way the gateway server being configured behaves.

Figure 4 – Managing gateway attributes



These attributes are described in the following table:

Table 4 – Gateway attributes

Attribute	Description
Logging level	Use this option to select the amount of EDGE logging information that the gateway server will output

2.4.4 Viewing logs generated by gateway servers

Gateway servers generate diagnostics logs that provide information on the gateway's activity and any errors that a gateway has encountered. To enable this logging, append the following logging filter criteria string to the filter that you are using when generating Portrait Foundation logs:

[(*):(*):(*):(44,45,46,47)]

Please refer to the Filter Criteria section of the *Portrait Management Console online help* for details on how to configure Portrait Foundation logging levels within the Management Console.

2.5 Foundation SDK

2.5.1 Build Environment

To create your own EDGE2020 implementation install that includes the web enabled EDGE client, you need to use SDK Build Enviroment. For details on how to install this please refer to the *Software Development Kit User Guide*.

At mimimum you need to create a simple Portrait Foundation Web Application that includes a web.config file with the relevant EDGE client settings. Use the Configuration Build Tool to add the CSPROJ to **components.txt** setting the Application name to **Edge2020**. This should generate an implementation install with a link to the Edge2020 web enabled client.

NB: The EDGE2020 implementation install generated by the SDK requires that Microsoft Internet Explorer Web Controls v1.0 is installed on the Web Server. Before this can be installed, IIS 6 Management Compatability (IIS Metabase) must first be enabled. See the *Foundation Installation Guide* for more information.

2.5.2 Edge to Portrait Template

The Edge to Portrait template can be installed using the Foundation SDK install and is required to make use of the Portrait browser ActiveX control (section 7.2) from within the EDGE GEO client.

3 Portrait Invoking EDGE Logics

3.1 Capabilities

Portrait Foundation can invoke EDGE logics from within a Portrait Foundation process model using the EDGE Logic node.

This allows Portrait Foundation to read or set data available to EDGE and run business logic contained in EDGE logics.

The call to the EDGE logic is synchronous and Portrait Foundation can pass input parameters to the EDGE logic as necessary. When the logic completes, any outputs will be passed back to Portrait Foundation.

3.1.1 Restrictions

The EDGE logics to be invoked by Portrait Foundation need to be 'atomic', that is they need to provide some functionality that makes sense as a discrete unit that doesn't rely on other parts of an EDGE guide being run before or afterwards. This is because subsequent calls from Portrait to EDGE will not have access to any state or session information set as a result of running the logic.

In order to handle scenarios where state information needs to be set up before a logic can run or where several logics need to share data and be run sequentially, it will be necessary to create a new logic that wraps these statements so as to appear as an atomic unit to Portrait Foundation; reading inputs, setting variables, calling the logics in the correct order, reading variables and returning the values as outputs to Portrait Foundation as necessary.

Any EDGE logic that can be run as a result of Portrait Foundation invoking it (or Portrait Foundation invoking another logic that in turn calls it) must not contain any user interface verbs. Whilst EDGE logics cannot contain screen definitions, they may contain verbs that cause message boxes or pop-ups to be displayed by GEO. If the EDGE-Portrait gateway encounters one of these verbs then an error will be generated, therefore, logics run in this way must be stripped of these types of verbs. Note that invoking an EDGE logic from Portrait Foundation causes the initialisation processing for the guide that houses the target logic to be executed. Hence this initialisation processing must also not contain any user interface verbs.

3.2 Using the EDGE Logic node

3.2.1 Overview

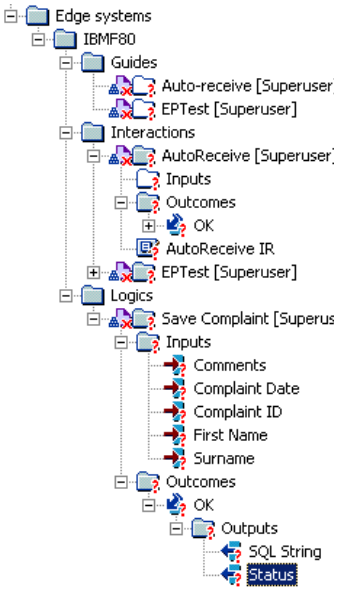
Making use of the EDGE Logic node comprises two separate configuration activities:

- Configure EDGE logic within the Configuration Suite workspace
- Drop the EDGE specific nodes required to launch the logic into a Portrait Foundation process model

The following sections describe how to achieve these steps.

3.2.2 Configuring the logic

Figure 5 – EDGE system configuration



To set up a new logic definition within the Configuration Suite, it is necessary to:

- Create a new EDGE System (if one does not already exist) to represent the EDGE server with which Portrait Foundation is interacting. You will be prompted for a name to represent the EDGE system.
- Add an EDGE guide definition. This definition represents the EDGE Guide that will be started when Portrait Foundation interacts with EDGE. This definition needs to be present whether you using Portrait Foundation to run logics (using the EDGE Logic node) or launching EDGE interactions (using the EDGE Guide node). You will be asked to supply:
  - A name to represent the guide within Portrait Foundation
  - The name of the guide on the EDGE server that houses the logic you wish to invoke
- Add the EDGE logic. You will asked to supply:
  - A name to represent the logic within Portrait Foundation.
  - The name of the logic on the EDGE server.
  - The EDGE guide within which the logic resides. You are presented with a list of the EDGE guide definitions that you have configured.
- Configure the inputs and outputs of the EDGE logic. Configuring the interface to a logic is identical to setting up the interface to a Portrait Foundation process model. However it is important that the system name specified for each input/output is identical to the EDGE field name that you wish Portrait Foundation to set/read when running the logic in EDGE. For example, if the EDGE logic reads an EDGE field called EPTest-EPComments then you must

specify EPTest-Comments as the system name of an input to the logic you are configuring in Portrait Foundation.

3.2.3 Connecting to EDGE

Log on

Prior to launching an EDGE logic it is necessary to log on to the EDGE server. This is achieved using the EDGE Logon node. This node accepts user/password and details of the EDGE system to connect to (identified by its Management Console alias, as described in section 2.4.3). Each connection to EDGE is identified by the EDGE Session Id returned by the gateway. This identifier is a required input to all other nodes and is an output of this node.

This node is configured with:

- EDGE system
- EDGE operator id
- EDGE operator password

If any of these are not configured for the node, they become inputs to the node. In addition the node has an EDGEWorkstationId input that can be used to supply the EDGE workstation identifier that is in use by telephony. For more details, please refer to the *EDGE Telephony Gateway* documentation.

The node always has two optional outcomes for the single (non configurable) output **OK**. These are

- EDGEOutputXml: XML describing the established connection
- EDGESessionId: The identifier that must be supplied to subsequent nodes in order to use the logon established by this node

Log off

Any EDGE connection established using the EDGE Logon node should be subsequently closed. This is achieved using the EDGE Logoff node. This node has no configuration. It has one single input – EDGESessionId – the EDGE Session Id returned from a previous Logon node. It has no outcomes or outputs.

3.2.4 Running an EDGE logic

The EDGE Logic node is used to execute a named EDGE logic. The node will then adopt the inputs and (outcomes and) outputs of the named logic. Unlike the EDGE Guide node, this node does not suspend as logics are seen as analogous to single method calls.

This node is configured with:

- EDGE system (mandatory)
- EDGE logic name (mandatory)

The node adopts the inputs and outcome (always **OK**) and outputs that have been configured for the selected logic.

3.3 Integration considerations

EDGE logics that are invoked from Portrait Foundation must have a clearly defined interface in terms of the inputs that Portrait Foundation will provide, and the outputs that are returned from EDGE. If you wish to invoke a pre-existing EDGE logic that does not have such an interface, it may be necessary to configure a wrapper logic in EDGE which calls the target logic. This wrapper logic will expose the interface required by Portrait Foundation and ensure that additional EDGE field values that are not supplied by Portrait Foundation are set before the target logic is called.

When Portrait Foundation invokes an EDGE logic, the whole guide that houses the target logic is started. Hence, for performance reasons, if you wish to repeatedly invoke a logic from Portrait Foundation, you should minimise initialisation processing carried out by the guide that houses the logic.

4 Portrait Adding Call Events to EDGE Queues

4.1 Capabilities

Portrait Foundation can add call events to EDGE call queues from within a Portrait Foundation process model by invoking the 'EDGE queue' logic from the EDGE Logic node.

This allows Portrait Foundation to schedule an outbound call that will be handled by the EDGE system.

4.1.1 Restrictions

In the first release of the EDGE-Portrait gateway it will not be possible for Portrait Foundation to access other EDGE call list management functionality such as subsequent call sequencing, re-prioritisation, record retrieval from queues, transfer of records from one call queue to another, removal of records from queues and so on. The gateway will simply allow Portrait Foundation to schedule an outbound call and will not be able to track or modify the subsequent status of the call (unless tracking information is available in a data source accessible to Portrait via another mechanism).

4.2 Using the EDGE Queue logic

4.2.1 EDGE configuration

It is assumed that the ETW development environment is available and the necessary EDGE server configuration and logins have been set up.

4.2.1.1 Database

Fields

The following fields should be added to the EDGE project, this is achieved by going to the **Database** area, **File/Fields** tab and clicking **Add Field**.

These fields can be added to any database file, but in the CreateCall logic detailed below it is assumed that the TEMP file has been used. If another file is used then all references to TEMP in the CreateCall logic and in the Portrait Foundation configuration should be replaced with the correct file name.

The default settings can be used for all fields except for ErrString, which should be given a length of 100, and PhoneNum, which should be given a length of 20.

- QueueName
- PhoneNum
- SchedDate
- SchedTime
- TimeZone
- CntryCode
- DST
- UseCustTZ
- ErrStatus
- ErrString

When all the fields have been added, click **Assign Field Positions** on the **File/Fields** tab.

4.2.1.2 Guide

Logics

CreateCall is the logic which Portrait Foundation will call.

To add a logic, in the **Guide** area, **Logic** tab, click **Add**, type the name of the logic, and then paste the code given below into the code area.

CreateCall

```
-- Logic: CreateCall
-- Description: Creates a callback in a generic way, taking parameters from fields in the
TEMP table
-- Inputs: (See ETW documentation for more information about required formats, meanings of
codes and so on)
--         TEMP-QueueName: the name of the EDGE queue to place the callback on
--         TEMP-SchedDate: the date on which to schedule the callback in any external EDGE
format (e.g. MM-DD-YYYY)
--         TEMP-SchedTime: the time at which to schedule the callback in any external EDGE
format (e.g. hh:mm:ss)
--         TEMP-PhoneNum: the phone number to call
--         TEMP-DST: is daylight savings time observed? ("1" means it is observed)
--         TEMP-TimeZone: customer time zone - the number of hours past Greenwich Mean
Time
--         TEMP-CntryCode: country code, as configured in EDGE

-- Copy inputs from TEMP file into local variables
ALWAYS COPY the value TEMP-QueueName into @QueueName
ALWAYS COPY the value TEMP-SchedDate into @SchedDate
```

```
ALWAYS COPY the value TEMP-SchedTime into @SchedTime
ALWAYS COPY the value TEMP-PhoneNum into @PhoneNum
ALWAYS COPY the value TEMP-DST into @DST
ALWAYS COPY the value TEMP-TimeZone into @TimeZone
ALWAYS COPY the value TEMP-CntryCode into @CntryCode
ALWAYS COPY the value TEMP-UseCustTZ into @UseCustTZ

-- Must convert date and time to internal format
-- TO DO: what if input date or time is invalid?
ALWAYS DATE-FORMAT the date @SchedDate to INTERNAL store result in @InternalSchedDate
ALWAYS TIME-FORMAT the time @SchedTime to INTERNAL store result in @InternalSchedTime

-- TO DO: What about (if any) current existing status record?  Need to save and restore
later?

-- Create the callid.  Use * to generate a system call id
ALWAYS CALL-EVENT command create call IDs with IDs * store status in @ErrStatus

IF COMPARE if the value @ErrStatus is = the value “”
-- null status means everything is OK
  THEN GOTO label {OK1}

-- We have encountered a problem, so we will set up error message and then exit
IF COMPARE if the value @ErrStatus is = the value “1004”
  THEN COPY the value “Create call ids error: One or more call events for the call ID
already exist” into @ErrString
  THEN GOTO label {ExitLabel}
IF COMPARE if the value @ErrStatus is = the value “1006”
  THEN COPY the value “Create call ids error: Call event or ID is null” into @ErrString
  THEN GOTO label {ExitLabel}
ALWAYS COPY the value “Create call ids: Unknown error” into @ErrString
ALWAYS GOTO label {ExitLabel}

------------------------------------------------------------------------------
LABEL {OK1}

-- Store new call id in local variable
ALWAYS COPY the value $CALLID into @CALLID

-- Schedule call event – need to copy some inputs into fields in the STATUS file first
ALWAYS COPY the value @DST into STATUS-USESDST
ALWAYS COPY the value @TimeZone into STATUS-ITZC
ALWAYS COPY the value @CntryCode into STATUS-COUNTRY
ALWAYS COPY the value @PhoneNum into STATUS-TELNO
ALWAYS CALL-EVENT command schedule call events with events @CALLID queue @QueueName date
@InternalSchedDate time @InternalSchedTime use customer timezone @UseCustTZ store status
in @ErrStatus

IF COMPARE if the value @ErrStatus is = the value “”
-- null status means everything is OK
  THEN GOTO label {OK2}

-- We have encountered a problem, so we will set up error message and then exit
IF COMPARE if the value @ErrStatus is = the value “1005”
  THEN COPY the value “Schedule call events error: The call event does not exist” into
@ErrString
  THEN GOTO label {ExitLabel}
IF COMPARE if the value @ErrStatus is = the value “1008”
```

```
  THEN COPY the value "Schedule call events error: Telephone number is null or invalid"
into @ErrString
  THEN GOTO label {ExitLabel}
IF COMPARE if the value @ErrStatus is = the value "1101"
  THEN COPY the value "Schedule call events error: Invalid queue specified. The queue does
not exist." Into @ErrString
  THEN GOTO label {ExitLabel}
ALWAYS COPY the value "Schedule call events: Unknown error" into @ErrString
ALWAYS GOTO label {ExitLabel}

------------------------------------------------------------------------
LABEL {OK2}

-- Unlock the call event
ALWAYS CALL-EVENT command unlock call events with events @CALLID store status in
@ErrStatus

IF COMPARE if the value @ErrStatus is = the value ""
-- null status means everything is OK
  THEN GOTO label {ExitLabel}

-- We have encountered a problem, so we will set up error message and then exit
-- TO DO: Other cleanup required in this case to clean the call id created previously

IF COMPARE if the value @ErrStatus is = the value "1005"
  THEN COPY the value "Unlock call events error: The call event does not exist" into
@ErrString
  THEN GOTO label {ExitLabel}
IF COMPARE if the value @ErrStatus is = the value "1007"
  THEN COPY the value "Unlock call events warning: Call event or ID was not locked.
(Warning only)" into @ErrString
  THEN GOTO label {ExitLabel}
ALWAYS COPY the value "Unlock call events: Unknown error" into @ErrString
ALWAYS GOTO label {ExitLabel}

------------------------------------------------------------------------
LABEL {ExitLabel}

ALWAYS COPY the value @ErrStatus into TEMP-ErrStatus
ALWAYS COPY the value @ErrString into TEMP-ErrString

-- TO DO: Now to restore any pre-existing status/callback information.... restore STATUS?
Is this needed?
```

```
ALWAYS DEBUG
```

4.2.1.1 Queues

There must be at least one EDGE queue defined where Portrait Foundation will be able to place call-backs.

To create a new queue, in the **Queues** area of the project, **Queue Definitions** tab, click **Add** and create a queue with Time Ordering and Open Access.

4.2.1.2 Project setup – Operator access

In the **Setup** area of the project, the operators who will be accessing the queue on which Portrait Foundation is creating call-backs must have the necessary access definitions. On the **Operator Access** tab for the operator(s) in question, in the **Search Queues** drop down, select the queue that was configured in the previous section. On the **Geo Options** tab, check the **View callback queues** and **Set callback**.

4.2.2 Portrait configuration

Create an EDGE logic definition (as described in section 3.2.2) for a logic with name CreateCall. Configure the following inputs for this logic:

Name	System name	Data type
Country Code	TEMP-CntryCode	String
Daylight Savings Time	TEMP-DST	String
Phone Number	TEMP-PhoneNum	String
Queue Name	TEMP-QueueName	String
Scheduled Date	TEMP-SchedDate	String
Scheduled Time	TEMP-SchedTime	String
Time Zone	TEMP-TimeZone	String
Use Customer Time Zone (Y or N)	TEMP-UseCustTZ	String

Note TEMP will need to be changed for a different file name if the TEMP file was not used in section 4.2.1.1.

Configure the following outputs under the **OK** outcome for this logic:

Name	System name	Data type
Error description	TEMP-ErrString	String
Error status	TEMP-ErrStatus	String

4.3 Integration considerations

The CreateCall logic provides a mechanism for Portrait Foundation to create call-backs in the EDGE environment. However, EDGE projects will differ in the way in which they interpret call-backs when they are retrieved from a queue. In particular, the way in which a call-back holds information about which customer it refers to will vary from project to project.

It will therefore be necessary to make minor modifications to the CreateCall logic on a per-project basis, in order to ensure that created call-backs contain the necessary customer identification information.

5 Portrait Displaying EDGE Guides

5.1 Capabilities

Portrait Foundation can display simple EDGE guides (including telephony functions) or subsets of guides in a browser window from within a Portrait Foundation business operation model using the 'EDGE Interaction' node. Portrait Foundation does not need to show its own user interface on the desktop for this feature to be available, although if Portrait Foundation is showing some user interface the EDGE screens will appear in a new browser window.

In this scenario, the EDGE server still controls the application screen-flow and provides screen definitions, business logic, CTI handling and links to external systems, but the screens are rendered through a Portrait Channel Enabler in an HTA window.

There are two implementations available for displaying EDGE guides through Portrait Foundation:

1. Single screen: A single screen will be visible and allow the user to navigate to other single EDGE screens as defined in the guide, similar to how Generated Interactions are displayed in Portrait Foundation. This option is suitable for simple guides that do not require multiple screens to be displayed simultaneously.

2. GEO emulation: This implementation partially replicates the EDGE GEO client and provides a multiple window environment.

Portrait Foundation can either display the EDGE project selection dialog (allowing DNIS information to automatically select the guide to be displayed) or launch a specified EDGE guide. The guide can be started 'from the beginning' or if a suitable entry point is available (that sets any state necessary to begin execution at this point in the application) a guide can be started 'part-way through'. In this latter scenario, the EDGE guide design must allow for a suitable method for the user to exit the guide, at which point the window displaying the EDGE guide disappears.

Portrait Foundation can be configured to launch the EDGE window in several ways:

- As a modal window: whatever the user is doing in EDGE must be completed before the user can do anything in the Portrait Foundation window.

- As a semi-modal window: the user can use the Portrait Foundation application window at the same time as the EDGE application window, but the Portrait Foundation application sidebar is disabled to prevent a user running further Portrait Foundation operations or leaving the current desktop.
- As a modeless-owned window: the user can do anything in either window, but if the user exits the Portrait Foundation application the EDGE window will disappear also.

It is the implementation engineer's responsibility to ensure that the launch method selected is suitable for the EDGE functionality in question. An optional 'X' close button is provided by default, though, if pressed, the user is asked to provide a result for the guide as EDGE guides cannot exit without a valid result. This 'X' button can be hidden by the implementation engineer or alternatively Portrait Foundation can provide a result and close the EDGE window programmatically. The most straightforward option is for each EDGE screen where it is possible to exit to have an exit button explicitly defined as part of the user interface.

When Portrait Foundation launches an EDGE window it has the ability to pass inputs to be used by the EDGE guide and similarly when the EDGE guide exits, it can pass data back to Portrait Foundation as output values.

5.1.1 Restrictions

Customisation of GEO client behaviour (controlled via settings in the **edgeo.ini** file) is not available.

When launching an EDGE window from Portrait Foundation, the size of the window will be determined by the EDGE screen definitions, not Portrait Foundation. The position of the window is set in the Portrait JavaScript; it is applicable to all EDGE windows launched from Portrait Foundation and cannot be modified from within the Configuration Suite.

When the Portrait Contact Centre application is being used to launch EDGE guides then the only supported screen resolution is 1024x768.

A guide can only be launched from a Portrait Foundation business operation model, not from a Custom Control. As stated in the capability section above, the EDGE guide appears in a separate window and hence is not suitable for displaying EDGE screens as part of a Portrait Foundation screen, for example on a tab.

Guides that are intended for use on ASCII-only terminals cannot be displayed using Portrait Foundation.

Guides with screen definitions containing of the following EDGE controls will not be successfully rendered: Audio, Graph, Image, OLE, Shape, Selection box,

Spinner and Video. Hence, it is recommended that guides shown via Portrait Foundation are stripped of these types of controls.

The following verbs that display user-interface are not supported: 3270-PLAY, CALL-EXTERNAL-FUNCTION for anything other than ActiveX controls, DATA-CONFERENCE, DDE, DISPLAY-CROSS-REFERENCE, EXECUTE, MENU and REFRESH.

The client-side variant of FLAT-FILE-ACCESS is not supported. The client-side variant of EXECUTE is supported, though the Minimize Geo = Yes and Create new instance = No settings are not supported.

All EDGE system screens (other than those designed to handle ASCII terminal clients) are supported with the exception of $CALLHIST, $QVIEW and $PREV. Menus and toolbars that form part of the GEO client are not supported when using Portrait Foundation to run EDGE guides.

EDGE styles and schemes are not fully supported; it is possible to customise the display of the guide to correspond to a particular set of EDGE styles, however, multiple schemes are not supported.

Whilst field validation logic executed when tabbing between fields on a screen will be supported, extensive use of this facility in a guide may result in performance degradation. Field help is not supported.

Rendering EDGE screens through the Portrait Channel Enabler places restrictions on the types of browser that can be used on client desktops. The browsers supported will be the same as those that are supported by the Portrait Contact Centre channel (that is Microsoft Internet Explorer only; see the latest Portrait Foundation release documentation for version information).

For these reasons it is recommended that this facility is only used over reasonably high bandwidth connections between client and server and, as such, would not be suitable for deploying an EDGE guide in a browser over the Internet.

All synchronous telephony events are supported, the only asynchronous event supported is the inbound call event. This means that although an EDGE guide running in a browser can initiate a call transfer or conference call, it cannot deal with incoming transfers or conferences (because the gateway does not support transfer cancel / complete and conference complete asynchronous events).

As stated in the capability section above, it is possible to launch EDGE in a modeless-owned window. However, doing this will expose the implementer to potential Portrait Foundation issues where a second operation can be launched and consequently there are two Portrait Foundation process models running simultaneously (the new operation plus the model that launched the EDGE guide). This can lead to situations where both models attempt to update the

same Portrait Global value, such as CurrentCustomerID, resulting in undefined behaviour.

Only one guide can be launched from a Portrait Foundation session at a time.

The following additional EDGE features are not supported: hot-keys, field cross-reference lookup and bulletins.

5.2 Using the EDGE Guide node

5.2.1 Overview

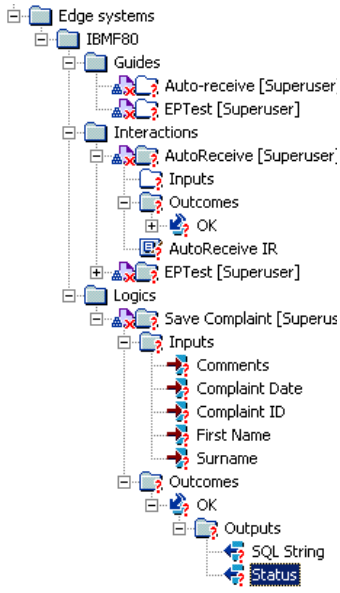
Making use of the EDGE Guide node comprises two separate configuration activities:

- Configure the EDGE guide within the Configuration Suite workspace.
- Drop the EDGE specific nodes required to launch the guide into a Portrait Foundation process model.

The following sections describe how to achieve these steps.

5.2.2 Configuring the guide

Figure 6 – EDGE system configuration



To set up a new guide definition within the Configuration Suite, it is necessary to:

- Create a new EDGE System (if one does not already exist) to represent the EDGE server with which Portrait Foundation is interacting. You will be prompted for a name to represent the EDGE system.
- Add an EDGE guide definition. This definition represents the EDGE Guide that will be started when Portrait Foundation interacts with EDGE. This definition needs to be present whether you using Portrait Foundation to run logic (using the EDGE Logic node) or launching EDGE interactions (using the EDGE Guide node). You will be asked to supply:
  - A name to represent the guide within Portrait Foundation.
  - The name of the guide on the EDGE server.
- Add the interaction that will be launched by the EDGE Guide node. You will asked to supply:
  - A name to represent the guide interaction within Portrait Foundation.
  - A description of the interaction.
  - The EDGE guide definition. You are presented with a list of the EDGE guide definitions that you have configured.

- Configure the inputs and outputs of the interaction. Configuring the interface to an interaction is identical to setting up the interface to a Portrait Foundation process model. However it is important that the system name specified for each input/output is identical to the EDGE field name that you wish Portrait Foundation to set/read when running the guide in EDGE. For example, if the EDGE guide reads an EDGE field called EPTest-EPComments, then you must specify EPTest-Comments as the system name of an input to the interaction you are configuring in Portrait Foundation. Note that you are only likely to have to specify inputs to an interaction when you wish to start a guide 'part way through'.
- Configure the Implementation Reference that Portrait Foundation should use to display the interaction.

5.2.3 Connecting to EDGE

Prior to launching an EDGE guide, it is necessary to log on to the EDGE server. See section 3.2.3 for details on how to do this.

If you are using the AIT_HRZ_EDGE_Integration package to launch an EDGE guide, you will need to modify the Portrait Foundation configuration to retrieve the name of the EDGE server to which you're trying to connect. To do this:

- Create a new data access model that outputs the EDGE server name. This model can be as simple as a start node, a data manipulation node that initialises a string to be alias of the server and an end node.
- Create an overlay to the **EDGE - Retrieve server name** data access class by right-clicking the class and selecting **Extend and override>Create overlay**.
- Add your new data access model as a data access class member to the **EDGE - Retrieve server name** data access class.
- Map the server name output by your data access model on to the **Server name** output on the **OK** outcome of the data access class.

5.2.4 Running an EDGE guide

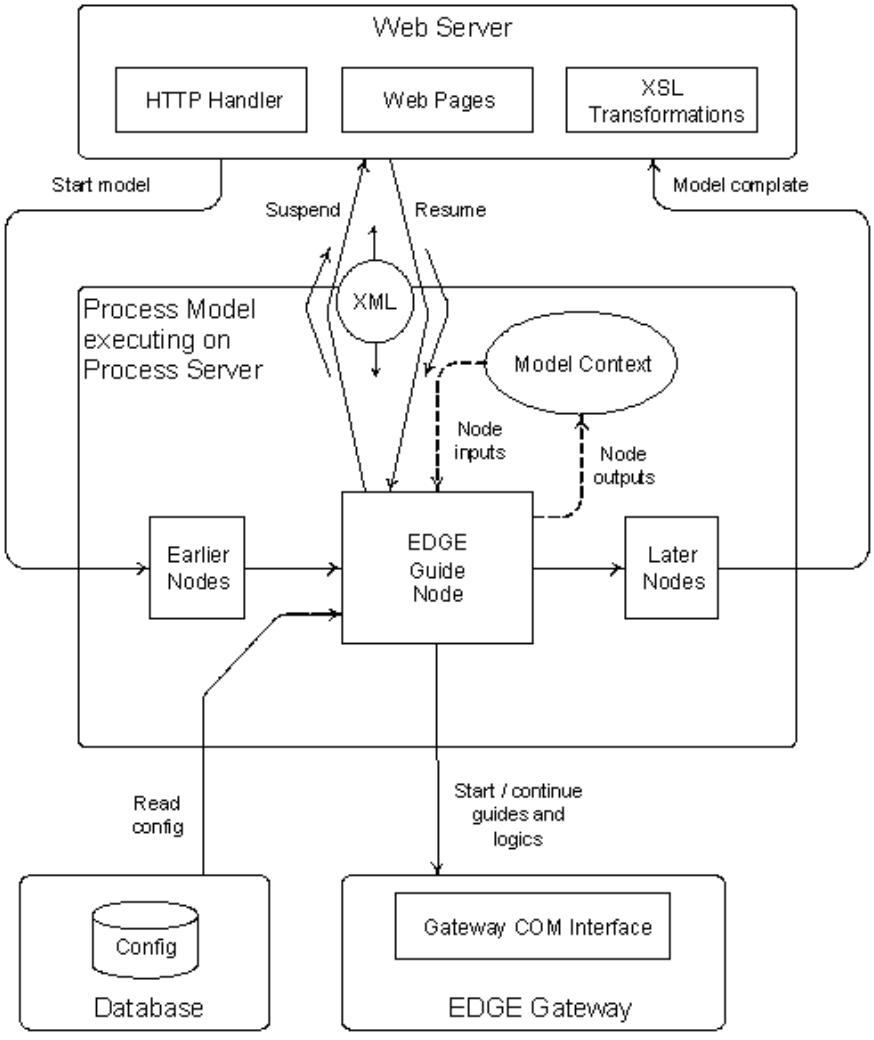
EDGE Guide node

This node can be configured to execute a named EDGE interaction. The node will then adopt the inputs and (outcomes and) outputs of the named interaction. This node is a Portrait Foundation client action node (as are the Client Interaction and Generated Interact nodes) so when it suspends (at a guide screen) it returns control to the client. The node is resumed in exactly the same manner as other

client action nodes and requires the response to the Gateway as its input parameter from the client.

Figure 7 shows the relationship between the EDGE Guide node and other components in the Process Engine environment, as well as the Web Server and EDGE Gateway.

Figure 7 – EDGE guide node placement with other Portrait components



This node is configured with:

- EDGE system.
- EDGE guide name.
- Start screen: the screen at which you want to guide execution.
- Inbound/Outbound indicator: whether you are running the guide in inbound or outbound mode.

If any of these are not configured for the node, they become inputs to the node.

The node has the following fixed BOOLEAN optional inputs:

- EDGEGuideDevelopmentMode: use the development guide instead of the operations guide. Please refer to the *EDGE Client Reference Manual* for details.
- EDGEGuideDebugMode: use the guide in test mode instead of live mode. In test mode no data is written to EDGE files. Please refer to the *EDGE Client Reference Manual* for details.

If a guide is configured then the node adopts the inputs and outcome (always **OK**) and outputs of the guide selected. If no guide is configured the node will not have any additional inputs or outputs; the guide will be selected at runtime by the input EDGEGuideName. In this scenario the Guide must be able to execute without inputs, and its outputs, if any, will not be available to the caller of the node.

EDGE Autoreceive node

This node provides the ability to call into EDGE and have EDGE prompt when a guide is to start execution. This supports a 'wait for call' style user interface. The name of the guide that will be started is not known by the caller because it is chosen by EDGE processing. As a result the Autoreceive node differs from the guide node in that it does not have any configuration of which guide to run and, therefore, has no inputs or outputs. Like the EDGE Guide node the Autoreceive node may suspend and resume any number of times until the guide is completed.

This node has one single input – EDGESessionId.

This node is configured with:

- Inbound/Outbound indicator (compulsory).

It has one fixed outcome 'OK' and produces no outputs.

EDGE Project List node

This node returns a list of the guides that are present in the system currently logged onto.

This node has one single input – EDGESessionId.

It has one output for the outcome **OK**: EDGEProjects. This is a data object collection which contains a list of all of the available guides in the project associated with this logon.

5.3 Customising the display of guides

Portrait Implementation References are used to control how guides are displayed on the browser. There are two standard Implementation References provided with the EDGE-Portrait gateway:

- **EDGESingleScreen.aspx**. This Implementation Reference should be used to show simple screen to screen guides that can be run directly from the Portrait Foundation desktop.
- **EDGEGeoEmulation.aspx**. This Implementation Reference should be used to provide an EDGE desktop (like the GEO client MDI application) that manages the windows that are being displayed and their modality.

If you wish to create new Implementation references, you should consider:

- Re-using the client script event handling and client server communication used by the supplied Implementation References.
- Deriving the new pages from the **EDGEGuideBase** base class. You will then be able to make use of the generic XML parsing and control building infrastructure.

5.3.1 Styles

Styles have been implemented to reflect those used by AdvantEDGE; however, custom styles can be added by editing the cascading stylesheet **Portrait Implementation\Portrait Application Centre\Includes\EDGEStyles.css**.

Guide screen specific styles are dynamically assigned during the XML parsing. The name of the style is assumed to be 'EDGEStyle_??', where ?? will be the style index represented in the xml received from EDGE.

Styles applied to the generic EDGE session CI Dialogs are also available in the same style sheet.

5.3.2 Running EDGE guides without any Portrait user interface

The EDGE-Portrait gateway release provides an HTA that can be used to run an EDGE guide in a browser without any Portrait Foundation user interface. This HTA should be used where you are trying to distribute an EDGE guide via the browser mechanism without providing any additional functionality from Portrait Foundation. See section 2.5.1 for details of how to install the web enabled EDGE client.

The HTA can be found at **Portrait Implementation\Portrait Application Centre\AIT_HRZ_EDGEClient.hta**. This HTA starts the Portrait Application Framework Model called **EdgeAFM** found in the **AIT_HRZ_EDGE_Integration** package.

The AIT_HRZ_EDGE_Integration package itself has a number of extension mechanisms. Two model classes, 'Edge – Standalone desktop pre-process' and

'Edge – Standalone desktop post-process' allow custom pre and post processing models to be run. The 'Edge – Standalone desktop' CI class allows a modified version of the desktop to be used, e.g. with an extra menu or toolbar. In order to 'select' which model/CI class member to run, the **StartModel.amc** query string in the HTA must be modified to set the lookup criteria exposed as EdgeAFM inputs.

The behaviour of the Login, Projects and CallCentre dialogs can be altered by modifying the **SetClientVariables.amc** query string used in the HTA to include a **'_ClientSettings'** attribute. Values that can be encoded on this attribute are:

Value	Screen	Description
SkipLoginAtStartup:true	Login	Don't show the Login screen (if there is a session already established)
QuitAppOnExit:true	Login	On exit from the CallCentre screen, quit the application.
AutoReceiveCallAfterGuide:<n>	CallCentre	On completion of a Guide, automatically click the 'Receive call' button after <n> seconds if there is no other button clicked.
AutoExitAfterGuide:true	CallCentre	On completion of a Guide, automatically click the 'Exit' button.

For example:
'**_ClientSettings**=SkipLoginAtStartup:true,AutoReceiveCallAfterGuide:10'

5.4 Integration considerations

Style of guide execution

When running guides through Portrait Foundation, care must be taken in how the guide is executed. A simple screen to screen guide can be run directly from the Portrait Foundation desktop. To run a more complex guide, it is highly recommended that the guide is launched via the EDGESession Custom Interaction. This provides an EDGE desktop that manages the windows that are being displayed and their modality. All screens are modeless children of the EDGE desktop and command windows/message boxes are, where possible, launched as modal children of the window that triggered them.

Note that it is not possible to run a specific guide in stand-alone and session modes, they are restricted by Implementation Reference configuration.

Distribution of guides via browser

Certain EDGE guides, for example AdvantEDGE, rely heavily on ActiveX controls and images. This collateral will need to be downloaded to the browser to allow correct functioning of the guide. The Portrait Foundation application that is launching the guide will need to trigger the download of these items.

To support secure download, AIT provides a signed version of an installation script for these items. This is installed by the Portrait Foundation implementation install to **Inetpub\wwwroot\Portrait_Client\downloads** on the Portrait web server.

Running guides over low bandwidth connections

If you need to run EDGE guides over a low bandwidth connection, you should design the guide to minimise network traffic by considering:

- Events are either sent immediately to the server or queued. When the control event being handled has a logic associated, the event will be sent immediately. When there is no associated logic the event will be queued. The event queue will be cleared every time a client-server communication takes place.
- Certain control events will always be sent immediately, i.e. button click.
- Client-side event firing could be modified to limit the event types being handled. This will be completely dependant on how a guide has been built.

web.config settings

The following settings must be present in the web.config file of the web application.

To avoid errors when submitting data to the Portrait request handler you will need to add requestValidationMode="2.0" to the <httpRuntime> node in the <system.web> section:

```
<system.web>
  ...
  <httpRuntime requestValidationMode="2.0" />
  ...
</system.web>
```

To avoid errors when using a tab control in a guide screen you need to add controlRenderingCompatibiltyVersion="3.5" to the <pages> node in the <system.web> section:

```
<system.web>
```

```
  ...
  <pages controlRenderingCompatibilityVersion="3.5" />
  ...
</system.web>
```

The following may be defined in the <appSettings> section of the web.config file.

<add key="ROWCOL_X" value="7"/> <add key="ROWCOL_Y" value="16"/>	Should match the column width and height in pixels as defined in your EDGE Screen Defaults.
<add key="autoSelectSingleProject" value="0"/> <add key="autoSelectSingleProject" value="1"/> <add key="autoSelectSingleProject" value="2"/>	Matches the equivalent edgeo.ini parameter. Displays a list of projects. If there is only one project, opens it automatically. If there is only one project, opens it and clicks the 'Receive call' button automatically.
<add key="autoTab" value="N"/>	*Matches the equivalent edgeo.ini parameter.* Determines whether an automatic ENTER is generated when the last character of an object has been typed.
<add key="execLogicOnComboSelChange" value="N"/>	*Matches the equivalent edgeo.ini parameter.* Determines whether logic is executed on selection change in a combo or drop down box.
<add key="forceLogicOnTab" value="N"/>	*Matches the equivalent edgeo.ini parameter.* Determines whether the TAB key should be treated the same as the ENTER key for triggering logic.

<add key="maxLoadSize" value="-1"/>	A setting to optimise loading of large lists: define the maximum number of list entries to be loaded at a time, or -1 for unlimited.
<add key="serverIdentifier" value="[MachineName]"/>	Used to display a server identifier on the EDGE status bar. If the special value "[MachineName]" is specified, then the machine name is displayed, otherwise the quoted value is displayed.
<add key="pathSubstitutionCount" value="0"/> <add key="pathSubstitution1" value="C:\GeoAddOn\BMP|http://server/Portrait_MyPortrait_ApplicationCentre/images/"/> …	Defines how many *pathSubstitutionN* entries follow. Each pathSubstitution entry (1..*pathSubstitutionCount*) defines a rule for substituting a server URL in place of part of a CommandButton icon local image path. This eliminates the need to have image files on each client PC. They are case-insensitive and applied in sequence. In this example any occurrence of "C:\GeoAddOn\BMP\" will be replaced by "http://server/Portrait_MyPortrait_ApplicationCentre/images/".
<add key="UseWaitWindow" value="False"/>	Defines whether to display a wait window for long transactions.
<add key="WaitWindowDelay" value="5000"/>	Defines after how many milliseconds the wait window should be displayed.

6 EDGE Invoking Portrait Process Models

6.1 Capabilities

EDGE can invoke Portrait Foundation process models that have been exposed as a web service and do not contain any Portrait Foundation user interface using the 'PORTRAIT' verb.

This allows EDGE to read or set data available to Portrait Foundation and run business logic contained in Portrait Foundation process models.

The call to the Portrait Foundation process model is synchronous and EDGE can pass input parameters to the process model. When the process model completes, any outputs and the outcome will be passed back to EDGE. Data object collections are not supported by the PORTRAIT verb.

Some example web services are pre-configured to access existing Portrait Foundation process models.

6.1.1 Restrictions

As stated above, data object collections are not supported by the PORTRAIT verb.

Whilst Portrait Foundation allows any process model to be exposed via a web service, the implementation engineer must ensure that process models called by the PORTRAIT verb in EDGE do not contain any nodes that require a model to resume, for example, client action nodes such as Generated Interaction or Custom Interaction nodes.

As is normal when configuring Portrait Foundation process models, nodes such as 'Create task' should not be placed in models that will run synchronously such as those invoked by the PORTRAIT verb, instead, they should be placed in a Workflow Process Model that is configured as a sub-model. Workflow Process Models (or WPMs) run asynchronously and consequently can contain nodes such as 'Create task' that eventually requires the model to resume. No user interface is permitted in WPMs.

It should be noted that because of the method used to create tasks in asynchronous Workflow Process Models, whilst a model can be configured to run a WPM that creates a task, the task ID of the new task will not be subsequently available to the calling EDGE guide because of the asynchronous nature of WPMs.

This is not a restriction of the gateway, simply an artefact of how Portrait Foundation creates workflow tasks.

It is possible for Portrait Foundation to launch an EDGE guide in a browser which subsequently uses the PORTRAIT verb to run a Portrait Foundation process model. However this model runs in a separate Portrait Foundation session to the one that launched the EDGE guide and hence cannot interact with the Portrait Foundation desktop displayed.

6.2 Using the PORTRAIT verb

Please refer to the EDGE documentation set for instruction on how to:

- Configure Portrait Foundation processes in EDGE
- Launch a Portrait Foundation process using the PORTRAIT verb.

6.3 Integration considerations

⇒ EPL indirectly makes use of v1.9.1 of the iconv library, which is licensed under the Lesser General Public License (see http://www.gnu.org/licenses/lgpl.html for details). All EPL's usage of iconv is via v2.6.9 of the libxml2 library.

The PORTRAIT verb makes use of a library called EPL (**E**DGE to **P**ortrait **L**ibrary) to communicate with Portrait Foundation web services. A command line test utility called EPLTest is provided with the EDGE-Portrait gateway that can drive the EPL. This utility is useful in debugging EDGE invoking Portrait Foundation process models as it can be used to isolate whether a problem lies on the EDGE server, or within the Portrait servers.

For help on how to use EPLTest, run **EPLTest -h** on the command line on the EDGE server.

7 EDGE Launching Portrait Windows

7.1 Capabilities

EDGE can invoke Portrait Foundation process models (including Application Framework Models), that display user interface by placing an ActiveX control on an EDGE screen displayed in GEO. The ActiveX control uses the Internet Explorer browser control to show the user interface presented by the Portrait Foundation process. EDGE gives the ActiveX control the URL of a Portrait Foundation web page, which can then start a Business Operation Model, an operation or display Custom Controls as normal.

The Portrait Foundation browser window is embedded in the EDGE screen on which the ActiveX control has been placed. EDGE can launch the EDGE window with the ActiveX control showing Portrait Foundation content in the usual ways, that is:

- as a modal window: whatever the user is doing in Portrait Foundation must be completed before the user can do anything in GEO (other than access the GEO client toolbar and menus).
- as a modeless-owned window: the user can do anything in either windows but if the user exits the EDGE application the EDGE window displaying Portrait Foundation content will disappear also.

It is the implementation engineer's responsibility to ensure that the EDGE guide shows the screen in a way that is appropriate to the type of Portrait Foundation functionality launched.

Once the window with Portrait Foundation content is launched, EDGE can subsequently close the window automatically or alternatively, suitable exit points can be built into the Portrait Foundation application or process being displayed. Alternatively, the EDGE developer can choose to implement a standard 'X' close button on the window. The method chosen will usually depend on whether a modal or modeless window has been launched.

When EDGE launches a window displaying Portrait Foundation content it has the ability to pass inputs to be used by Portrait Foundation, as well as read outputs when the Portrait Foundation process finishes.

Multiple windows can be launched from an EDGE guide.

7.1.1 Restrictions

To expose a Portrait Foundation process in EDGE in this manner, it is necessary to undertake the standard Portrait Foundation configuration and web development activities required to expose a business process in a new channel. This means that implementation references need to be written to ensure that the user interface is presented correctly. In addition, it may be necessary to:

- Configure PONDs to control screen navigation after an operation completes.
- Build .NET controls suitable for use in the channel.

Pre-existing implementation references that have been written for the web channel are suitable for use with Portrait Foundation processes that are shown via the MS Browser ActiveX control.

Note that the user interface for Portrait Foundation processes that currently exist within the Contact Centre are unlikely to be presented correctly without modifying the implementation references used to be consistent with running in a browser control.

When launching a Portrait Foundation browser window from EDGE, the size of the window will be determined by the size of the ActiveX control, when placed on the EDGE screen. The size of the Portrait Foundation window that is presented within the window will be determined by the implementation references used by the Portrait Foundation process being run.

The inputs EDGE can pass to Portrait Foundation are restricted to those that can be passed via a query string in a URL. Consequently data objects cannot be passed in this manner and as a result, BOMs may need to be wrapped or re-written to 'flatten' data object structures before they can be called by EDGE.

It is only possible to launch Portrait Foundation browser windows from EDGE guides running in GEO, not from an EDGE guide running in a browser window rendered through Portrait Foundation. This capability may be developed for a future release of the gateway.

7.2 Using the Portrait browser ActiveX control

The Portrait Foundation browser ActiveX control is used to launch Portrait Foundation models and operations from EDGE screens. When the model/operation has completed, an event is fired after which the control can be queried for the outcomes and outputs of the model/operation.

Note that to make use of this ActiveX control on an EDGE screen, the control needs to be defined within the EDGE guide in the normal way, see the *EDGE ETW-W Reference Manual* for details. The control is called **PortraitWebBrowserU.dll** and can be found in the **Common\Bin** folder beneath the Portrait installation directory.

7.2.1 Control methods

The control exposes methods that are used to set up and launch a Portrait Foundation model/operation. These are described in the following table:

Table 5 – Portrait browser ActiveX control methods

Method name	Description
AddData	Use this method to add name/value pairs that represent the inputs to the Portrait model/operation.
ClearData	Use this method to clear all name/value pairs that have been added using the AddData method.
Launch	Once all the inputs to the model/operation have been set up, use this method to launch the Portrait model/operation.
Navigate	Use this method to navigate to a given URL. Parameters can be supplied using the AddData method.

7.2.1.1 AddData

The parameters to AddData are described in the following table:

Table 6 – AddData parameters

Name	Direction	Type	Description
bstrName	in	BSTR	The name part of the name/value pair to add
bstrValue	in	BSTR	The value part of the name/value pair to add

7.2.1.2 ClearData

This method has no parameters.

7.2.1.3 Launch

The parameters to the Launch method are described in the following table:

Table 7 – Launch parameters

Name	Direction	Type	Description
bstrOperation	in	BSTR	Indicates whether a Portrait model or operation is to be launched. If this parameter has the string value "0" then a Portrait **model** will be launched. If the parameter has the string value "1" then a Portrait **operation** will be launched
bstrWebServer	In	BSTR	Specifies the name of the Portrait web server that the model/operation will be run on
bstrDirectory	In	BSTR	Specifies the virtual directory of the Portrait web application

7.2.1.4 Navigate

The parameters to the Navigate method are described in the following table:

Table 8 – Navigate parameters

Name	Direction	Type	Description
bstrURL	In	BSTR	The URL to navigate to

7.2.2 Control events

The browser control fires the events described in the following table:

Table 9 – Browser control events

Event	Description
OnOperationComplete	Once the Portrait model/operation that has been launched has run to completion, or an error has occurred, then this event will be fired

7.2.2.1 OnOperationComplete

The parameters supplied when this method is invoked are described in the following table:

Table 10 – OnOperationComplete parameters

Parameter	Type	Description
bError	boolean	If the launched Portrait model/operation completed successfully then this parameter will be zero, otherwise the parameter is non-zero

7.2.3 Control properties

Once the OnOperationComplete event has been fired there are several properties that can be retrieved from the browser control. These are described in the following table:

Table 11 – Control properties

Name	Type	Description
Outcome	BSTR*	The outcomes of the successful Portrait model/operation
Outputs	BSTR*	The outputs of the successful Portrait model/operation
Error	BSTR*	If the Portrait model/operation failed then this property returns the error information

Other properties available are described in the following table:

Table 12 – Control properties

Name	Type	Description
WebBrowserControl	Iunknown**	A pointer to the Microsoft Web Browser ActiveX control that is contained by the Portrait browser control

7.2.4 Launching Portrait models

When launching Portrait Foundation models the following name/value pairs are mandatory and must be added using the AddData method:

Table 13 – Mandatory model name/value pairs

Name	Value
_MODEL	The name of the model to launch
_REQUEST_TYPE	_START

It should be noted that the above mentioned name/value pairs are case sensitive.

After the mandatory name/value pairs have been added, the name/value pairs that represent the inputs to the model should be added.

7.2.5 Launching Portrait operations

When launching Portrait Foundation operations the following name/value pairs are mandatory and must be added using the AddData method:

Table 14 – Mandatory operation name/value pairs

Name	Value
operation	The name of the operation to launch

It should be noted that the above mentioned name/value pairs are case sensitive.

When Portrait Foundation operations are launched the inputs to that operation are taken from the Portrait Globals data context. This means that when calling AddData to add the name/value operation input pairs, the name should be prefixed with "_GLOBAL.". The inputs will then be written to the Portrait Globals data context where the operation will expect to find them.

7.2.6 Passing data objects as inputs to Portrait models/operations

Data objects can be passed as inputs to models/operations as name/value pairs using the AddData method as with other inputs. An example of passing a data object is (simplified for demonstration purposes):

AddData("dataObject1._Category", "General");
AddData("dataObject1._Type", "TabVariable");
AddData("dataObject1._Version", "0");
AddData("dataObject1.Property1", "property value");

7.2.7 Considerations when using the browser control

A sample web application named EDGEToPortrait.NET is supplied in the Portrait Foundation SDK. This application has all the elements required to support the browser control. To install this web application you will need to make the relevant selection from the User Custom install of the Portrait Implementation install.

The following sections outline the key features of the EDGEToPortrait.NET application.

7.2.7.1 EDGERequestHandler.cs

This class is responsible for receiving requests from the browser control and starting the requested model/operation. Resuming a model/operation after a custom interaction is also handled within the class.

When a model/operation completes, the EDGE request handler returns a form with an id of 'EndOfOperation' which the browser control will detect and fire the OnOperationComplete event, after which the model outputs/outcomes are available.

7.2.7.2 web.config settings

The following setting must be present in the web.config file of the web application. It will direct all start model/operation calls made by the browser control to the EDGE request handler.

```
<system.web>
  <httpHandlers>
    <add verb="*" path="*.amc" type="AIT.Portrait.Web.AspNetRequestHandler,
AIT.Portrait.Web" />
  </httpHandlers>
</system.web>
```

7.3 Integration considerations

When exposing a Portrait Foundation business process via EDGE, you will need to consider the standard Portrait Foundation configuration tasks associated with implementing a business process within a new channel, namely:

- Write suitable Implementation References for user interface elements.
- Potentially configure PONDs to control navigation.

- Potentially write new .NET controls required to present user interface elements.