



# Deploying EGL CE programs



---

## Contents

### Deployment of EGL CE applications . . . 1

Deploying an EGL application using the EGL  
deployment descriptor . . . . . 2

Adding Web service deployment information to  
the deployment descriptor. . . . . 5

Creating and using a shared protocol . . . . . 7

### Index . . . . . 9



---

## Deployment of EGL CE applications

After developing EGL applications in the workbench, deploy them to the environment where they are used.

You develop EGL applications in one or more EGL projects, then deploy them to dynamic Web projects. Although the deployment project is in the workbench, you configure the project to match the characteristics of the environment where you plan to deploy your application. If you need to run the application in different environments, you can deploy a single EGL project to multiple dynamic Web projects. For example, you might create an application to run on two different local area networks, each of which uses a different version of Tomcat. You can create a dynamic Web project for each network, and deploy your application to both projects.

An additional benefit of deployment is that you can elect to remove all EGL source code from your deployed application. Once you have deployed your application, you can package the deployed files (for example, in an archive file) and ship them to customers.

During the development process, EGL automatically generates code (JavaScript from Rich UI programs, and Java from services) each time you save a source file. When you deploy that code, EGL performs the following additional actions:

- Packages the .js JavaScript files into HTML files
- Adds server specific code to project files

### Services

If your code accesses any services, the declaration of the Service or Interface part for that service must include one of the following complex properties:

#### **@RESTBinding**

Binds a service reference to a REST service at generation time.

#### **@WebBinding**

Binds a service reference to a SOAP service at generation time.

#### **@bindService**

Binds a service reference to a REST, SOAP, or EGL service at runtime, using the information in the EGL deployment descriptor specified by the **bindingKey** property field.

### Related concepts



#### Overview of EGL

Enterprise Generation Language (EGL) is a technology to help you write professional software quickly. EGL is effective because it shields you from the details of other software technologies. Even developers who are new to those technologies can create Web-based logic, as is supported in Rational® EGL Community Edition, and other kinds of logic, as is additionally supported in Rational Business Developer, Rational Business Developer for System z®, and Rational Business Developer for i for SOA Construction.

### Related tasks

“Deploying an EGL application using the EGL deployment descriptor”  
The EGL deployment descriptor controls the details of deployment.

**Related reference**

---

## Deploying an EGL application using the EGL deployment descriptor

The EGL deployment descriptor controls the details of deployment.

The actual deployment process is virtually automatic. To deploy an EGL CE application:

1. Locate the EGL deployment descriptor for one of the application projects. The EGL deployment descriptor is located in the `EGLSource` folder and has the file extension `.egldd`. By default, the base name is the same as that of the current project.
2. Right click the project name in the workbench and click **Deploy**.

EGL then deploys the application automatically, based on information in the deployment descriptor.

Use the EGL deployment descriptor editor to make changes to the deployment descriptor. To launch the editor, double click the deployment descriptor file.

The following sections describe the information you need for each page of the EGL deployment descriptor.

### Overview

The overview page combines the information from the various pages of the file, and provides links to take you to each page that you might need to change.

### Service client bindings

Service client bindings give the client of a service the information it needs to access that service. You can either provide that information here, or in a complex property associated with the declaration of the Service or Interface part. The **@RESTBinding** complex property contains binding information for REST services, and the **@WebBinding** complex property contains binding information for SOAP services. Both properties bind the associated service at generation time.

If the Service or Interface part includes the **@bindService** complex property instead of **@RESTBinding** or **@WebBinding**, EGL uses the deployment descriptor for binding information at run time.

If you use the deployment descriptor for service client binding, the information you need depends on the type of service you access:

#### SOAP Web Binding

A simple object access protocol (SOAP) service typically uses HTTP for transmission and XML for its message format. A SOAP service always uses a Web Services Description Language (WSDL) file as an interface to the client. You need the following information to access a SOAP service:

- The name of a WSDL file in your workspace.
- An EGL Interface part to use with the service.

- A Web Binding Name, which must match the value of the **bindingKey** property field of the **@bindService** complex property for the associated service.
- A WSDL URI, if you want to override the WSDL file name (for example, if you need a different version of the service for production or testing).

### REST Web Binding

For the purposes of EGL CE, a representational state transfer (REST) service provides a single unit of business data for each request. Requests specify a uniform resource identifier (URI), typically a base URL for a Web site, followed by qualifiers for the requested data. For example, the URI `www.example.com/employee/123` requests data for an employee with the number 123. REST services do not use WSDL files. You need the following information to access a REST service:

- A REST Binding Name, which must match the value of the **bindingKey** property field of the **@bindService** complex property for the associated service.
- A Base URL that qualifiers will be added to.
- A Session Cookie ID that allows you to keep a session open with the host for the service. The default value is `JSESSIONID`, which is always the session ID when your application runs on Apache Tomcat.

### EGL Binding

An EGL service is based on an EGL Service part. The service may be local to your server or accessible through one of several standard protocols. You need the following information to access an EGL service:

- An EGL Binding Name, which must match the value of the **bindingKey** property field of the **@bindService** complex property for the associated service.
- A Service Name, which is the name of the EGL Service part.
- An alias, if the Service part uses one.
- A protocol type. If the EGL service is not local, you can choose one of the listed protocols, or click **Choose from protocols** to display a list of available options.

### Native binding

This binding provides access to a System i service program. You need the following information to access a System i service:

- A Native Binding Name, which must match the value of the **bindingKey** property field of the **@bindService** complex property for the associated service.
- A protocol type. You can choose one of the listed protocols, or click **Choose from protocols** to display a list of available options.

## Web Service Deployment

After you have coded an EGL Service part that you want to expose as a Web service, you must add information to the deployment descriptor that tells EGL to generate the necessary Web service wrapper. For more information, see “Adding Web service deployment information to the deployment descriptor” on page 5

## EGL Rich UI Deployment

An EGL application can consist of multiple Rich UI Handlers, although the Handlers must all be in the same project. On this page, you can select the various

Rich UI handlers that make up an application so that EGL can deploy them together. In the EGL Rich UI Application table, click **Add** to create an application. The editor automatically assigns the name `new_application`, followed by a number if you have more than one in the table. For each application, provide the following information:

#### EGL Rich UI Handlers

The editor displays all available Rich UI Handlers in the project. Select the check boxes for the Handlers you want to include in the application.

#### Locale Settings

The editor displays all locales that you have configured. Select the check boxes for the locales to which you wish to deploy the application. Click **Configure** to go to the Rich UI Preferences page and add or remove locales.

#### Deployment Targets

The editor displays all dynamic Web projects that you have configured on the **Deployment Targets** page of the EGL deployment descriptor. Select the check boxes for the target projects you want to deploy to. Click **Configure** to go to the **Deployment Targets** page and add or remove Web projects.

#### Additional Artifacts

The editor displays all artifacts for all projects. Note that the following projects are included:

- The source project (containing the deployment descriptor)
- The base Rich UI project (`com.ibm.egl.rui`)
- The Dojo widgets project (`com.ibm.egl.rui.dojo`)

All artifacts are automatically selected. Clear the check boxes for any artifacts you do not want included in the deployment.

## Protocols

Shared protocols are reusable definitions of a service connection that you can apply to one or more of the entries in an EGL deployment descriptor file. For more information, see “Creating and using a shared protocol” on page 7.

## Deployment Targets

Deployment targets are typically dynamic Web projects. If you do not use any services in your application, you can also deploy your project to a File System Directory (folder).

To add a deployment target to the table, click **New**.

In the Add Deployment Targets window, you can take the following actions:

- Move existing Web projects in and out of the list of deployment targets.
- Create a new project in your workspace. To do this, you need a name for the project and a target runtime, such as Apache Tomcat v6.0.
- Add a file system directory to the list of deployment targets.

The deployment targets you configure on this page are available on the **EGL Rich UI Deployment** page.



## Imports

You can import the information from another deployment descriptor, adding the service bindings, sharable protocols, and other information to your current deployment descriptor. Click **Add** and choose the deployment descriptor from the navigator panel.

### Related concepts



#### Overview of EGL

Enterprise Generation Language (EGL) is a technology to help you write professional software quickly. EGL is effective because it shields you from the details of other software technologies. Even developers who are new to those technologies can create Web-based logic, as is supported in Rational EGL Community Edition, and other kinds of logic, as is additionally supported in Rational Business Developer, Rational Business Developer for System z, and Rational Business Developer for i for SOA Construction.

### Related tasks

“Adding Web service deployment information to the deployment descriptor”  
After you have coded a service part that you want to expose as a Web service, you must add information to the deployment descriptor that tells EGL to generate the necessary Web service wrapper.

“Creating and using a shared protocol” on page 7

*Shared protocols* are reusable definitions of a service connection that you can apply to one or more of the entries in an EGL deployment descriptor file.

### Related reference

## Adding Web service deployment information to the deployment descriptor

After you have coded a service part that you want to expose as a Web service, you must add information to the deployment descriptor that tells EGL to generate the necessary Web service wrapper.

### Prerequisites

- An EGL project
- A Service part
- An EGL deployment descriptor file

### Adding the deployment information

1. Open the EGL deployment descriptor file and go to the **Service Deployment** tab.
2. Click **Add**.
3. In the **Add Web Services** window, select the service parts to be exposed as a Web service by moving them from the **EGL service parts found** list to the **EGL service parts to be generated as Web services** list.
4. Click **Finish**. Each service part that you chose is listed on the **Services Deployment** page.
5. Select a service part in the list on the left:
  - In the **Generate** column, indicate whether you want EGL to generate the Web service each time you generate the deployment descriptor; and if so, whether you want the service generated as a SOAP (Web) service, a REST

(Web) service, or both. Avoid the overhead of generating the Web service if you are not changing the service part when you generate the deployment descriptor.

- To access the service logic, double-click an entry in the **Implementation** column; or highlight the service-binding name and click **Open**.

6. Additional information varies by service type:

- For a SOAP (Web) service, indicate whether you want to identify the service characteristics with an existing WSDL file. If you check **Use Existing WSDL file**, you can specify the WSDL file and, within that file, the WSDL service and port elements. If you intend to create the WSDL file later, specify a value in the **Style** field; select **document-wrapped** unless the requesters need **rpc**. If you intend to run the SOAP (Web) service on CICS®, you also may need to specify the access details under **Platform-Specific Properties**; specifically, the access protocol, as well as the URI used to access the service.

In the CICS URI field, assign the low-level qualifier for the address used to access the SOAP service. The full address is as follows:

`http://domain:portNumber/URI`

*domain*

The domain name; for example, `www.example.com`.

*portNumber*

The number of the server-machine port that receives the request.

*URI*

The qualifier you are specifying. By default, the value is as follows, where *serviceName* is the name of the Service part:

`services/serviceName`

- For a REST (Web) service, you can select or clear the **Stateful** checkbox to indicate whether the service is providing access to a stateful host program on IBM® i. The issue is explained in *Accessing IBM i programs as Web services*.

Also, in the **URI** field, assign the low-level qualifier for the address used to access the REST service. The full address is as follows:

`http://domain:portNumber/contextRoot/restservices/URI`

*domain*

The domain name; for example, `www.example.com`.

*portNumber*

The number of the server-machine port that receives the request.

*contextRoot*

A setting in the Web project. The default is the name of the Web project.

In relation to WebSphere® Application Server, the value is in the JEE EAR deployment descriptor (`application.xml`).

*URI*

The qualifier you are specifying.

7. Save the deployment descriptor, which in most cases causes an automatic generation of output from that file.

**Related concepts**



`topics/pegl_serv_elements_cpt.dita`



`Accessing IBM i programs as Web services: overview`



`topics/pegl_serv_overview.dita`

### Related tasks



topics/pegl\_serv\_exposing\_tsk.dita

## Creating and using a shared protocol

*Shared protocols* are reusable definitions of a service connection that you can apply to one or more of the entries in an EGL deployment descriptor file.

To create and use a shared protocol:

1. Open the EGL deployment descriptor file for your project. If your project does not have a deployment descriptor file, see topics/pegl\_serv\_create\_deployment\_descriptor\_tsk.dita.
2. In the deployment descriptor editor, go to the **Protocols** tab.
3. Under **Sharable Protocols**, click **Add**. The Add Protocol window opens.
4. In the **Protocol Name**, enter a mnemonic for the new protocol.
5. Under **Choose protocol type**, select a type of protocol. Use **Local** for services in the same project.
6. Under **Attributes**, set the options for the protocol. These options differ for each type of protocol. See Deployment descriptor options for service clients.
7. Click **Finish**. The new protocol is listed under **Sharable Protocols**, and you can use that protocol when you create a new service client binding or Web service binding.

### Related tasks



topics/pegl\_serv\_using\_service.dita



topics/pegl\_serv\_calling\_local\_tsk.dita



topics/pegl\_serv\_add\_client\_binding\_wsdl\_tsk.dita



---

## Index

### Special characters

1, 2

### D

deployment descriptors  
  adding information 5

deployment descriptors (*continued*)  
  shared protocols 7

### S

services  
  shared protocols 7

shared protocols 7

### W

Web services  
  deployment information 5  
  shared protocols 7