K2 Solutions Deployment Guide

STEP BY STEP GUIDE

November 16

CORPORATE HEADQUARTERS
2615 151st Place NE
Redmond
Washington
98052
USA
PH +1 (425) 883 4200
FAX +1 (425) 671 0411

EMEA HEADQUARTERS
26 Worple Road
Wimbledon
London
UK
PH +1 44 (0) 845 612 0912
FAX +1 44 (0) 845 612 0911

APAC HEADQUARTERS
9 Shenton Way #06-02
Singapore
068813
PH +1 65 6327 4110
FAX +1 65 6327 4120

[WWW.K2.COM]

CONTENTS

INTRODUCTION

K2 blackpearl provides the option to design and test workflow solutions in a development environment allowing the developer to change or add functionality without affecting the day to day business flow. After testing these solutions, they can be deployed from the development environment to a UAT, staging or production environment to proceed with these new solutions in practice.

This document explains how to achieve the successful deployment from a development environment to production. A lot of preparation will have to be considered as well as understanding these environments, the tools used in the deployment and finally the process of moving to the production environment. Each of these will be explained in this document.

DEPLOYMENT CHECKLIST

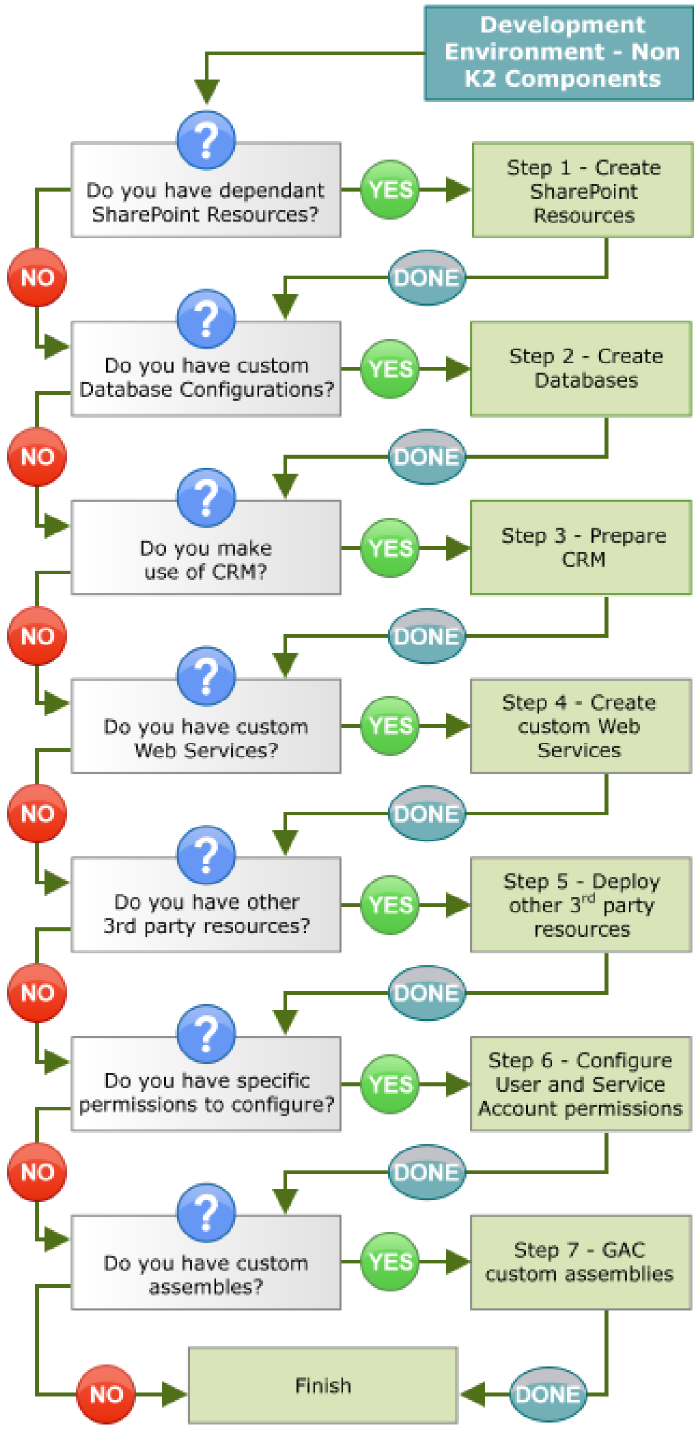
Use the checklist below to ensure that all the steps have been executed in the correct order for a successful deployment between environments. Each of these steps are discussed in more detail later in the document.

Step	Task	Complete
Check and prepare the Development environment	Verify that all tests are complete and have passed	
	Create MSBuild package for SmartObjects	
	Create MSBuild package for Workflows	
Prepare the Production environment (Manual process)	Make a backup of the Production environment	
	Verify that the Production environment is running the same K2 version as development environment	
Non K2 Components	Create SharePoint Resources	
	Create Databases	
	Prepare CRM	
	Create Custom Web Services	
	Deploy other 3rd party resources	
	Add User and Service Accounts permissions	
	Manage GACed Assemblies	
K2 Components	Manage K2 Product Assemblies	

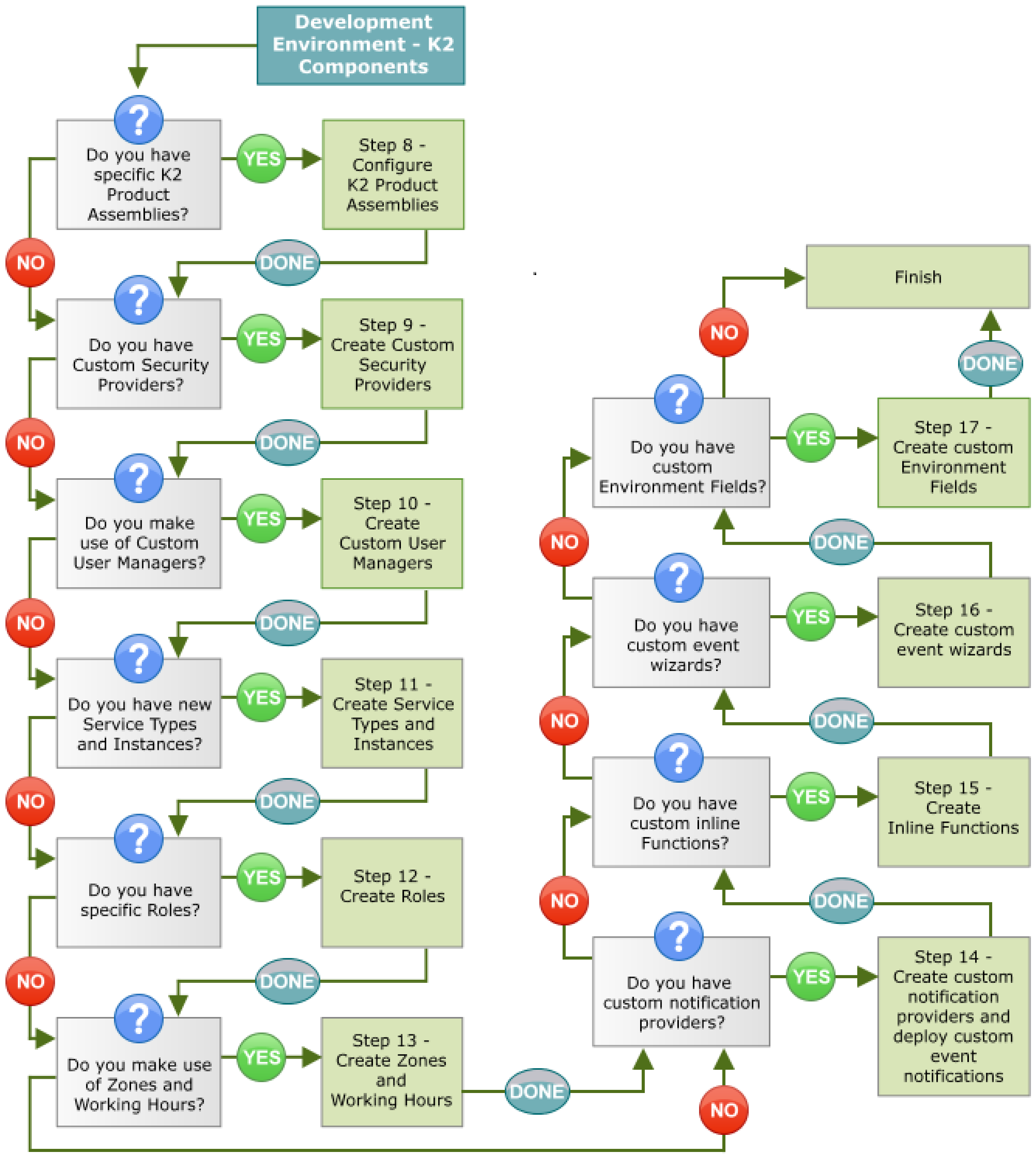
	Create Custom Security Providers	
	Register Custom User Managers	
	Register Service Types and New Service Instances	
	Create required Roles	
	Create required Zones/Working Hours	
	Register Custom Notification Providers and deploy Custom Event Notifications	
	Deploy Custom Inline Function Assemblies	
	Deploy Custom Event Wizards	
	Create Custom Environment Fields	
Deployment	Deploy the MSBuild package for SmartObjects	
	Deploy K2 Connect Service Instances	
	Deploy the MSBuild package for Workflows	
	Deploy User Forms	
	Assign K2 Security	
Test Production	Test Production environment	

DEPLOYMENT FLOWCHART

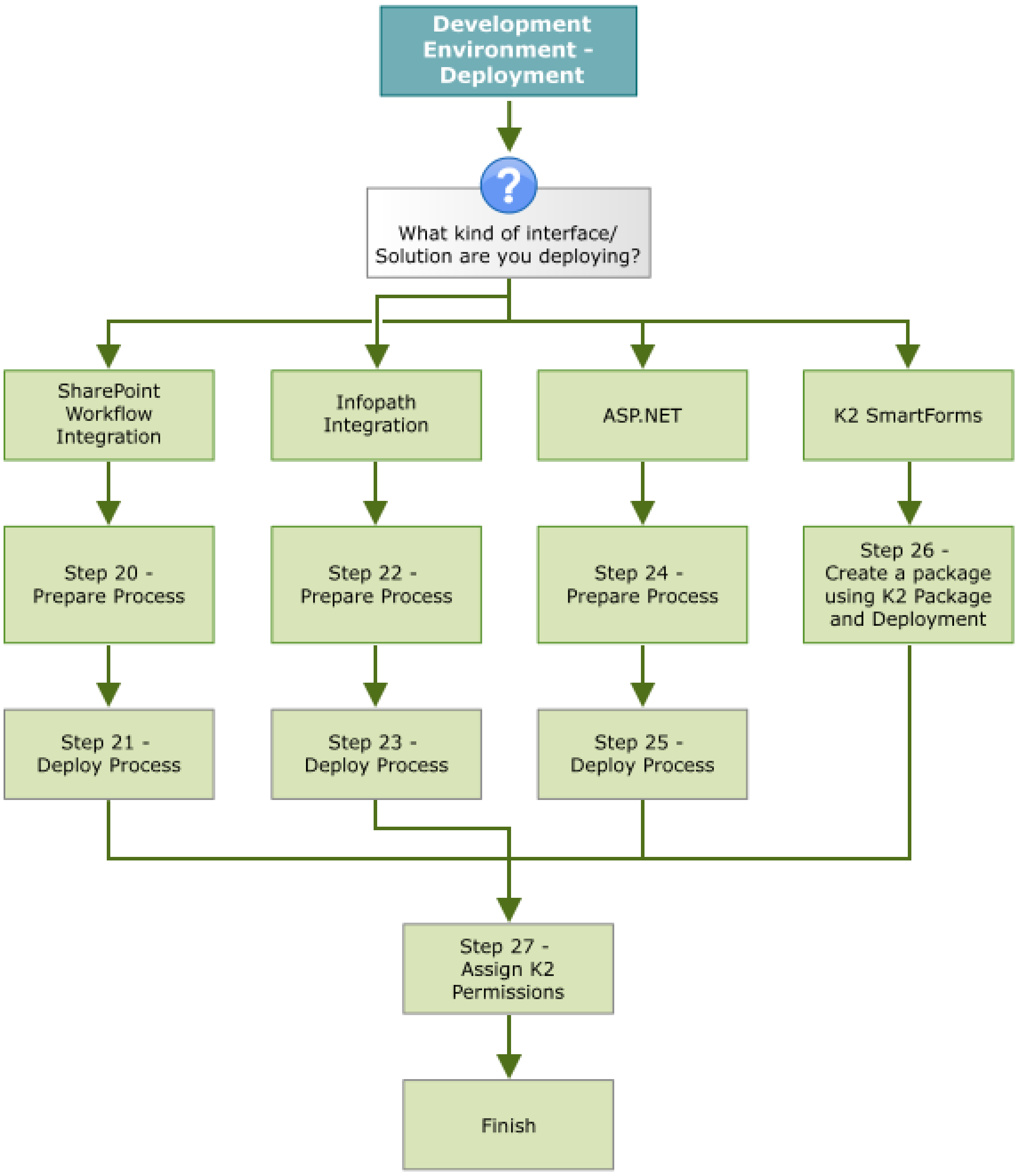
NON K2 COMPONENTS


Development Environment - Non K2 Components
Do you have dependant SharePoint Resources?
YES
Step 1 - Create SharePoint Resources
DONE
NO
Do you have custom Database Configurations?
YES
Step 2 - Create Databases
DONE
NO
Do you make use of CRM?
YES
Step 3 - Prepare CRM
DONE
NO
Do you have custom Web Services?
YES
Step 4 - Create custom Web Services
DONE
NO
Do you have other 3rd party resources?
YES
Step 5 - Deploy other 3rd party resources
DONE
NO
Do you have specific permissions to configure?
YES
Step 6 - Configure User and Service Account permissions
DONE
NO
Do you have custom assembles?
YES
Step 7 - GAC custom assemblies
NO
Finish
DONE

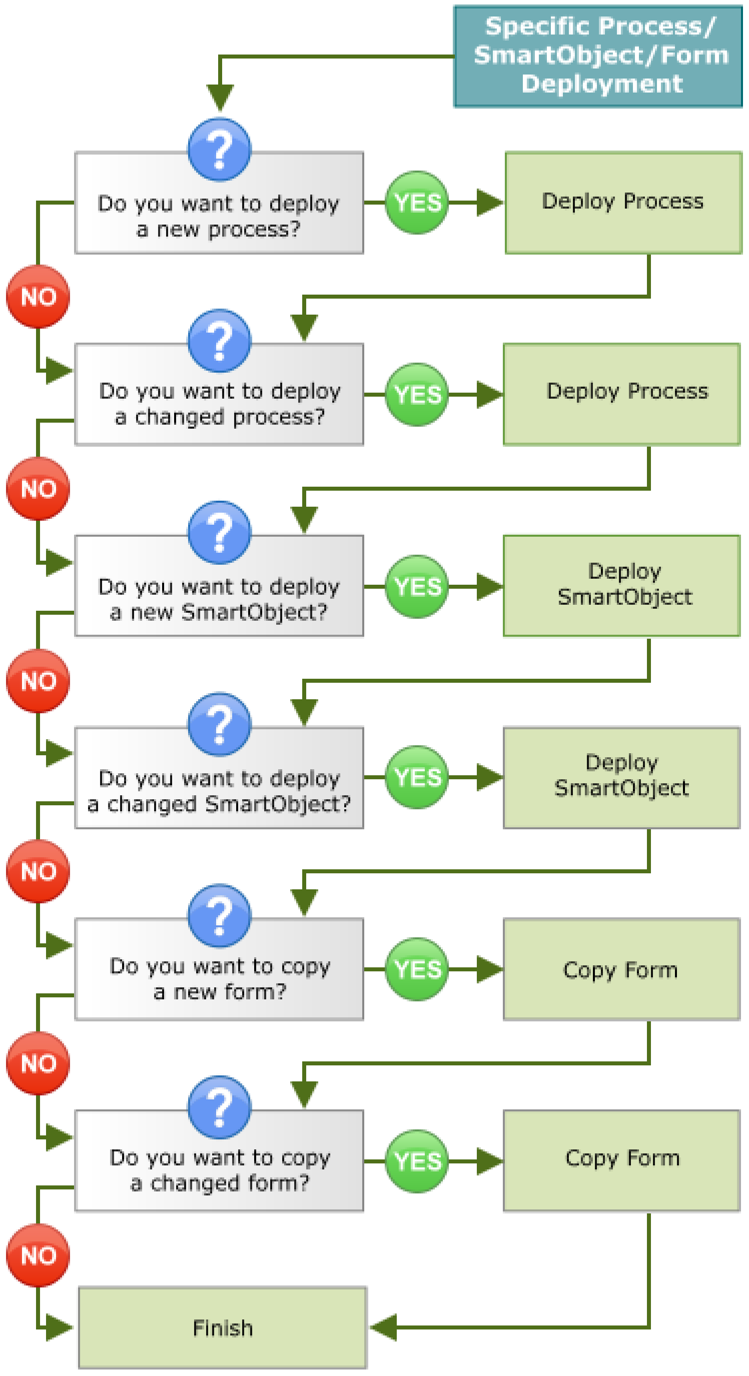
K2 COMPONENTS


Development Environment - K2 Components
Do you have specific K2 Product Assemblies?
YES
Step 8 - Configure K2 Product Assemblies
NO
DONE
Do you have Custom Security Providers?
YES
Step 9 - Create Custom Security Providers
NO
DONE
Do you make use of Custom User Managers?
YES
Step 10 - Create Custom User Managers
NO
DONE
Do you have new Service Types and Instances?
YES
Step 11 - Create Service Types and Instances
NO
DONE
Do you have specific Roles?
YES
Step 12 - Create Roles
NO
DONE
Do you make use of Zones and Working Hours?
YES
Step 13 - Create Zones and Working Hours
DONE
NO
Do you have custom notification providers?
YES
Step 14 - Create custom notification providers and deploy custom event notifications
DONE
NO
Do you have custom inline Functions?
YES
Step 15 - Create Inline Functions
DONE
NO
Do you have custom event wizards?
YES
Step 16 - Create custom event wizards
DONE
NO
Do you have custom Environment Fields?
YES
Step 17 - Create custom Environment Fields
DONE
NO
Finish

DEPLOYMENT


Development Environment - Deployment
What kind of interface/ Solution are you deploying?
SharePoint Workflow Integration
Infopath Integration
ASP.NET
K2 SmartForms
Step 20 - Prepare Process
Step 22 - Prepare Process
Step 24 - Prepare Process
Step 26 - Create a package using K2 Package and Deployment
Step 21 - Deploy Process
Step 23 - Deploy Process
Step 25 - Deploy Process
Step 27 - Assign K2 Permissions
Finish

SPECIFIC PROCESS / SMARTOBJECT / FORM DEPLOYMENT


Specific Process/ SmartObject/Form Deployment
Do you want to deploy a new process?
YES
Deploy Process
NO
Do you want to deploy a changed process?
YES
Deploy Process
NO
Do you want to deploy a new SmartObject?
YES
Deploy SmartObject
NO
Do you want to deploy a changed SmartObject?
YES
Deploy SmartObject
NO
Do you want to copy a new form?
YES
Copy Form
NO
Do you want to copy a changed form?
YES
Copy Form
NO
Finish

DEVELOPMENT ENVIRONMENT

TESTING OF DEVELOPMENT SOLUTIONS

Testing of the deployed solutions on the development environment will not influence the day to day business processes as these solutions are deployed separately from the production environment. This is the reason why it is important for the environment fields to be different from that of the production environment and the environment setup to be separate from the production environment.

Note that it is extremely important to ensure that the testing of these solutions on the development environment are completed successfully before moving to the production environment.

Moving artifacts from development to production will include artifacts but no data. All test data will remain on the UAT or staging environment.

PRE-DEPLOYMENT PLANNING

Planning for deploying the development solutions to production is necessary to ensure a successful move without interruptions of the day to day business processes. For this document the assumption is made that the target environment is fully operational. New environments should be installed and validated before installing solution-specific artifacts. Failure to do so will complicate and prolong troubleshooting. An environment that is not operating before installing the solution will not be improved by installing the solutions.

BEST PRACTICES FOR DEPLOYMENT METHODOLOGY

Generally there are multiple deployment environments. For example development, integration test, QA, staging, and production. The complexity of the project should dictate the level of effort dedicated to testing and deployment practices. A solution that took a week to develop will require less rigor than one that took a dedicated team months of effort.

Development and integration environments tend to have a lot of test/fix/deploy cycles and the developers may still be deciding which components and techniques will be used. In these environments, manual deployments are allowed. Developers should create notes about what they are deploying and any dependencies on other components. As the deployments move up the chain of environments towards production, more rigor should be applied through the use of scripted deployments.

The staging environment should mirror the complexity of the production environment, but does not necessarily have the same level of capacity. For example, if the production environment has multiple load-balanced K2 nodes, the staging environment should have load balanced nodes, but there may be fewer nodes. If the production SQL Server is clustered, the staging environment should be clustered, even if the servers in the cluster are less powerful. The reason for making this environment a similar layout to production is to validate developer assumptions about the environment and make sure any project solutions created will operate correctly in the full production environment. An additional use of this environment is to research and simulate problems occurring in production that would be too disruptive to test in the production environment. The staging environment can also be used for performance and capacity testing of new solutions before deployment to production.

When deployment from the development environment to production environment takes place for the first time, all components need to be deployed. Thereafter only new workflows, new SmartObjects and new forms can be deployed as required.

STEP BY STEP GUIDE FOR A FULL ENVIRONMENT DEPLOYMENT

DEPLOYMENT OF NON-K2 SOLUTION COMPONENTS

Non-K2 solution components can vary greatly depending on customer requirements. The order and installation procedures for these components will have to be designed on a case-by-case basis. Generally these components should be deployed and validated before installing K2 artifacts as there are likely to be dependencies.

K2 only provides tooling and mechanisms to deploy its own artifacts. Other integrated solution components such as web services, user interfaces, and SharePoint features can be made part of a scripted deployment via MSBUILD. K2 build packages do not handle or even manage the deployment of other solution components.

STEP 1 – SHAREPOINT RESOURCES

Create SharePoint resources such as SharePoint Sites, Lists and Libraries and SharePoint Groups, etc.

STEP 2 – DATABASES

Create the SQL or Oracle database, which ever is relevant. Create tables, stored procedures, etc as required.

STEP 3 – CRM PREPARATION

Prepare CRM as per required setup.

STEP 4 – CUSTOM WEB SERVICES

Deploy any custom web services required.

STEP 5 – OTHER 3RD PARTY RESOURCES

Deploy any other 3rd party resources that might be required as per the environment setup.

STEP 6 - PERMISSIONS (USERS & SERVICE ACCOUNTS)

Assign the appropriate permissions related to users and service accounts. This will be specific to deployment and access rights for the deployment of processes and SmartObjects in later steps.

STEP 7 - GACED ASSEMBLIES

It is not common to include assemblies in the Global Assembly Cache as part of K2 solutions, however it does occur for some solution architectures. Examples include 3rd party components and custom code utility assemblies invoked from the workflow. These assemblies should be deployed to the GAC on each server in the farm before deploying dependent solution items. Custom MSBuild extensions or .msi pacakges should be used to automate the deployment.

Assemblies referenced in a process definition that have the copy local attribute should not be deployed to the target K2 server farm. The build package created for the process definition will handle packaging and deployment of these assemblies to the K2 server.

DEPLOYMENT OF K2 COMPONENTS

The types of K2 components deployed will vary by project, but the general order described in this section should be followed

STEP 8 - K2 PRODUCT ASSEMBLIES ON AN APPLICATION SERVER

In some cases, solution architecture will call for a separate application server layer. A simple approach to deploying product assemblies is to use the K2 product installer to install only the K2 Workspace. K2 Workspace has all of the components and assemblies commonly used for API programming. The actual K2 Workspace web site can be disabled after installation. If the application server will make use of only a limited number of product assemblies, those assemblies and their dependencies can be packaged into a .msi package and scripted for deployment.

STEP 9 - CUSTOM SECURITY PROVIDERS

The blackpearl platform can be extended as part of the solution. The product contains a number of component servers that provide common services for the platform. The URM (User Relationship Manager) server handles security and directory services for user, group, and role resolution. The URM provides pluggable architecture that can be extended. A common reason customers might want to create their own user provider is for a custom application that handles its own user and group management.

Occasionally solution architectures dictate a need for a custom security provider. Creation and deployment of these providers is covered in detail within the K2 product documentation. Generally once a custom security provider has been deployed, it does not change unless there are bug fixes. This differs from the broader solution implementation effort which frequently changes over time based upon updated business requirements and/or project implementation cycles.

Depending on your installation, one or more Out of the Box (OOTB) user managers may have already been registered in your environment.

Label	Use
K2	Active Directory (Default)
K2ADFS	Active Directory Federation Services (Claims)
K2FORMS	Forms-based authentication
K2SAP	SAP User Manager
SP	SharePoint Users and Groups
SQLUM	SQL Server User Manager
K2LDAP	ADFS LDAP Identity Provider

For a list of default Security Provider Labels, refer to the following section in the K2 blackpearl Getting Started Guide:

Installation and Configuration > Prerequisites > Environment Configuration > User Authentication and Security > Introduction to User Managers

Deploying a Security Provider

Custom security providers may add specific details to the following steps because of their custom nature.

1. Decide on a Security Label for the provider. The label can be any value that is not already registered on the K2 server.

2. Prepare the configuration information for the provider according to its documentation. Configuration data is stored in the AuthInit field of the SecurityLabels table when the provider is registered.

General instructions for registering security providers can be found in K2 knowledge base articles KB000186 and KB000331 and in an article on K2 Underground.

1. Copy the project output assembly to each K2 server in the target environment farm in the directory <Drive>:\<Program Files>\K2 blackpearl\Host Server\Bin\securityproviders
2. Make a backup copy of <Drive>:\<Program Files>\K2 blackpearl\Host Server\Bin\K2HostServer.config file.
3. Stop each K2 Host Server in the farm.
4. Edit the K2HostServer.config on each server and add the useassemblyregistration key to the appSettings section:

```
<appSettings>
...
<add key="useassemblyregistration" value="true" />
...
</appSettings>
```

Do not copy the K2HostServer.config file between servers. It contains encrypted sections that are keyed to the machine where it resides.

5. Execute the following SQL command using SQL Studio or another tool. The exact values for this command should be provided in the documentation of the provider. Before running the command, change the assembly name if needed.
6. Include the public key token of the assembly in the SQL command. You can obtain the public key of your assembly by running the Visual Studio command line tool command "SN –T" against your assembly.

```
use [K2HostServer]
INSERT INTO [K2HostServer].[dbo].[AssemblyRegistration]
           ([AssemblyID]
           ,[AssemblyName]
           ,[PublicKeyToken]
           ,[Enabled])
     VALUES
           (NEWID(),'YOUR.ASSEMBLY.NAME','e712067b8c7deb62',1)
GO
```

7. Start each K2 Host Server in the farm to register the assembly.
8. Execute the following SQL command, replacing the sample values with the your security label in place of 'LABEL', the fully qualified namespace and name of your provider class, and the AuthInit values you created in step 2.

```
declare @id uniqueidentifier, @a_newid uniqueidentifier;
set @a_newid =  newid()
select @id = SecurityProviderID from SecurityProviders where ProviderClassName = 'YOUR.ASSEMBLY.NAME.CLASS'
insert into SecurityLabels values (@a_newid, 'LABEL', @id,
'<AuthInit>
        …provider specific configuration
</AuthInit>', @id, null, '0')
GO
```

9. Restart each K2 Host Server and your security provider will now be available.

Note: It is recommended to use the same security label in each environment when registering custom providers.

STEP 10 - REGISTER CUSTOM USER MANAGERS

If custom user managers were created and used in the development environment, these need to be created in the production environment as well. For more information on how to create these custom user managers, see the "Creating a Custom User Manager" topic in the K2 Developer Reference documentation.

STEP 11 - SERVICE TYPES AND INSTANCES

Services are the part of the SmartObject infrastructure that does the work for SmartObjects. A ServiceType is a .NET assembly that contains the executable code and it must be registered with the blackpearl server. A service instance is the union of a ServiceType with its configuration data. SmartObjects are the public definition of methods and properties provided by underlying service instances but they do not contain any executable code. Each service instance has a unique GUID that SmartObjects bind to.

Some ServiceTypes and instances are supplied as part of the product and are pre-configured. These service instances use fixed GUIDs that are the same for every K2 installation. Therefore, any SmartObjects that use these service instances can be deployed to any environment without configuring a service instance. However, if you create a new instance on one of these services, you will have to configure that instance in a new environment where dependent SmartObjects are deployed. The following SmartObjects are OOTB SmartObjects and are automatically available in every K2 environment:

ServiceType	Instance
Account Management	Account Management Services
AD Service2	Active Directory Service2
Exchange Administration	Exchange Administration
Exchange Management	Exchange Management
Exchange Metadata	Exchange Metadata Service
K2 Generic Settings Service	K2 Generic Settings Service
SmartBox Service	SmartBox Service
Task Allocation Service	Task Allocation Service
User Role Manager Service	URM Service
Workflow Reporting	Workflow Reporting Service
Workflow Service	Workflow Service

The following ServiceTypes can be configured manually if required:

- CRM – Can be configured during installation of K2
- Oracle – Needs to be manually configured and an instance needs to be created
- CSOM - Needs to be manually configured and an instance needs to be created
- SQL - Needs to be manually configured and an instance needs to be created

Built-in service instances should not be reconfigured or have their GUID changed.

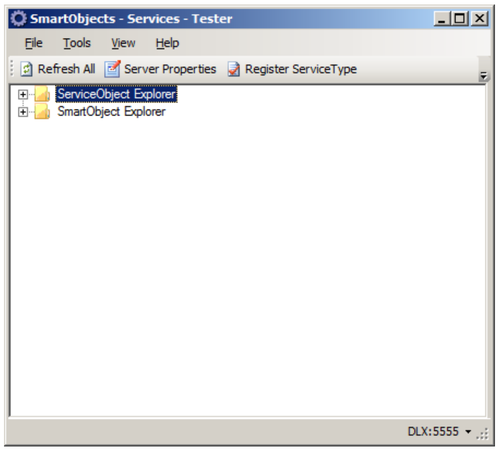
Because a service instance has configuration data, multiple instances of the service type can be registered within a single K2 blackpearl instance with different configuration data. For example, the SQL Server Service can have multiple instances that each connect to a different database. When SmartObjects are deployed to a new environment, any new Service Instances will need to be registered. These strict dependencies are enforced on these components due to their close interrelation and are managed by the K2 server. If a SmartObject is deployed and there is a missing service instance, the deployment will fail with an error stating that the "Parent does not exist in this environment"; which by "parent" it is meant K2 cannot locate a ServiceInstance that is registered with the same GUID that is included within the method mappings of the SmartObject definition.

ServiceType and Service Instance registration is not currently part of the blackpearl product deployment package. These can be registered manually using the steps in the sections below. A sample project containing MSBuild tasks to handle these deployments as part of a package can be found on K2 Underground.

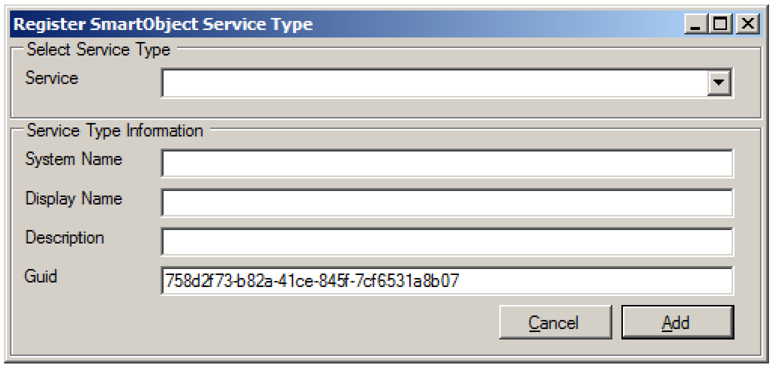
Registering a Service Type

Registering a new ServiceType is only needed if the type has not been added to the environment before. A ServiceType only needs to be registered once per environment regardless of how many service instances of this type are needed.

1. Build the custom service type assembly in Visual Studio. The .NET framework target should be compatible with the version used for the blackpearl server.
2. Locate the output assembly in the <project's >\bin\Debug (or Release) folder.
3. Copy the assembly to the **<drive>:\<Program Files>\K2 blackpearl\ServiceBroker** folder of *each blackpearl server in the target environment*. The .pdb file is not needed.
4. On *only one* of the blackpearl servers navigate to the **<drive>:\<Program Files>\K2 blackpearl\bin** folder and double-click **SmartObject ServiceTester.exe.** Note that the SmartObject Service Tester.exe gets installed with Client Tools only. The SmartObject Service Tester.exe can be copied from the client machine to one of the Server machines.


SmartObjects - Services - Tester
File Tools View Help
Refresh All Server Properties Register ServiceType
ServiceObject Explorer
SmartObject Explorer
DLX:5555

5. Click on the **Register ServiceType** button on the top command bar.



Guid: 758d2f73-b82a-41ce-845f-7cf6531a8b07

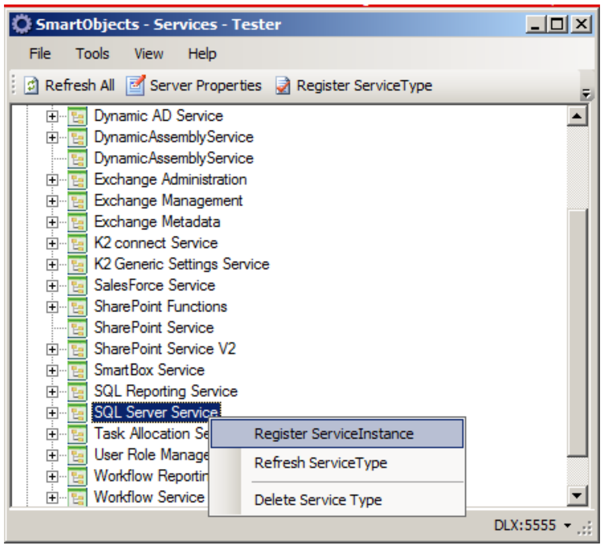
6. In the registration dialog:
   a. Select the service assembly from the Service drop down list.
   b. Enter the **System Name, Display Name, and Description**
7. *Note you need to use the exact same GUID in all environments as used in the Development Environment for each ServiceType.*
8. Click **Add**, and then OK again on the Service Type Added message.
9. If you are going to register a service instance, leave the utility open. Otherwise, close the utility.

Registering a Service Instance

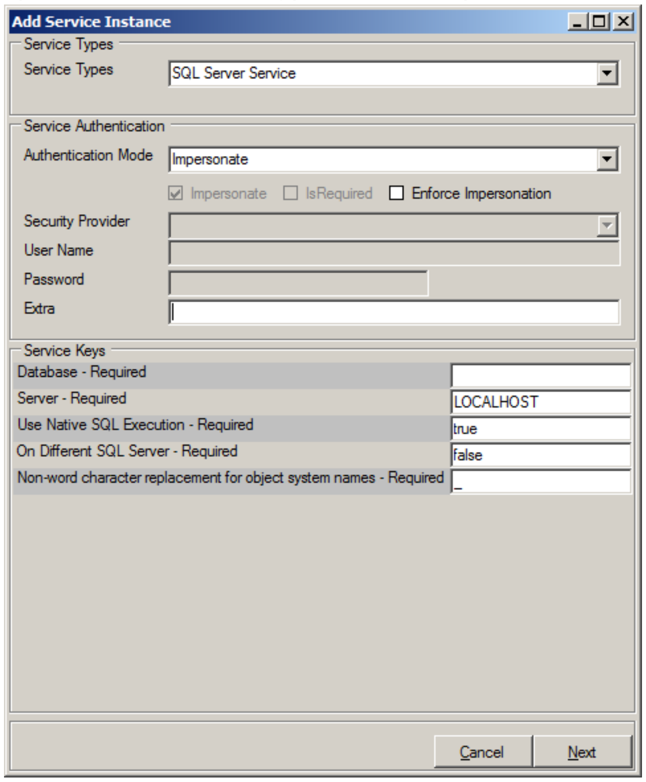
Before a new Service Type can be used, an instance must be created. A service type can be thought of as the base definition of how to talk to an underlying system to retrieve data, where a service instance is the configuration information for that instance of the service. You can register multiple instances of a service. A service instance only needs to be registered on one K2 server in a farm per environment; all servers in the farm will be able to make use of the instance.

To register a new service instance for a service type:

1. Use the **SmartObject Service Tester** utility.
2. Drill down into the Service Object Explorer node of the tree view and locate the appropriate service type.
3. Right-click on the service type and choose **Register ServiceInstance** from the context menu.



4. The resulting dialog is used to configure the service instance. The top part of the dialog contains values common to all service instances whereas the bottom section **Service Keys** contains configurable items specific to this service type. The example below is for a SQL Server Service instance.



Service Types: SQL Server Service

Authentication Mode: Impersonate

☑ Impersonate ☐ IsRequired ☐ Enforce Impersonation

Security Provider

User Name

Password

Extra

Database - Required	
Server - Required	LOCALHOST
Use Native SQL Execution - Required	true
On Different SQL Server - Required	false
Non-word character replacement for object system names - Required	_

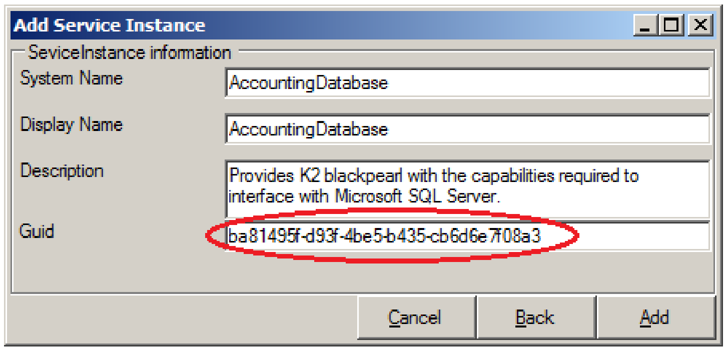
a. Select the **Service Type** from the drop-down list if needed

b. There are four **Authentication Mode**s that can be used:

i. **Impersonate** will use the credentials of the user making the request if it originates from a user interface. For example, if the SmartObject method is being called from an InfoPath form, the credentials of the user who opened the form will be used. If the request originates from within the workflow process, the service account running blackpearl will be used.
ii. **Service Account** will use the credentials of the service account running blackpearl.
iii. **SSO** will store the supplied credentials in blackpearl's single-sign on subsystem. The person creating this instance can then use the single-sign on area of K2 Workspace in the future to update the password used by the service.
iv. **Static –** credentials entered on this dialog will be used to make all requests. If the password needs to be changed in the future, an administrator will need to use this dialog to change it.

*Note: The two most common authentication options are **Impersonate** and **Service Account**.*

5. Consult the ServiceType's documentation for appropriate values to enter into the lower section of the dialog. These are specific to each service type.
6. Click **Next** to complete the details of the service instance:
   a. SystemName
   b. DisplayName
   c. Description
   d. Guid



System Name	AccountingDatabase
Display Name	AccountingDatabase
Description	Provides K2 blackpearl with the capabilities required to interface with Microsoft SQL Server.
Guid	ba81495f-d93f-4be5-b435-cb6d6e7f08a3

7. GUID's needs to be consistent throughout environments per Service Instance. Ensure to use the GUID from the development environment on the testing environment and then later the production environment.
8. Click **Add**
9. Exit the utility once all instances are configured.

IMPORTANT - The Service Instance GUID is the one that SmartObjects bind to when they are deployed. When you are deploying a service instance to multiple environments, you should record the GUID you initially used in the development environment and then use that GUID for the same service instance in all environments. Using the same GUID for this instance in all environments will allow the SmartObjects to deploy without error.

Document the Service Types in the production environment for future reference in case changes occur.

K2 Service types

	Detail	Note
Physical K2 server	K2 server 1	The IP, server aliases and server name to the server where this type needs to be registered/validated
	K2 server 2	The IP, server aliases and server name to the server where this type needs to be registered/validated
Service Type Name	Application specified name here	
Type GUID	1111-1111-11111111	
Config	Name/Value pair of the config specific to this type OR <xml/> text of the required configuration	

Syncing Service Instances between Environments

If you did not correctly register a service instance in an environment with the same GUID as the other environments you will get an error when you try to deploy the SmartObjects:

```
Error 1 Deploy smartobjects: Task error: SmartObjectServer Exception: Dependancy could not be
created: System.Exception: Dependancy could not be created. Parent does not exist in this
environment. Check Data property of exception.
   at SourceCode.Hosting.Services.DependancyService.VerifyObjects(List`1 parents)
   at SourceCode.Hosting.Services.DependancyService.CreateDepenancyBatch(Dictionary`2
depenancyBatch). SmartObject: 'SmartObject1'
Parent ID = 'ae6c687b-af27-4160-843d-016c782757ed'
Parent Name = 'Denallix News'
Parent Class Name = ''
Parent Version = ''
Parent Extra Data = 'SourceCode.SmartObjects.Services.SharePoint.SharePointService'
```

In the error message the Parent ID Guid is the ID of the missing service instance.

Although utilities exist on K2 Underground to change the GUID of an existing service instance, the best course of action is to correctly register the service instance using the same GUID that was used in the other environments. If you change the GUID of an existing instance, any SmartObjects already deployed against that instance will stop working.

STEP 12 - ROLES

K2 Roles must be created on the production environment the same way they were created on the development environment. Roles can be defined in the management console of K2 Workspace to define a user or group of users for task assignments. By defining roles outside of a process, the membership of that role can be changed without changing the process. This technique allows test users to fill the membership in development and test environments while real users are members in production.

There is no built-in deployment package support for role deployment to a new environment. However, there is at least one utility on K2 Underground to assist with moving roles between environments.

The following manual steps can be used to create roles in a target environment:

1. Make a list of all roles used in the processes to be deployed.
2. A user with administrative rights in the target environment should go to the management console of K2 Workspace.
3. Drill down into the tree view for the appropriate server farm to the **Roles** node.
4. Right-click on **Roles** and select **Add New Role**
5. From the resulting dialog users and groups can be added to the role from any security provider.
6. Once role membership has been defined, remember to click the **Save** button on the toolbar.

STEP 13 - CREATE REQUIRED ZONES/WORKING HOURS

If zones and/or working hours were used in the solutions being deployed to production, these must be manually created in the production environment in the same way it was created in the development environment. The setup is done in K2 Workspace>Management Console. For more information on zones/working hours, refer to the topic in the K2blackpearl User Guide found in the following location:

Management and Administration > Workspace Management > Management Console > Workflow Server > Working Hours Configuration

STEP 14 - REGISTER CUSTOM NOTIFICATION PROVIDERS AND DEPLOY CUSTOM EVENT NOTIFICATIONS

The custom notification providers used in the development environment must be manually registered in the production environment in K2 Workspace>Management Console. The custom event notifications can then also be deployed in the production environment. For more information on custom providers and custom event notifications, refer to the topics found in the following locations:

Developer Reference Guide > Deverloper Reference > Extending the K2 Platform > How to add a 3rd party event recorder to the K2blackpearl Server

AND

K2blackpearl User Guide > Management and Administration > Workspace Management > Notification Event > Notification Custom Event

STEP 15 - INLINE FUNCTIONS

Inline functions were introduced in K2 blackpearl release 4.5. They are useful utility functions that can be used in many of the event wizards to convert or manipulate data. There are many in-line functions supplied OOTB with blackpearl, and developers can create their own. If custom in-line functions are created as part of the solution, they must be deployed to the K2 server farm for runtime consumption. In-line function implementation assemblies are not included in K2 deployment packages. They must be deployed manually using the following steps:

1. Build the custom inline function project.
2. Locate the output assembly.
3. Shut down each K2 Server in the farm of the target environment.
4. Deploy the output assembly to the following locations:
    a. For use in K2 Designer for SharePoint copy to **<install drive>:\<Program Files>\K2 blackpearl\Bin\Function**s on all SharePoint web front-ends.
    b. GAC the assembly on all SharePoint web front-ends running the K2 for SharePoint components. *Note: It is only necessary to deploy the assembly to the GAC if it will be used in K2 Designer for SharePoint.*
    c. For each K2 Server in the farm copy to **<install drive>:\<Program Files>\K2 blackpearl\Host Server\Bin**.
    d. Restart each K2 server.

STEP 16 - CUSTOM EVENT WIZARDS

Custom event wizards allow advanced developers to create process event wizards that can be used by other workflow designers in Visual Studio and K2 Studio. If created according to K2 development guidelines, nothing needs to be deployed to the K2 server farm unless the wizard references an assembly in the GAC. The runtime code of the custom event wizard will be included in the process deployment package.

STEP 17 - CREATE ENVIRONMENT FIELDS

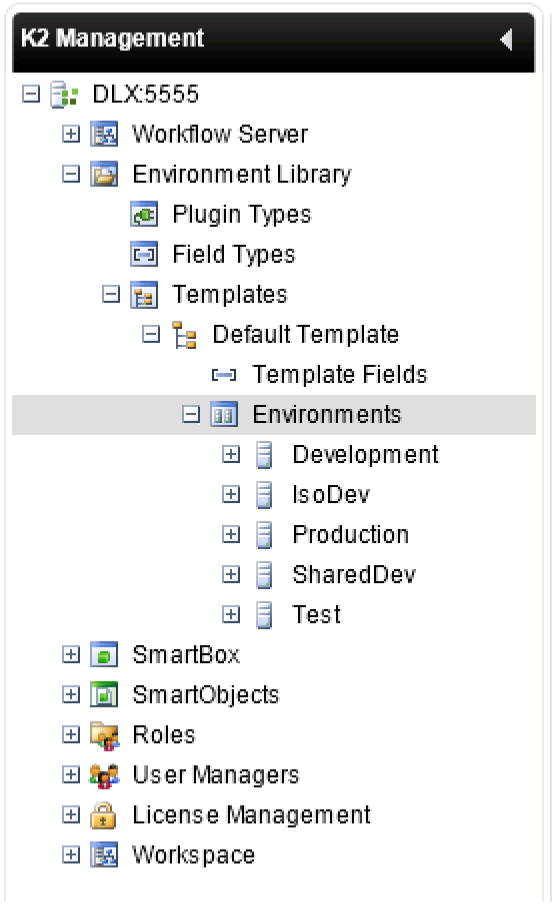
Creating environment fields will include built-in as well as speciality Fields for example SharePoint Server URL's. The explanation below is based on a common scenario, however not all environments might be set up as per this scenario. If your scenario is different, use this only as a guideline.

Requirements for the environment

The environment might consist of:

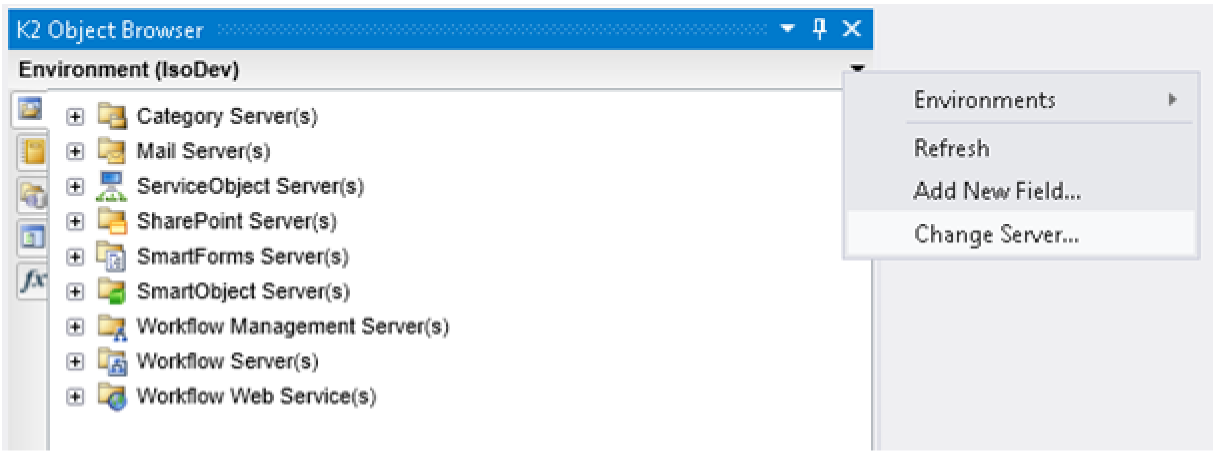
- A number of isolated virtual machines for developers. It would be ideal if these have a network connection to the SharedDev environment (listed below), but have the same local settings.
- A SharedDev environment
- A number of other environments for example the Test Environment and Production Environment.

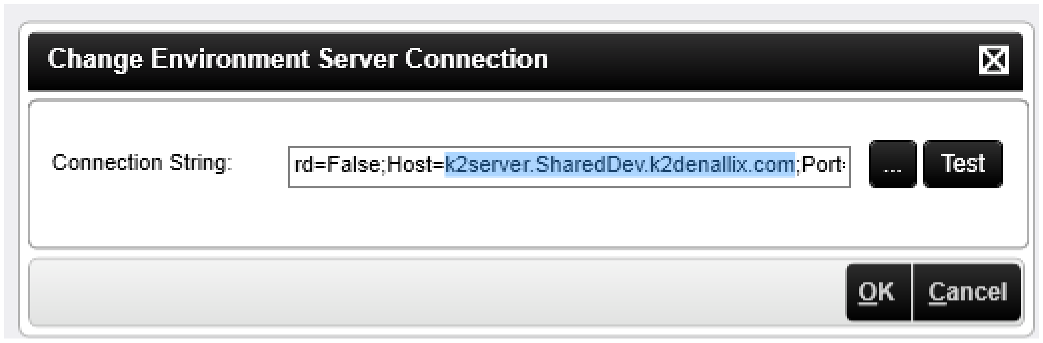
Before development starts, create an entry for each of these environments in K2 Workspace>Management Console>Environment Library>Templates>Default Templates>Environments. See image below:



Note that the Isolated Development Environment, SharedDev Environment, Test Environment and Production Environment each have their own entry. Each of these environments will have their own set of built-in environment fields. Custom environment fields can be added if necessary.

On each Developer Environment, change the Server to the SharedDev Server from the K2 Object Browser in K2 Designer for Visual Studio as illustrated below:

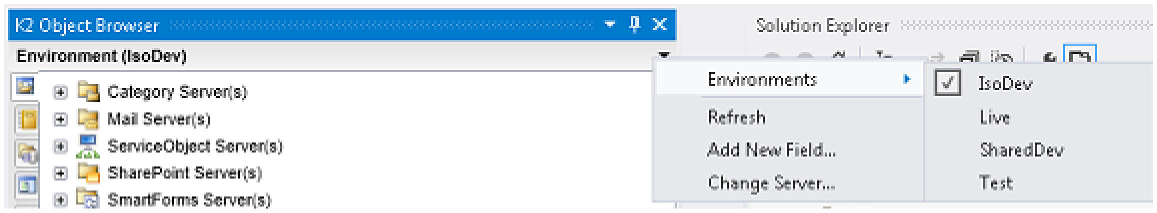



Change Environment Server Connection

Connection String: rd=False;Host=k2server.SharedDev.k2denallix.com;Port

Note: this image is only an example. Ensure to use the correct server name.

Change the Deployment Environment to the IsoDev Environment as illustrated below:

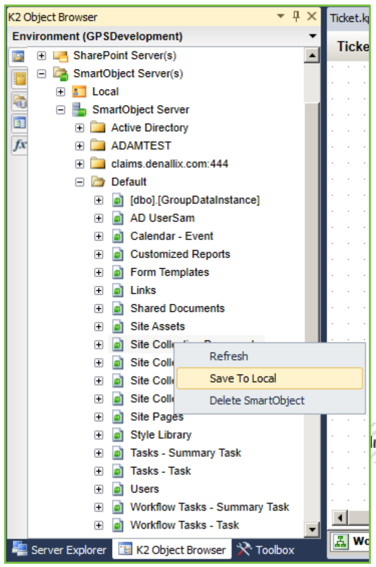


This now means that all custom environment fields created by any developer are **shared** amongst the team and **stored** on the SharedDev server. However it also enables developers to deploy to their own IsoDev server when required.

Before deployment, the development team needs to review all new **Custom Environment** and set/change the value for all environments (SharedDev, test, live etc.)

STEP 18 – SMARTOBJECTS

SmartObjects can be deployed by using deployment packages or directly from Visual Studio/K2 Studio. Deploying directly from the design tool is generally only done for the development and integration test environments. As a best practice, all other environments should use a deploy package to insure repeatability and consistency. Deploy packages can be created in K2 Designer for Visual Studio or K2 Studio using the **Create Deployment Package** menu option. The deployment package is an MSBUILD script that contains everything needed to deploy its contents to the K2 server farm and update the environment settings. A deploy package only needs to be run on one K2 server in a farm of multiple servers. Refer to KB000188 for details on the creation and contents of deployment packages.



Sometimes in the early stages of development, developers might create SmartObjects through the K2 Site Settings menu in SharePoint. When the site settings option is used, two steps happen automatically: a new instance of the SharePoint Service is created and SmartObjects are automatically built and deployed based on that service. While this is fine for quickly testing a new idea, it is difficult to move between environments.

If SmartObjects created this way need to be deployed to other environments, use the SmartObject Service Tester tool (discussed previously) to look up the

GUID of the auto-created service instance. This same GUID will be needed to register the service instance in other environments. To get the auto-created SmartObjects into a project (and source control) use the K2 Object Browser in Visual Studio or K2 Studio. Drill down into the **SmartObject Server** node to locate the generated SmartObjects. Right-click on the SmartObject and choose **Save To Local** to import the SmartObject definition (SODX) file into the current project. Importing the definition into a project also allows access to advanced settings.

It is a K2 best practice to put SmartObjects and process designs into different Visual Studio projects. Since process designs frequently depend on SmartObjects, they should be deployed first, independently of the process project.

For more information on the import/export functionality on CRM and SQL SmartObjects refer to the following topic found in the K2blackpearl User Guide:

Management and Administration > Environment Setup > Interacting with SharePoint > SmartObjects > SharePoint SmartObjects > Import and Export SQL and CRM SmartObjects

STEP 19 - K2 CONNECT SERVICE INSTANCES

K2 connect is an additional component to K2 blackpearl. K2 connect is another service that can be used to provide functionality to SmartObjects. As such, before SmartObjects can be moved into a different environment from where they were originally deployed, the K2 connect service must be setup with the dependent service instances.

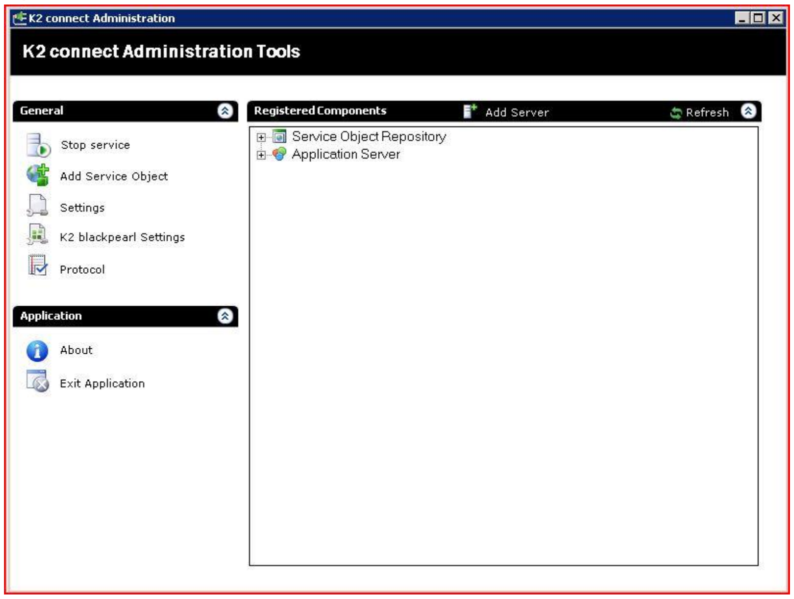
Unlike other services in K2, the connect service has its own design and deployment tools to make the experience simpler than custom code services.

File Based deployment method

This method uses the K2 connect SVD files that were created during service development to deploy the service definitions to the next environment. To deploy the K2 connect service objects:

1. Save the SVD files created using the K2 connect Service Object Designer to a file location where they can be accessed from the server to which they are being deployed.
2. Log into the K2 blackpearl server in the target environment.

3. Open the K2 connect Administration Tool.

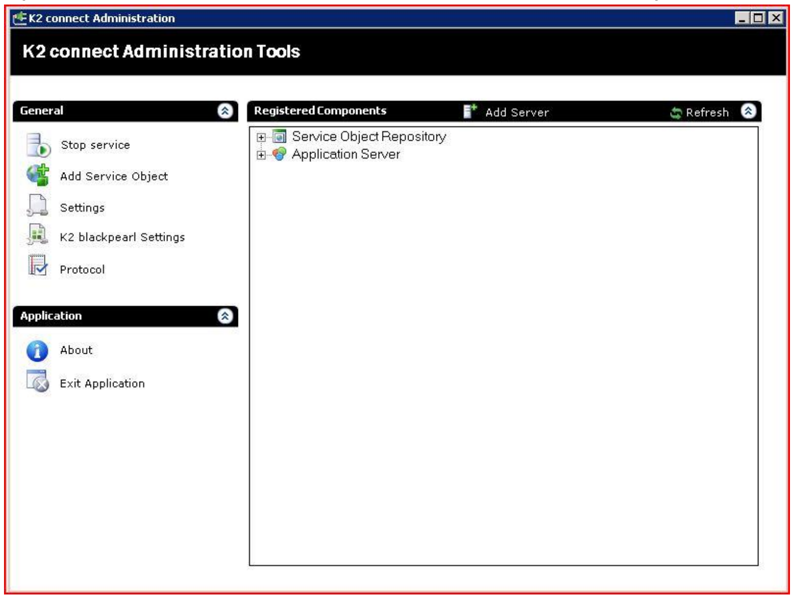


4. Click “Add Service Object”.
5. In the file dialogue, select the SVD files that were created with the Service Object Designer in Visual Studio.
6. Click “Open”.

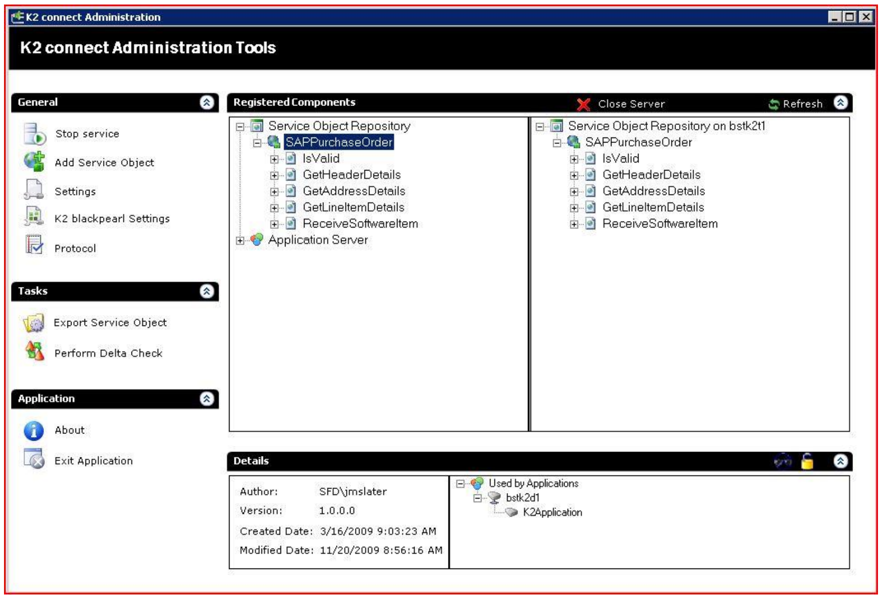
K2 connect Administration Tool deployment method

This method uses the K2 connect Administration tool to deploy the service definitions to the target environment. To deploy the K2 connect service objects:

1. Open the K2 connect Administration Tool on the development K2 server



2. In the right hand panel, click on “Add Server” on the tool bar and provide the name of the target K2 server to where you want to deploy your Service Object



Author:	SFD\jmslater
Version:	1.0.0.0
Created Date:	3/16/2009 9:03:23 AM
Modified Date:	11/20/2009 8:56:16 AM

3. Click on the Service Object you wish to deploy to the target environment. Drag the service object (in this example , the service object is called SAPPurchaseOrder) from the left panel (source server) to the right panel (target server).

PROCESSES

Like SmartObjects, workflow process designs are deployed using the design tools or a deployment package. Processes are generally the last K2 artifacts deployed as they can have a chain of dependencies on all of the items that preceded it. A deployment package for workflows is created in the design tool the same way they are for SmartObjects. Once the deployment MSBuild package is created, it can be moved to each environment for deployment without re-creating the package. This insures the package that was built, is the same one that was tested before it was moved to production.

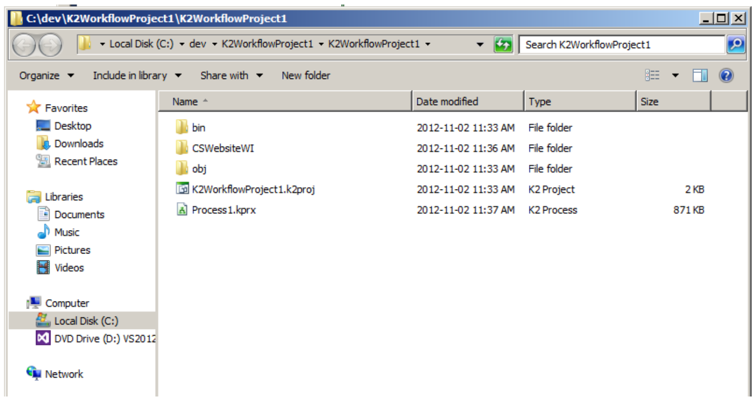
Also refer to KB000188

SHAREPOINT WORKFLOW INTEGRATION

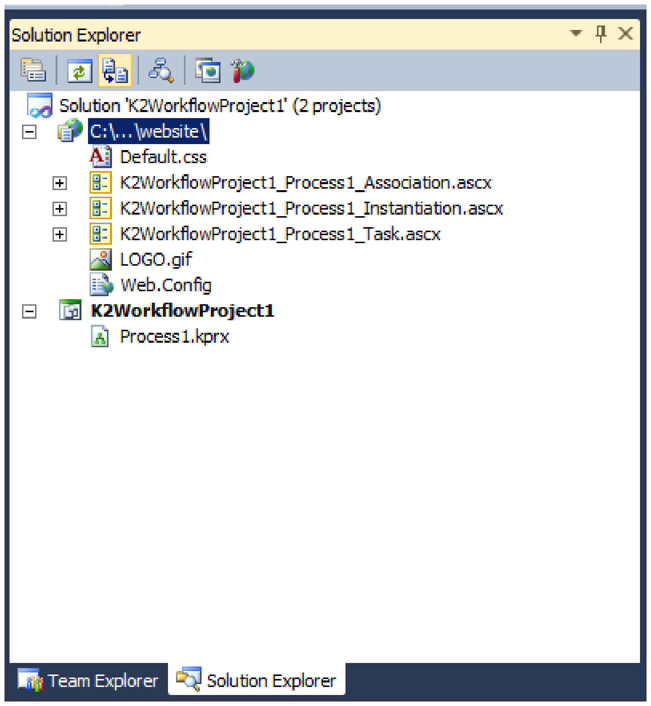
This section will explain the deployment of K2 SharePoint Integrated workflows and associated custom forms from a development environment to a production environment..

STEP 20 – PREPARE PROCESS

In the case of users selecting to have custom forms generated for their K2 SharePoint Workflow Integrated process, on completing the process wizard a folder will be created in the root of the K2 Project Folder called “CSWebsiteWI”.



Name	Date modified	Type	Size
bin	2012-11-02 11:33 AM	File folder	
CSWebsiteWI	2012-11-02 11:36 AM	File folder	
obj	2012-11-02 11:33 AM	File folder	
K2WorkflowProject1.k2proj	2012-11-02 11:33 AM	K2 Project	2 KB
Process1.kprx	2012-11-02 11:37 AM	K2 Process	871 KB



```
Solution 'K2WorkflowProject1' (2 projects)
⊟ C:\...\website\
      Default.css
  ⊞   K2WorkflowProject1_Process1_Association.ascx
  ⊞   K2WorkflowProject1_Process1_Instantiation.ascx
  ⊞   K2WorkflowProject1_Process1_Task.ascx
      LOGO.gif
      Web.Config
⊟ K2WorkflowProject1
      Process1.kprx
```

This folder will hold the ASP.Net files and resources needed to successfully create runtime pages which users can use to associate, start and action workflows in SharePoint.

Adding the generated web site to the solution explorer will give users the abilty to customize the page controls such that they fulfill requirements to execute the workflow in SharePoint.

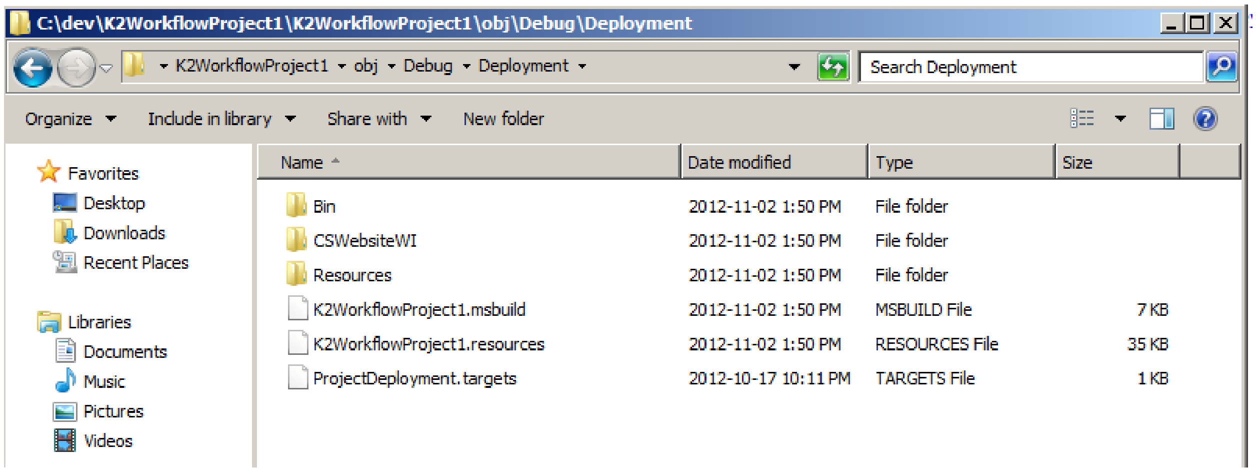
The controls generated are pre-compiled which means users are permitted to change the look, styling and functioning of the pages prior to deployment package creation. The wizard will generate the controls. During the deployment phase, host ASP.Net pages are dynamically constructed to hold these controls. The hosting pages that hold the controls will contain the chrome element which will give the pages the SharePoint style.

It is good practice to insure that the website builds during development stages since this could minimize many issues that could arise during the deployment package execution phase that follows on the Production WFE Servers.

Once users have successfully altered and customized the pages to suit the needs and requirements for SharePoint pages they can create a deployment package and can transfer the package to a WFE within the Farm.

STEP 21 – DEPLOY SHAREPOINT WORKFLOW INTEGRATED PROCESS

Once a deployment package has been created for the K2 SharePoint Workflow Integration process it will reside within the root folder of the K2 Project under “obj\debug”. This deployment folder holds all the required files needed for successful package deployment to the SharePoint Farm.



Name	Date modified	Type	Size
Bin	2012-11-02 1:50 PM	File folder	
CSWebsiteWI	2012-11-02 1:50 PM	File folder	
Resources	2012-11-02 1:50 PM	File folder	
K2WorkflowProject1.msbuild	2012-11-02 1:50 PM	MSBUILD File	7 KB
K2WorkflowProject1.resources	2012-11-02 1:50 PM	RESOURCES File	35 KB
ProjectDeployment.targets	2012-10-17 10:11 PM	TARGETS File	1 KB

Of the files included in the deployment package are the pre-compiled workflow controls. When the package is transferred to the WFE for deployment and execution occurs the following K2 deployment tasks are performed:

1. ASP.Net controls are compiled and deployed

   The ASP.Net controls are the first to be processed and there are certain criteria that needs to be met on the WFE system before these controls can be successfully deployed.

   The following components will be needed on the SharePoint WFE before the pre-compiled pages can be deployed to the system.

- Windows SDK (Runtime components only)
- Visual Studio Web Deployment Projects

Once the controls have compiled, their binaries will reside in the deployment packages “CSWebsiteWI” folder. The compiled controls and assemblies will reside within the “publish” folder and it will be these files that will be transferred to the WFE.

When the controls are built successfully, they will propagate to a single WFE. Depending on which WFE is targeted the pages will reside within the Layouts in the SharePoint Hive. Each control will have their own hosted aspx pages generated.

Successfully generated and deployed aspx pages and controls will reside in:

SP2007
12\TEMPLATE\LAYOUTS\k2blackpearlpages

SP2010
14\TEMPLATE\LAYOUTS\k2blackpearlpages

Successfully generated and deployed ascx controls will reside in:

SP2007
12\TEMPLATE\LAYOUTS\k2blackpearlpages\controls

SP2010
14\TEMPLATE\LAYOUTS\k2blackpearlpages\controls

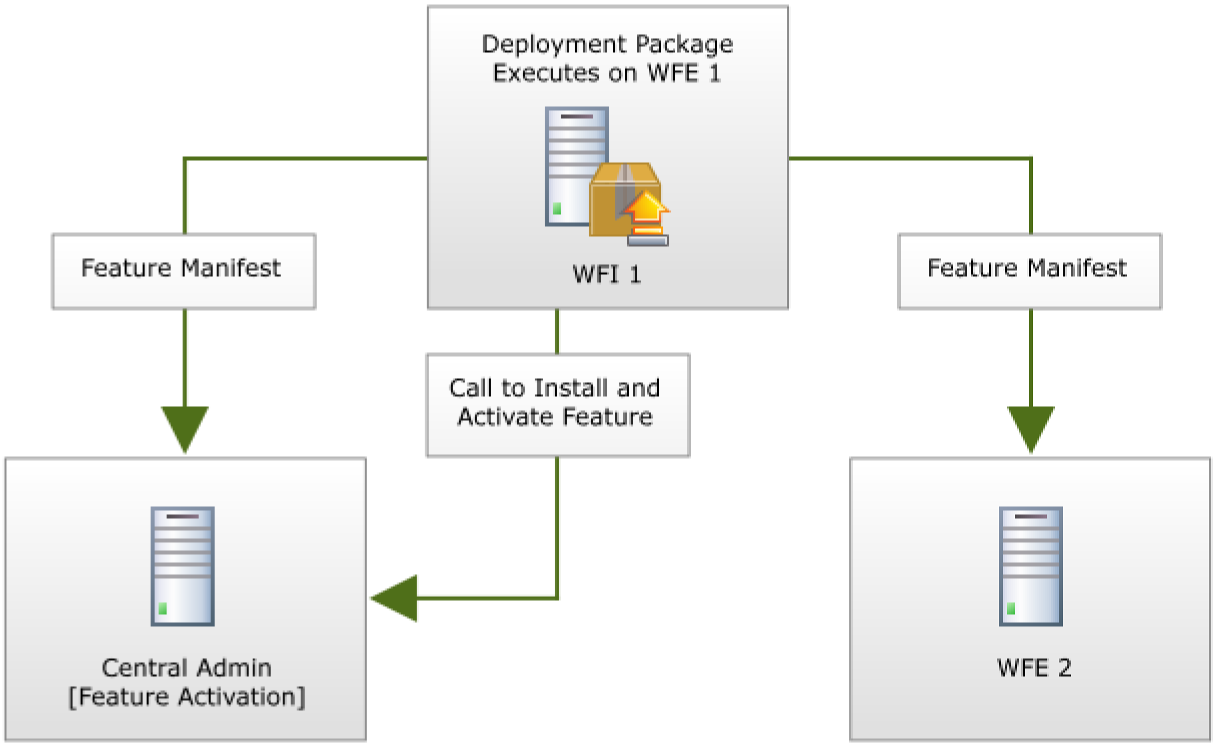
Once the hosting aspx pages have been created and generated they can be manually altered by users for more customized style fitting to the business requirements.

The generated pages and controls do not get propagated to all the Servers in the Farm. Due to this fact network administrators will need to distribute the resultant binaries, pages and controls to the remaining WFE's in the Farm. As indicated above the files reside within the SharePoint Hive under layouts and will need to be copied to the same location on the rest of the WFE's.

2. SharePoint Workflow Feature Installation and Activation

With each deployment of custom pages an associated feature is created, installed and activated on the farm. During the execution of the deployment package feature, manifest files are created and dropped in the **Features** folder under the **SharePoint Hive**. The name for the given feature is in the format {K2PROJECTNAMEPROCESSNAME}. This folder will hold the feature.xml and workflow.xml manifest files.

During the deployment the K2 workflow integration task will connect to a WFE from which point all manifests will propagate to the rest of the WFE's and CA systems in the farm.

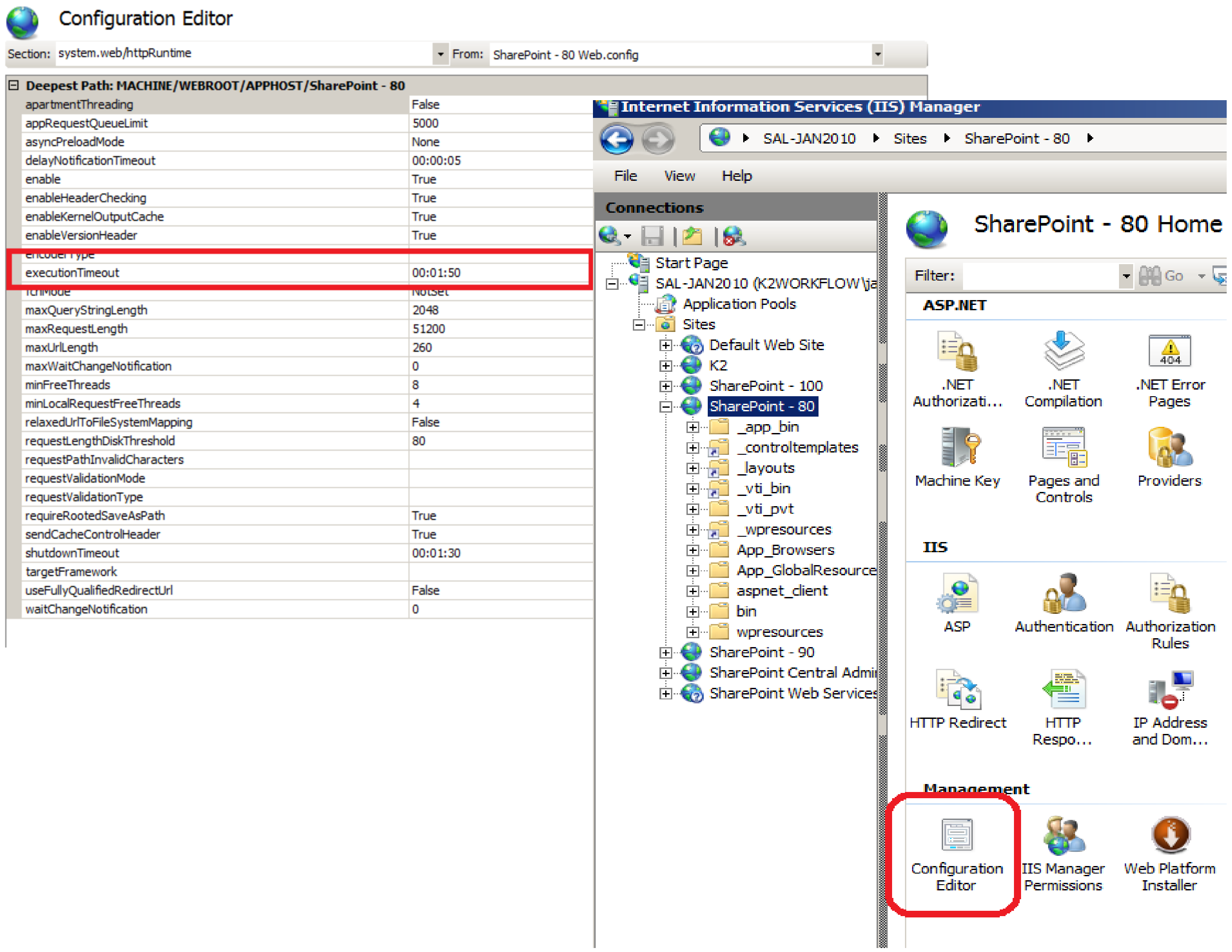

Deployment Package Executes on WFE 1
WFI 1
Feature Manifest
Feature Manifest
Call to Install and Activate Feature
Central Admin [Feature Activation]
WFE 2

Note should be taken at this point that during this stage there are numerous web service calls performed throughout the farm in order to propagate the files needed for successful feature install. The final step is a call from the WFE to the Central Admin server from which a feature install and activation occurs.

Note: If permissions are not correct or there are network related issues which prohibit the WFE to successfully contact the WF's or CA's throughout the farm it might occur that feature manifests are not propagated throughout the Farm. In this case network administrators will need to copy the newly created feature folder manually to WFE's and CA systems in the Farm.

Once the feature installation and activation has occurred, associations are possible on the SharePoint site.

In many cases the execution of the deployment can take a long time and it is not uncommon to run into timeout issues when deploying a Workflow Integrated process. One can get past this by increasing the runtime timeout in the web config.



Configuration Editor

Section: system.web/httpRuntime From: SharePoint - 80 Web.config

Deepest Path: MACHINE/WEBROOT/APPHOST/SharePoint - 80

apartmentThreading	False
appRequestQueueLimit	5000
asyncPreloadMode	None
delayNotificationTimeout	00:00:05
enable	True
enableHeaderChecking	True
enableKernelOutputCache	True
enableVersionHeader	True
encoderType	
executionTimeout	00:01:50
fcnMode	NotSet
maxQueryStringLength	2048
maxRequestLength	51200
maxUrlLength	260
maxWaitChangeNotification	0
minFreeThreads	8
minLocalRequestFreeThreads	4
relaxedUrlToFileSystemMapping	False
requestLengthDiskThreshold	80
requestPathInvalidCharacters	
requestValidationMode	
requestValidationType	
requireRootedSaveAsPath	True
sendCacheControlHeader	True
shutdownTimeout	00:01:30
targetFramework	
useFullyQualifiedRedirectUrl	False
waitChangeNotification	0

Permissions need to be taken into consideration when performing a deployment of workflow integration with custom pages since the account under which the deployment occurs must be able to establish a secure web service connection. This becomes critical when working with systems in which SSL has been setup. If certificates are not correct in the environment one can hit many authentication errors when performing a deployment.

During a deployment a detailed log of events is registered on the primary WFE that performs the deployment. This log can be referenced to troubleshoot any connection problems or failed deployment.

3. SharePoint Hidden list updates

The final task to execute is the population of the K2 Hidden SharePoint list. This list resides in the root of the target site and holds all the integration data needed to successfully start a K2 workflow integration process when a SharePoint Workflow is started.

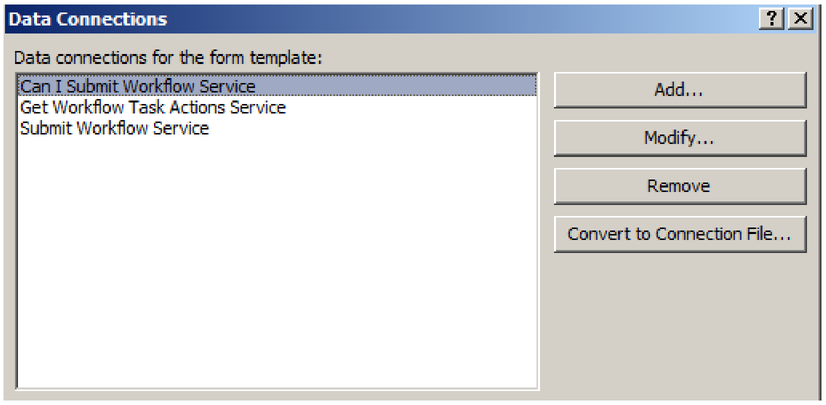
INFOPATH INTEGRATED WORKFLOW

STEP 22 – PREPARE PROCESS

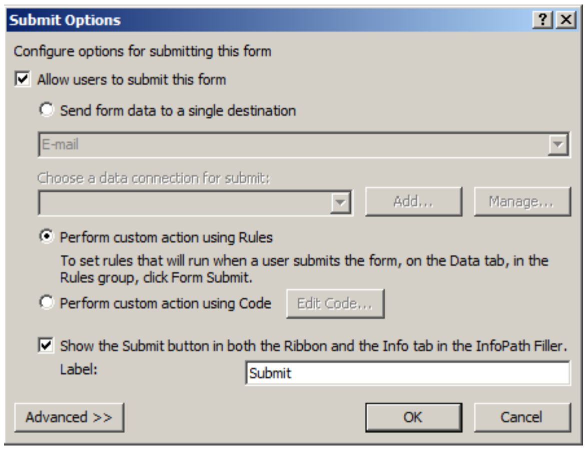
There are a few best practices one can follow and certain constraints one should be aware of when deploying a K2 integrated InfoPath form. Prior to the creation of a deployment package and carrying out a production deployment one needs to take the following into consideration:

Submit Rule

When the integration occurs for an XSN, K2 creates a data connection called "Submit Workflow Service". This data connection is used to perform the submission of the form when the user clicks the submit button. It is not advisable to change the submit rules in any way since K2 will be unable to successfully perform the execution needed to submit the form and start the K2 workflow.

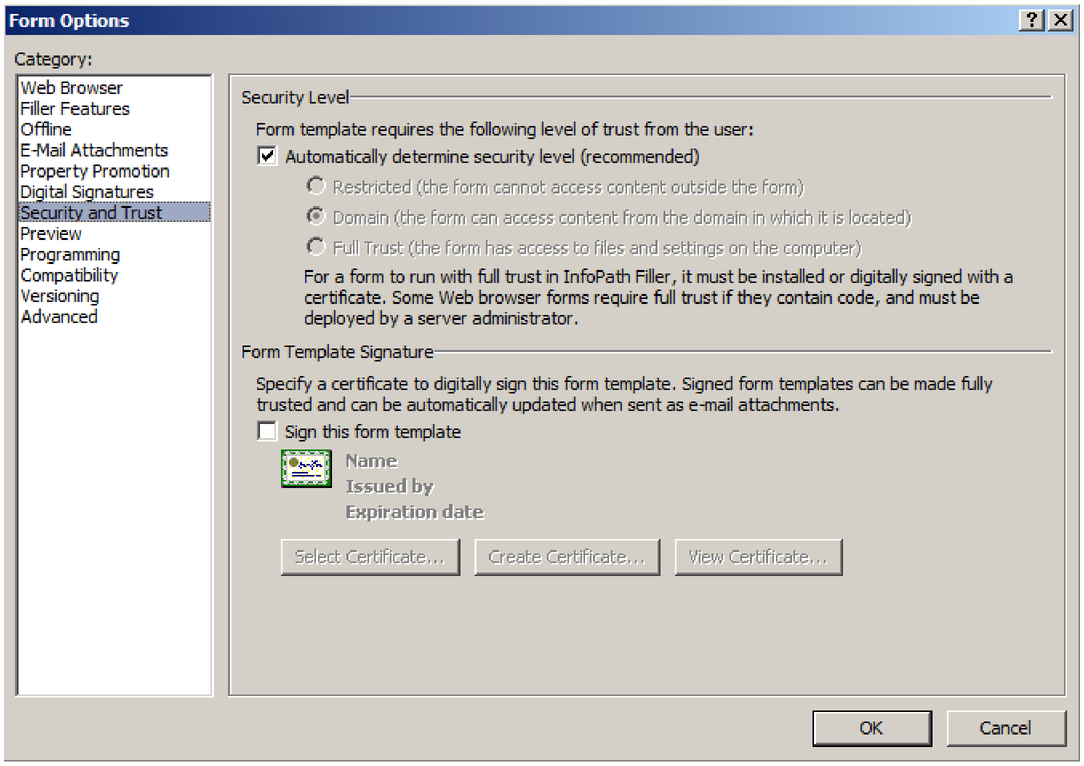

Data Connections
Data connections for the form template:
Can I Submit Workflow Service
Get Workflow Task Actions Service
Submit Workflow Service
Add...
Modify...
Remove
Convert to Connection File...

The default rule selected in the submit options is "Perform custom action using Rules". Users should not change this to any of the available options in the "Submit Option" dialog.


Submit Options
Configure options for submitting this form
Allow users to submit this form
Send form data to a single destination
E-mail
Choose a data connection for submit:
Add...
Manage...
Perform custom action using Rules
To set rules that will run when a user submits the form, on the Data tab, in the Rules group, click Form Submit.
Perform custom action using Code
Edit Code...
Show the Submit button in both the Ribbon and the Info tab in the InfoPath Filler.
Label:
Submit
Advanced >>
OK
Cancel

Form Template Signature

During the publishing phase of the XSN to a SharePoint Form library, any certificates configured on the form will be lost. Users that have configured certificates on the InfoPath form need to realize this and make provision for it by updating the form after publishing has occurred. Users can reconfigure the certificate on the form by downloading the form after the form has been published, re-adding the certificate and then republishing the form to the Form library.



Security Level

Form template requires the following level of trust from the user:

[✓] Automatically determine security level (recommended)

( ) Restricted (the form cannot access content outside the form)

(●) Domain (the form can access content from the domain in which it is located)

( ) Full Trust (the form has access to files and settings on the computer)

For a form to run with full trust in InfoPath Filler, it must be installed or digitally signed with a certificate. Some Web browser forms require full trust if they contain code, and must be deployed by a server administrator.

Form Template Signature

Specify a certificate to digitally sign this form template. Signed form templates can be made fully trusted and can be automatically updated when sent as e-mail attachments.

[ ] Sign this form template

Name

Issued by

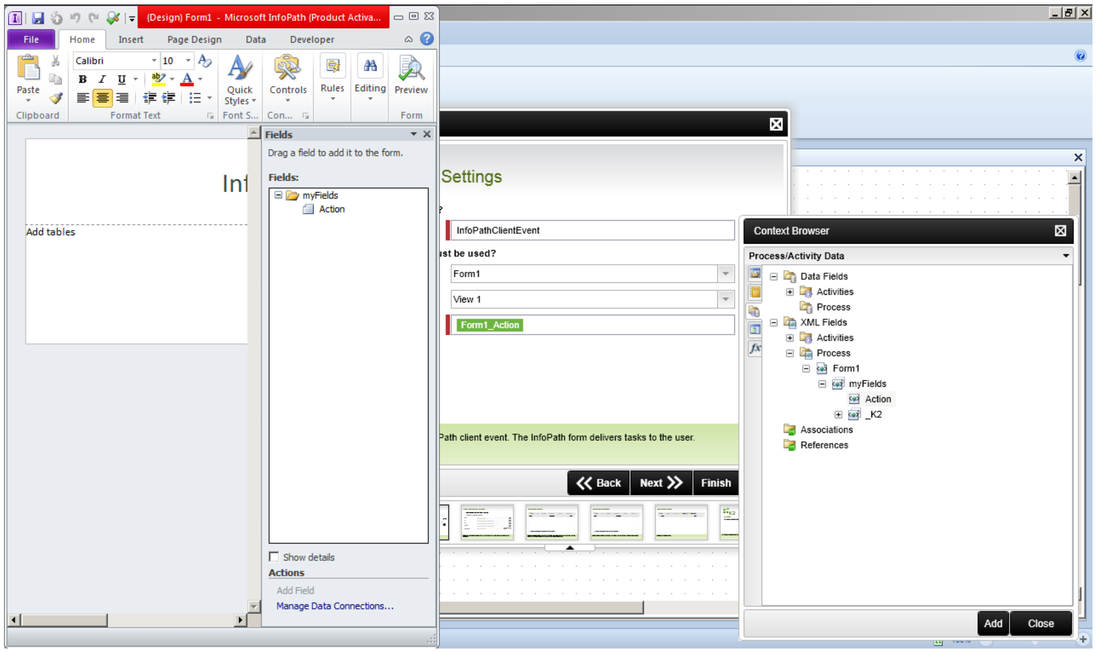
Expiration date

Custom Code

SharePoint InfoPath integration does not support custom code for InfoPath forms deployed via K2. Should a user have custom code for their form this code will not function after deploying the package to the production environment.

Client Event action field

During the design phase of the InfoPath intergrated process one must supply an action field on the general event settings page.



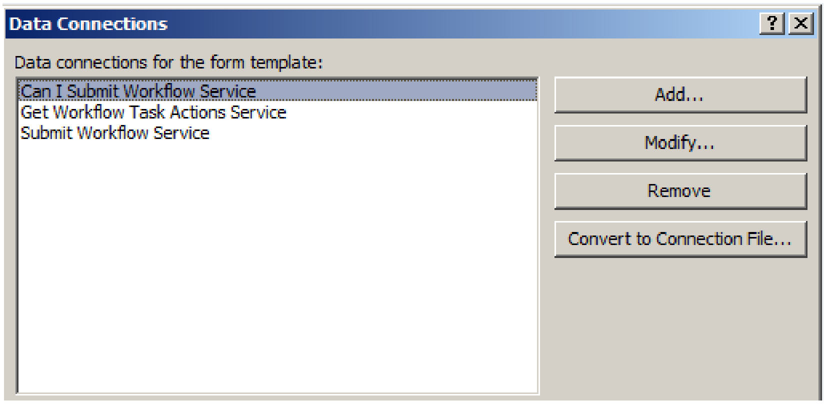
Users must create a valid field in the design of the InfoPath form prior to integration in order to map this field to the action field in the wizard. This field will be used during execution to map the action selected by the user when deployment occurs.

K2 Data Connections

Some of the data connections added to the InfoPath form during a K2 InfoPath Integration are:

“Can I Submit Workflow Service”
“Get Workflow Task Actions Service”
“Submit Workflow Service”

It is best practice to not alter these data connections in any way via updating values through design or InfoPath manifest editing or through republishing the form after process deployment has occurred. These data connections are necessary for the form to correctly execute and integrate with the K2 process.


Data Connections
Data connections for the form template:
Can I Submit Workflow Service
Get Workflow Task Actions Service
Submit Workflow Service
Add...
Modify...
Remove
Convert to Connection File...

STEP 23 – DEPLOY INFOPATH INTEGRATED PROCESS

Once a package has been created the process is ready for deployment. One can move the process over to production and perform the deployment. Whilst performing the deployment of the Infopath package it is important to note that all SmartObject connection strings and webservice urls wil be updated to those specified by the environment.

Browser Enabled Forms

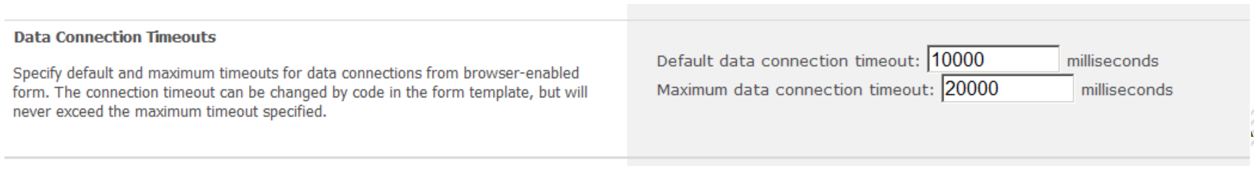
Deploying the process to the production environment in which the InfoPath forms target a SharePoint Form Library requires the InfoPath form to be published to the Form Library. This process involves two primary actions. The first is to publish the form to the library, and the second, should the user have specified the forms to be browser enabled, is to browser enable the forms. These actions during deployment require the user to have specific rights in the SharePoint Site Collection to perform this action successfully. Minimum rights for this requires design rights on site level and users should ensure that the deploying user account adheres to this.

Data Connection Timeouts

During the execution of browser based or thick client filler forms it might occur that data connections takes longer than the usual period to complete. In the case where this occurs one might have to increase Data Connection Timeout setting in SharePoint Central Administration of the SharePoint Farm.

To increase Data Connection Timeouts, follow the steps below:

1. Navigate to the SharePoint Central Administration site on the SharePoint Server Farm.
2. Navigate to the General Settings page.
3. Click on the “Configure InfoPath Forms Services” link.
4. When the page opens, locate the Data Connection Timeouts section and increase the default data connection timeout value.



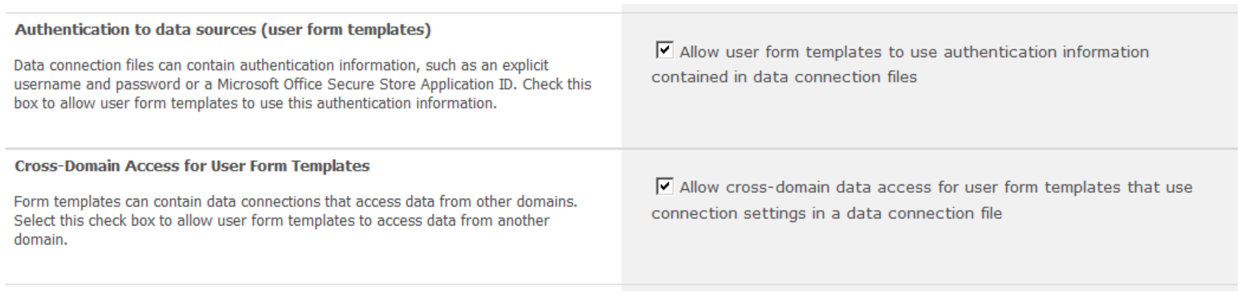
Data Connection Timeouts

Specify default and maximum timeouts for data connections from browser-enabled form. The connection timeout can be changed by code in the form template, but will never exceed the maximum timeout specified.

Default data connection timeout: 10000 milliseconds
Maximum data connection timeout: 20000 milliseconds

InfoPath Forms Services Settings

The following settings needs to be correctly configured in the SharePoint Central Administration site before InfoPath thin client forms can be successfully executed.



Authentication to data sources (user form templates)

Data connection files can contain authentication information, such as an explicit username and password or a Microsoft Office Secure Store Application ID. Check this box to allow user form templates to use this authentication information.

☑ Allow user form templates to use authentication information contained in data connection files

Cross-Domain Access for User Form Templates

Form templates can contain data connections that access data from other domains. Select this check box to allow user form templates to access data from another domain.

☑ Allow cross-domain data access for user form templates that use connection settings in a data connection file

Authentication to data sources (user form templates) check box must be selected as well as Authentication to data sources (user form templates) in the InfoPath Forms Services Setting.

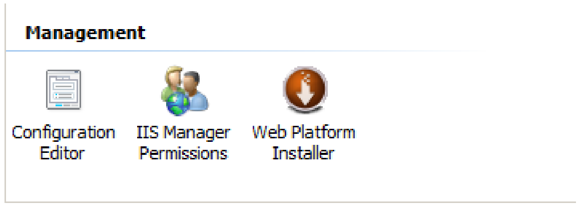
SmartObject deployment connection timeouts

Depending on how many SmartObject data connections have been configured on the InfoPath form, it might occur that users experience timeouts when deploying to the production environment. These timeouts are visable in the command window as a red error message when performing a deployment of the package or in the development environments error dialog.

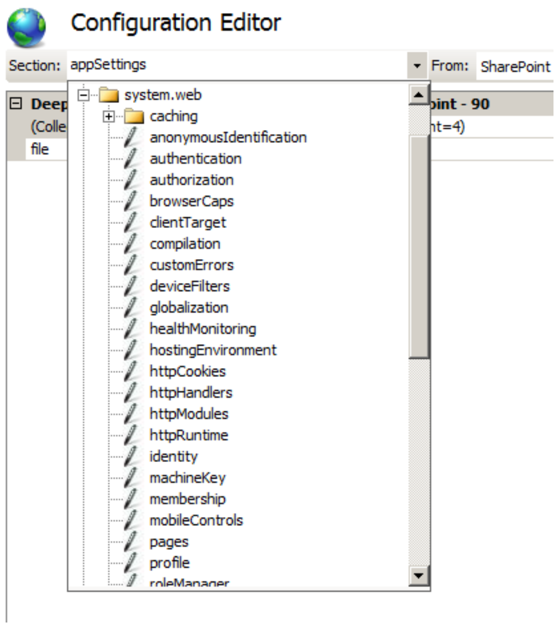
A way to stop these timeouts is to increase the HttpRuntime executionTimeout value on the SharePoint Web Front Ends for the targeted website. These settings can be changed by following the steps below:

On all the SharePoint Web Front Ends:

1. Click Start, Run, type in “inetmgr” and click OK.
2. Expand the **Server** node under connections pane to the left of the window that opens.
3. Expand **Sites** node under the Server node.
4. Locate the **Website** to which you are deploying to from the list of websites.
5. Click the **Server** and you will notice the main window panel will update to show all the websites settings.
6. Find **Configuration Editor** under the **Management** category in the main window settings page of the website.



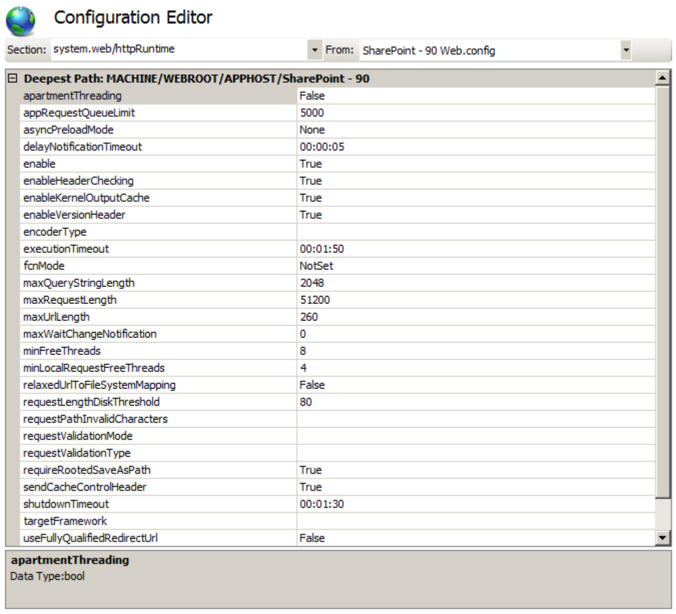
7. Click the Configuration Editor and a Configuration Editor window will open.
8. In the “Section” dropdown at the top of the window click and locate “system.web\httpRuntime” node.



Section: appSettings

From: SharePoint

9. When the main window changes, locate the **executionTimeout** value and change it to a larger value.



Section: system.web/httpRuntime

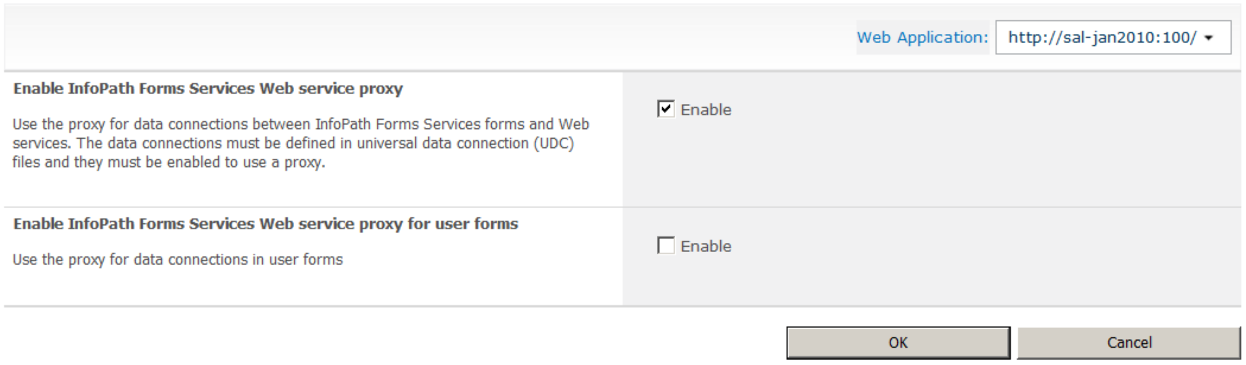
From: SharePoint - 90 Web.config

Deepest Path: MACHINE/WEBROOT/APPHOST/SharePoint - 90	
apartmentThreading	False
appRequestQueueLimit	5000
asyncPreloadMode	None
delayNotificationTimeout	00:00:05
enable	True
enableHeaderChecking	True
enableKernelOutputCache	True
enableVersionHeader	True
encoderType	
executionTimeout	00:01:50
fcnMode	NotSet
maxQueryStringLength	2048
maxRequestLength	51200
maxUrlLength	260
maxWaitChangeNotification	0
minFreeThreads	8
minLocalRequestFreeThreads	4
relaxedUrlToFileSystemMapping	False
requestLengthDiskThreshold	80
requestPathInvalidCharacters	
requestValidationMode	
requestValidationType	
requireRootedSaveAsPath	True
sendCacheControlHeader	True
shutdownTimeout	00:01:30
targetFramework	
useFullyQualifiedRedirectUrl	False

apartmentThreading
Data Type:bool

Claims Based Authentication

In the case where users are working on a site that has Claims Based Authentication, one needs to Configure InfoPath Forms Services Web Service Proxy. This can be done through the SharePoint Central Adminstration site for the SharePoint Server farm and can be located under Central Administration> General Application Settings> Configure InfoPath Forms Services Web Service Proxy.


Web Application: http://sal-jan2010:100/

Enable InfoPath Forms Services Web service proxy

Use the proxy for data connections between InfoPath Forms Services forms and Web services. The data connections must be defined in universal data connection (UDC) files and they must be enabled to use a proxy.

☑ Enable

Enable InfoPath Forms Services Web service proxy for user forms

Use the proxy for data connections in user forms

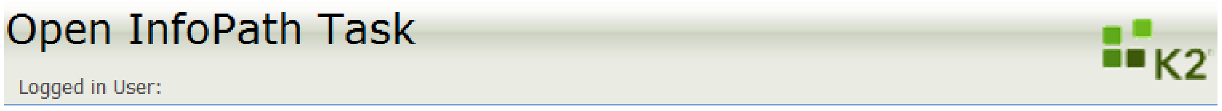
☐ Enable

Users need to specify the correct Web Application in the top right corner of the page when they load the page. This will be the web application on which the claims based forms authentication has been configured. Once the web application has been selected, users need to tick the “Enable InfoPath Forms Services Web service proxy” check box and submit the page via clicking the OK button.

For Claims Based Forms Authentication InfoPath makes use of a Forms Service Web service proxy in order to correctly validate users on forms submission. Without this correctly configured for web applications, InfoPath execution at runtime will fail for K2 InfoPath Integrated processes.

OpenInfoPathTask.aspx

In order to open InfoPath task pages, users need to be able to access an OpenInfoPathTask.aspx which resides on all SharePoint web front ends in the “http://{site}/_layouts/K2/RuntimeServices” folder. Full path to this page is http://{site}t/_layouts/K2/RuntimeServices/OpenInfoPathTask.aspx.



This page forms part of K2 runtime components and its function is to serve as an intermediary page to redirect users to InfoPath forms used to action worklist items. The page will open an InfoPath form supplied during the K2 process design and allow users to successfully submit their tasks.

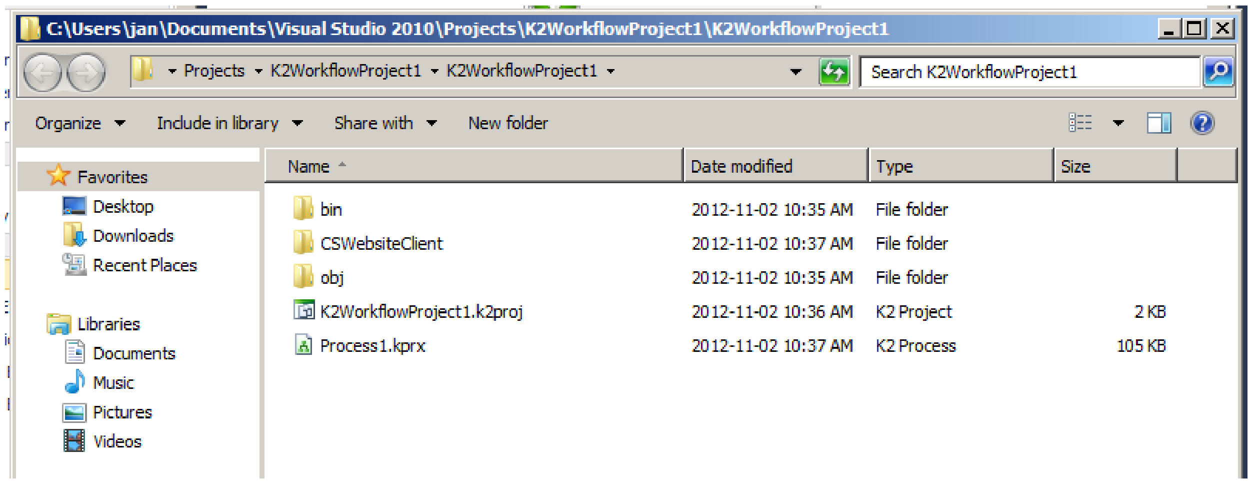
The required permissions to open this page is at the least view rights for the user on web level. Without this permission level users will be unable to access their InfoPath tasks.

ASP.NET

ASP.NET or Silverlight based UI deployment is done manually by creating a new web-site, copying the web-application resources and updating configuration settings. Refer to KB000523.

STEP 24 – PREPARE ASP.NET PROCESS

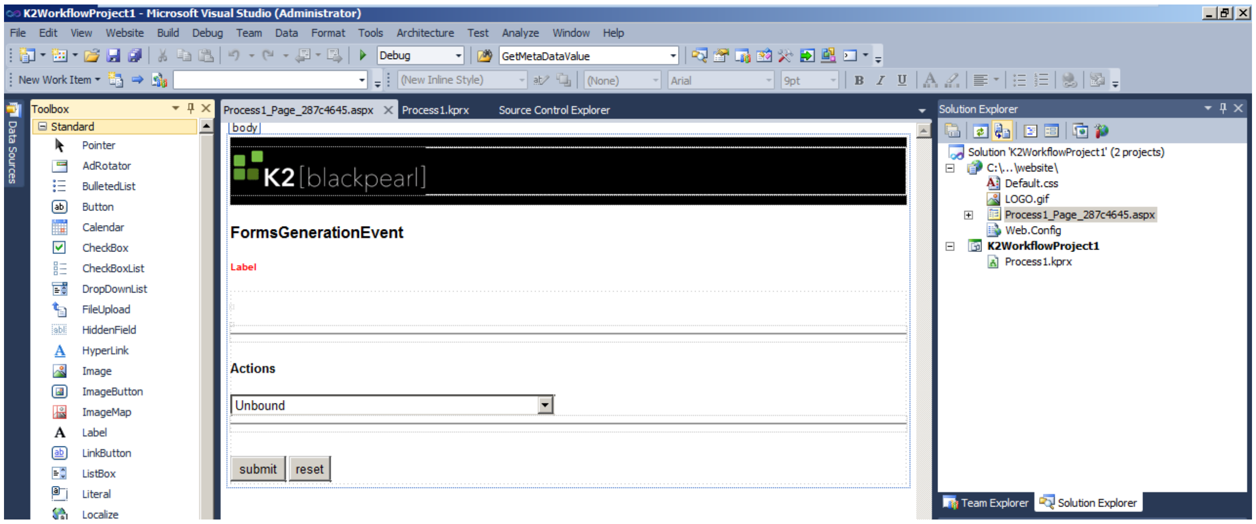
Once you have run through the Forms Generation Client Event the generated website will reside in the root of the K2 Project Folder with the name "CSWebsiteClient".



C:\Users\jan\Documents\Visual Studio 2010\Projects\K2WorkflowProject1\K2WorkflowProject1

Name	Date modified	Type	Size
bin	2012-11-02 10:35 AM	File folder	
CSWebsiteClient	2012-11-02 10:37 AM	File folder	
obj	2012-11-02 10:35 AM	File folder	
K2WorkflowProject1.k2proj	2012-11-02 10:36 AM	K2 Project	2 KB
Process1.kprx	2012-11-02 10:37 AM	K2 Process	105 KB

It is at this point that one can add the website to the K2 Solution via the Solution Explorer to further customize the website for use in the project.



K2WorkflowProject1 - Microsoft Visual Studio (Administrator)

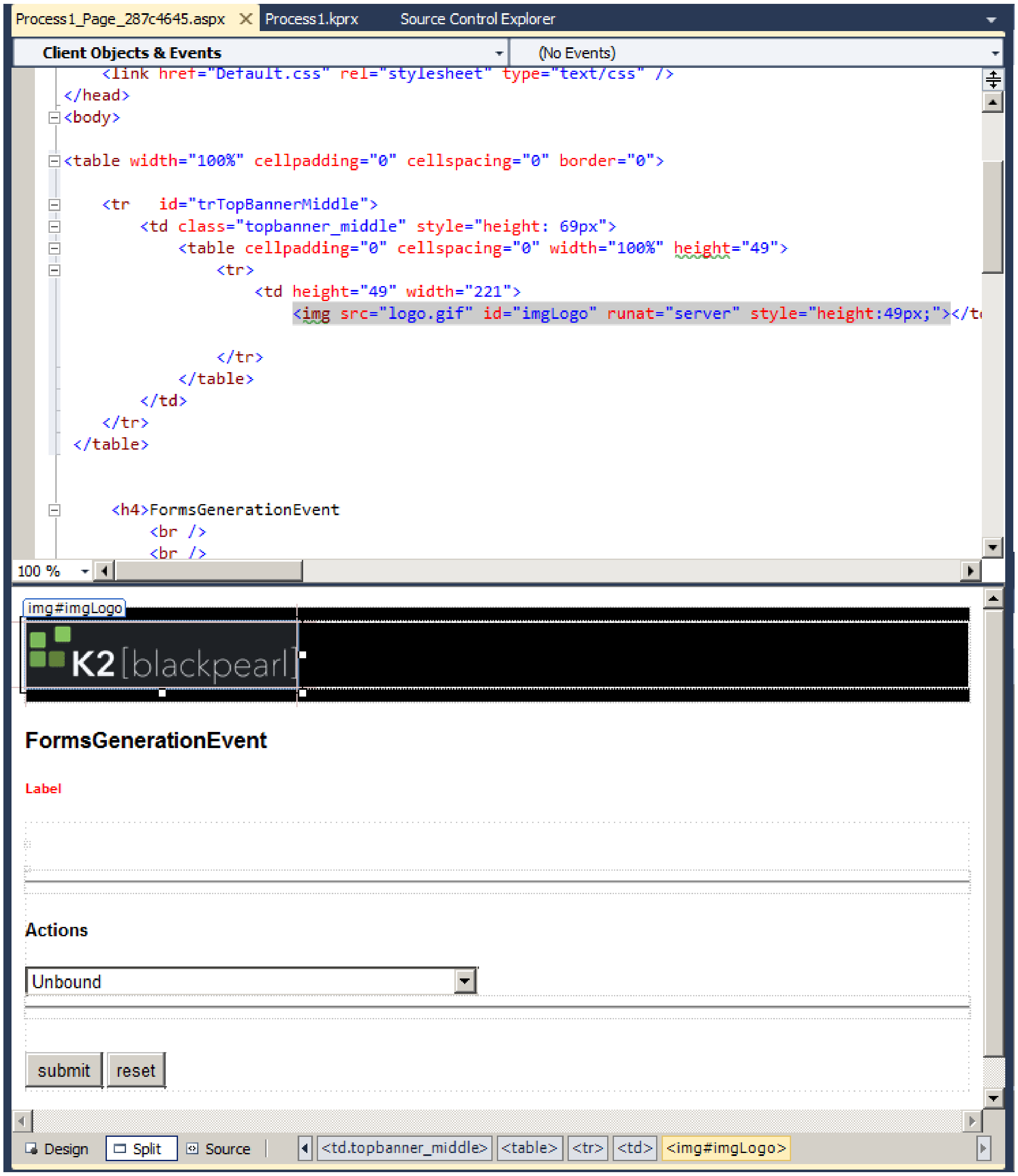
Solution 'K2WorkflowProject1' (2 projects)
- C:\...\website\
  - Default.css
  - LOGO.gif
  - Process1_Page_287c4645.aspx
  - Web.Config
- K2WorkflowProject1
  - Process1.kprx

By dragging .Net controls over from the Visual Studio toolbox and changing page code, one can customize the generated page to fulfill project requirements.

When adding resources such as images and media files ensure that the content resides within the "CSWebsiteClient\ website" folder and that all references are relative to the website directory.

The image below indicates a simple image reference that resides in the same folder as the ASP.net page.

Note that there are no absolute references in this example.

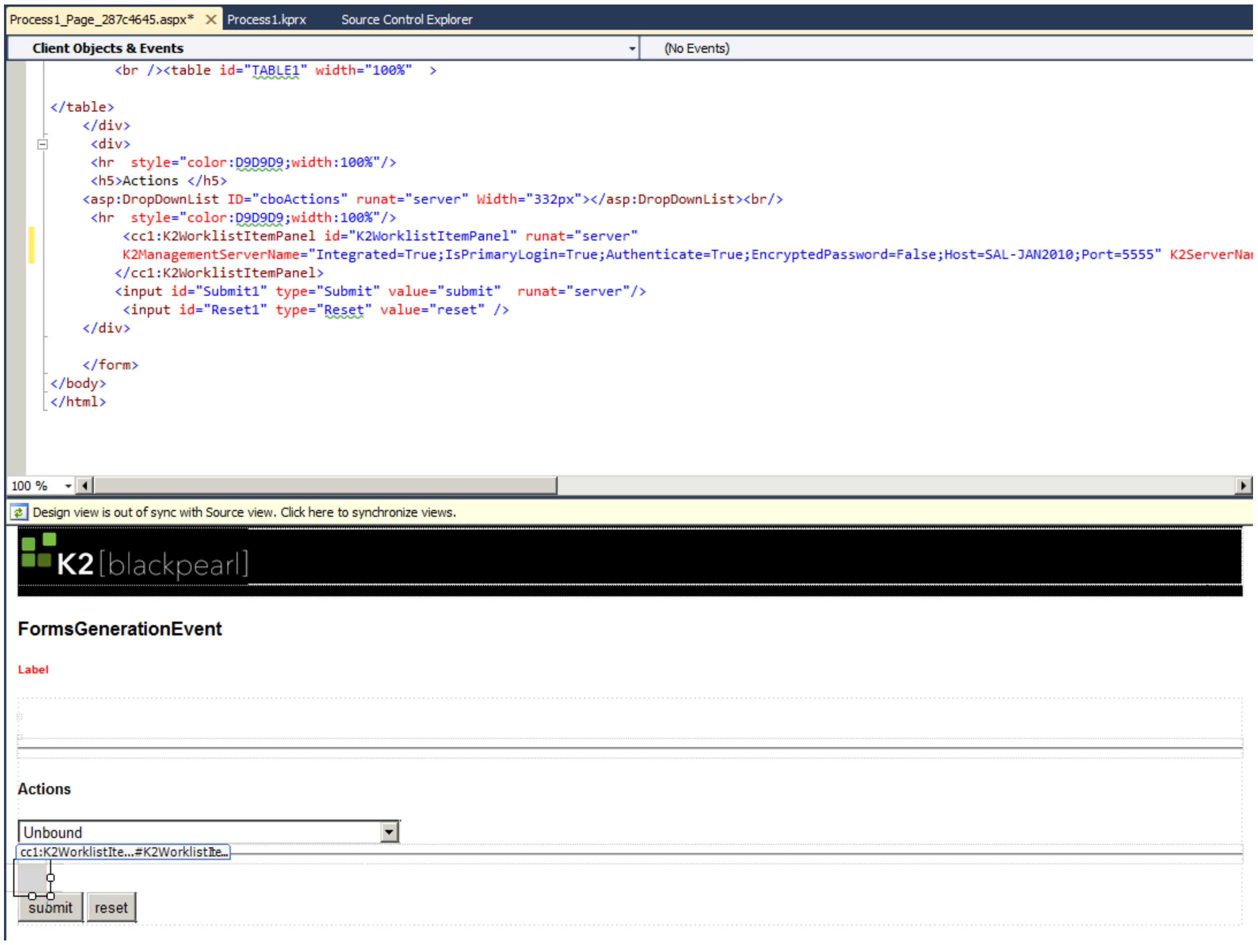


Process1_Page_287c4645.aspx Process1.kprx Source Control Explorer

Client Objects & Events (No Events)

```
        <link href="Default.css" rel="stylesheet" type="text/css" />
    </head>
    <body>

    <table width="100%" cellpadding="0" cellspacing="0" border="0">

        <tr   id="trTopBannerMiddle">
            <td class="topbanner_middle" style="height: 69px">
                <table cellpadding="0" cellspacing="0" width="100%" height="49">
                    <tr>
                        <td height="49" width="221">
                            <img src="logo.gif" id="imgLogo" runat="server" style="height:49px;"></t

                    </tr>
                </table>
            </td>
        </tr>
     </table>


         <h4>FormsGenerationEvent
             <br />
             <br />
```

The Forms Generation Client Event makes use of a K2WorklistItemPanel control which is used during execution to retrieve and action K2 worklist items. One can see below in the picture that when the page is generated the wizard creates an ASP.Net control with a K2ManagementServerName field that is set to the value of the K2 Host Server Connection String. It is this connection string that will permit the panel to retrieve and action worklist items when the page is opened by the user.



```
        <br /><table id="TABLE1" width="100%"   >

</table>
    </div>
     <div>
     <hr  style="color:D9D9D9;width:100%"/>
     <h5>Actions </h5>
    <asp:DropDownList ID="cboActions" runat="server" Width="332px"></asp:DropDownList><br/>
     <hr  style="color:D9D9D9;width:100%"/>
         <cc1:K2WorklistItemPanel id="K2WorklistItemPanel" runat="server"
         K2ManagementServerName="Integrated=True;IsPrimaryLogin=True;Authenticate=True;EncryptedPassword=False;Host=SAL-JAN2010;Port=5555" K2ServerNa
        </cc1:K2WorklistItemPanel>
        <input id="Submit1" type="Submit" value="submit"  runat="server"/>
         <input id="Reset1" type="Reset" value="reset" />
    </div>

    </form>
</body>
</html>
```

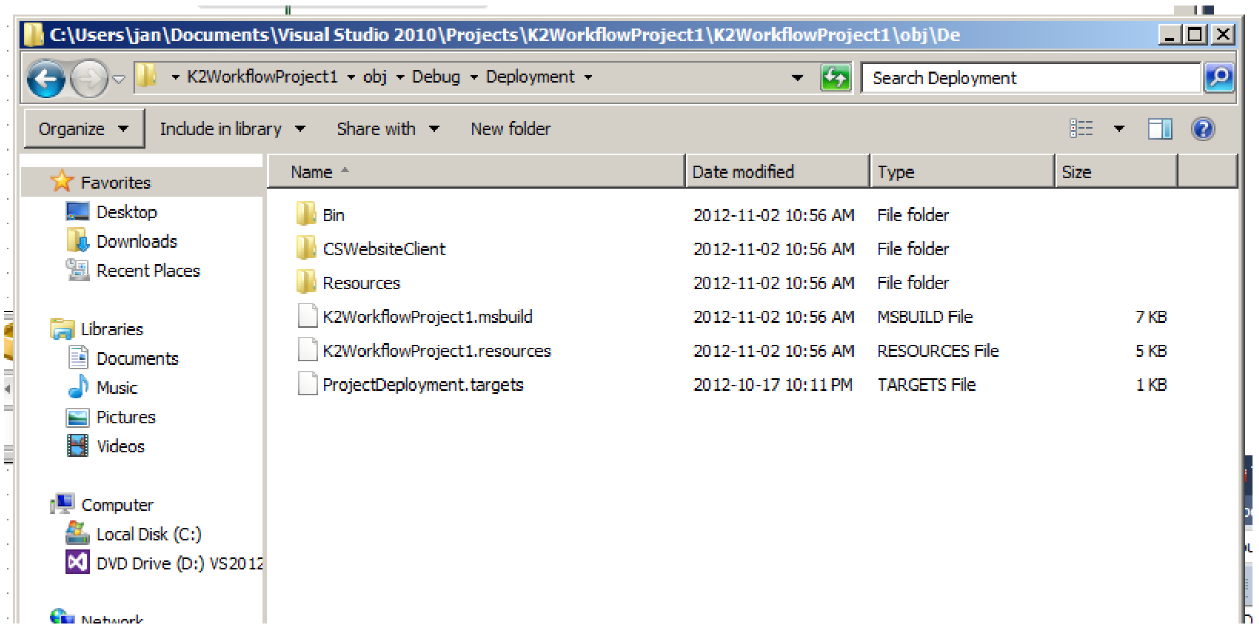
Design view is out of sync with Source view. Click here to synchronize views.

The value of K2ManagementServerName does not update once the wizard has completed the page generation and it is thus very important that should the K2ManagementServerName vary between production and development that the user change this value to reflect the correct connection string pre-package deployment creation.

It is good practice to always add the website into the solution explorer and ensure that the website builds successfully. This can help avoid unnecessary debugging and troubleshooting when one eventually deploys on the Production Server.

STEP 25 – DEPLOY ASP.NET PROCESS

Once a deployment package is created for the project, the “CSWebsiteClient” folder will be copied into the folder that holds the deployment package files. The deployment package folder will reside inside a folder called “obj\debug” in the K2 Project folder.



Name	Date modified	Type	Size
Bin	2012-11-02 10:56 AM	File folder	
CSWebsiteClient	2012-11-02 10:56 AM	File folder	
Resources	2012-11-02 10:56 AM	File folder	
K2WorkflowProject1.msbuild	2012-11-02 10:56 AM	MSBUILD File	7 KB
K2WorkflowProject1.resources	2012-11-02 10:56 AM	RESOURCES File	5 KB
ProjectDeployment.targets	2012-10-17 10:11 PM	TARGETS File	1 KB

Each time a deployment package is created a copy of the “CSWebsiteClient” folder, which holds the uncompiled ASP.Net pages, will be copied to the deployment package directory.

One must at this stage ensure that K2ManagementServerName is correctly set for the targeted Production environment K2 HostServer.

Once the deployment package is created, ready and one is confident the precompiled pages will build successfully on the Production environment the package can be copied to the Production Workspace Server for deployment.

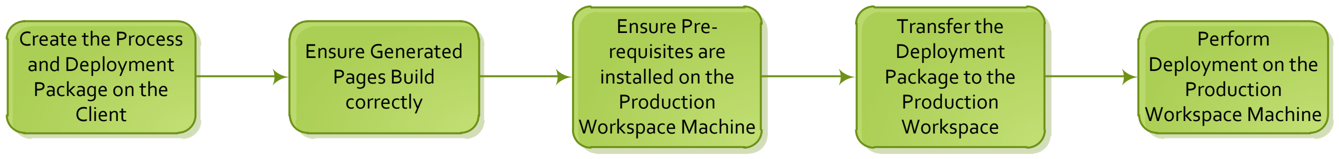
Forms Generated Client Event processes can only be deployed on a Production Workspace Server. One cannot launch the deployment package on a Client, K2 Server or SharePoint WFE and expect the website to successfully deploy. The deployment package MUST execute on the Production Workspace Server otherwise the website will not deploy.

It is important to understand that when the package is transferred to the Production Workspace server that the Generated Pages are in Pre-Compiled state. This means the pages have not yet compiled and in order for this to occur one needs a few pre-requisites installed on the Production Workspace Server.

- Windows SDK (Runtime components only)
- Visual Studio Web Deployment Projects

Note: More information will be added for Web Deployment Projects

Working with the deployment package will occur in the following order:


Create the Process and Deployment Package on the Client
Ensure Generated Pages Build correctly
Ensure Pre-requisites are installed on the Production Workspace Machine
Transfer the Deployment Package to the Production Workspace
Perform Deployment on the Production Workspace Machine

Step 26 – K2 SmartForms

Create a package using K2 Package and Deployment Beta. To make use of the K2 Package and Deployment component sign up for the current Beta program at beta@k2.com

STEP 27 - ASSIGN K2 SECURITY

Assign permissions for example SmartObjects, Workflows or K2 smartforms. This includes the K2 Server permissions assigned in K2 Workspace>Management Console as well as process rights which can also be set in K2 Workspace>Management Console. These rights can be set in the K2 Process Portal if that is the preference.

SPECIFIC PROCESS / SMARTOBJECT / FORMS DEPLOYMENT

SmartObjects and processes can continuously change. These changes will first be completed and tested on the development environment. The package deployment option within K2 Designer for Visual Studio or K2 Studio can then be used to deploy these changes to the production environment.

Existing running process instances will continue to use the previous version deployed of the Process. Only after a new process instance was started, will the new version of the Process be used.

Deployed processes will always use the most current SmartObject.

SUMMARY

Deploying K2 blackpearl artifacts between environments can be pain-free if it is planned for in advance. Deployment packages make the most common artifacts, SmartObjects and Processes, easy to move between environments. Other artifacts which change less frequently can either be configured by an administrator in a new environment, or they can be scripted using some samples found on K2 Underground. An important step to take is to plan for environment migration at the start of a project. All too often developers in their rush to meet a deadline do not plan for migration and it becomes a last minute effort to get it done any way possible. The 'get it done' approach is not repeatable, not documented, and can lead to difficulty in future deployments when no one was sure what was done the first time.

DEPLOYMENT TOOLS

Below are a list of tools available in K2 that can be used to deploy different K2 artifacts.

- MSBuild
- SmartObject import and export

- Exporting of workflows e.g. K2D4SP
- K2 Package and Deployment Beta

OTHER RESOURCES

Also refer to the K2 Installation and Configuration - checklist and guide