K2 ENVIRONMENT LIBRARY
Contents
Introduction ................................................................. 2
Environment Template .................................................. 3
Environment Definitions .............................................. 3
   Single physical K2 environment .............................. 3
   Multiple physical k2 environments ......................... 4
Plugins and Field Types .............................................. 4
Working with K2 Environments ................................. 4
   Viewing K2 Environments ..................................... 4
   Viewing K2 Environment Details ......................... 5
   Updating K2 Environment Details ......................... 5
   Updating K2 Environment Values ......................... 6
   Adding a new K2 Environment ............................ 7
   Deleting a K2 Environment ................................. 8
   Defining New Environment Template Fields ............ 8
   Environment Permissions ................................... 9
Environment Values and the String Table .................. 10
   String Table step-by-step scenario .................... 10
      Viewing the String Table on a K2 Server .......... 12
   Interacting Programmatically with Environment and String Table values .......... 13
Optimizing Multiple K2 Environments .................. 14
   Updating the K2 Object Browser in Visual Studio ... 16
   Changing Servers in Visual Studio .................... 17
   Changing Environments in Visual Studio .......... 18
   Distributed vs. Consolidated Environment Library – Which is Right? ... 18
Packaged deployment .............................................. 19
Appendix ................................................................. 20
   Recommended setup of the Environment Library .... 20
INTRODUCTION

The K2 Environment Library is a collection of variables describing connection settings to K2 services and servers. These variables are grouped by environment (i.e. Production or Development) which are, in turn, grouped by Templates. All this data is stored within the K2 database.

This document is meant to educate the user regarding the concept of the Environment Library. For in-depth information on how to interact with the Environment Library, please utilize the links within the ‘Packaged Deployment’ section of this document.

The term ‘Environment’ is a familiar one in software development. Best practice in business-application development requires that a solution be developed within a development environment, tested within a test environment, and once approved, deployed to a production environment.

Typically, a solution is developed on a Development computer, belonging to the Development Environment. There is direct connectivity between this computer (the client) and the Development Environment (server). When development work is complete, the solution is deployed to the QA/Test, or Staging environment, where it is subjected to a battery of functionality and user-acceptance tests. Finally, when the solution is ready for the ‘real world’, it is packaged and deployed to the Production Environment server. There is often no direct connectivity between Development computers (client) and the Production Server.

Once a solution is packaged, K2 uses values from environment fields when deploying artifacts to the target environment. Wherever a package is created by a user, configuration settings called environment fields are utilized in lieu of fixed values. Environment fields are commonly used for such things as the mail server location, workflow server location, and the SharePoint Server URL. These Environment field values are then populated to the target environment during deployment, and stored in the K2 database.

As of the release of K2 blackpearl 4.6, a single database is used. Prior to K2 blackpearl 4.6, multiple databases were present with the environment details stored in the EnvironmentSettings database. Environment information can be viewed and modified using the K2 Object Browser or the K2 Workspace Management Console.

<table>
<thead>
<tr>
<th>Definitions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>K2 Environment Library</td>
<td>A container for templates, plugin types and field types.</td>
</tr>
<tr>
<td>Environment Template</td>
<td>Contains multiple sets of environment definitions.</td>
</tr>
<tr>
<td>Environment Definitions</td>
<td>Contain environment settings for each environment (e.g. Production / Development).</td>
</tr>
<tr>
<td>Environment Field Types, Template Fields</td>
<td>Field-type definitions.</td>
</tr>
<tr>
<td>Environment Fields</td>
<td>Placeholders for K2 server / services connection values.</td>
</tr>
<tr>
<td>Environment Field Value</td>
<td>Connection settings for various K2 servers / services.</td>
</tr>
<tr>
<td>Runtime String Table</td>
<td>Instance values which are used by the K2 Runtime Engine to replace the placeholder Environment fields.</td>
</tr>
</tbody>
</table>
ENVIRONMENT TEMPLATE

K2 provides a Default Template containing environment definitions for Development and Production environments. More templates can be added as needed, for the HR Department’s development, testing and production environments for example.

ENVIRONMENT DEFINITIONS

Each environment definition within an environment template contains a set of fields with values for the Workflow server, Mail server, SharePoint URL and others pertinent to that environment definition (for example, the Development environment).

SINGLE PHYSICAL K2 ENVIRONMENT

In a small implementation, you may have only a single K2 server. In this case, you would host Development and Production processes on the same K2 server. K2 blackpearl allows for this by using the Environment Fields to specify different values for different environments. For example, you could point to a Development and a Production SharePoint URL, separate web-service URLs and so on.

Note: This configuration is not recommended. Best practice is to always have different servers (whether physical or virtual) for different environments.
MULTIPLE PHYSICAL K2 ENVIRONMENTS

A more common implementation of K2 blackpearl involves separate physical or virtual K2 servers for each different environment. A Development environment might consist of a Development K2 server, Development K2 clients, a Development SharePoint server, and so on. This Development environment may be in the same Active Directory domain as the Production environment, or reside within its own domain. The Production environment would then comprise a K2 server farm, a Production SharePoint farm, SQL Reporting Services, and so on. It is common for the Production environment to contain no Development clients, except where one might be reserved for repairing errors which may occur within Production processes.

PLUGINS AND FIELD TYPES

There are several pre-defined Field Types at both Template level and Environment level; more can be added to suit your needs. Additionally, custom fields can be created as needed, such as the base URL for your company’s ASPX forms (most likely a web service URL) or a SharePoint Records Center. Any web service, URL, server name, or other frequently-used information can be stored in an environment field. Custom field types are not usually required; however, should the need arise to create and expose custom field types within the Environment Library Object browser, a custom Plugin needs to be created (see http://help.k2.com/helppages/k2blackpearlUserGuide4.6.5/webframe.html#HowTo-WS_MCon-EnvironmentLibrary.html).

Certain Environment Fields are populated during K2 blackpearl installation and configuration, including the Category, Mail, ServiceObject, SmartObject, Workflow and Workflow Management Servers, and Web Service URL. Other fields such as custom or non-default SharePoint Site URLs can be manually set after installation.

WORKING WITH K2 ENVIRONMENTS

Environment values are placeholders that allow you to deploy your process in different environments, with the placeholders being replaced with the correct values for the target environment at runtime. Environment values are used to simplify process authoring by eliminating the need to update processes every time a resource changes.

For example, you specify a SharePoint site URL for your root SharePoint site. Within a process, you refer to this variable and optionally, a relative path. If you relocate your SharePoint server, you simply update the String Table in K2, and the running version of the process need not change. For a detailed explanation of how Environment values relates to String Tables, refer to the section “Environment Values versus String Tables” within this document.

VIEWING K2 ENVIRONMENTS

Existing environment definitions can easily be viewed within Workspace. Follow these steps to list the Environments:

1. Click on Management, and then Management Console
2. Expand the K2 server node
3. Expand Environment Library
4. Expand Templates
5. Expand Default Template
6. Select Environments

By default, K2 creates both a Development and Production environment definition which you can view in Workspace, as shown in Figure 1. At the time of creation, these two default environments contain identical values for all environment fields.

![Figure 1: Default Environments as seen in K2 blackpearl Workspace.](image)

**VIEWING K2 ENVIRONMENT DETAILS**

If you expand an environment in the left pane of Workspace and select Environment Fields, you see a list of the environment fields currently configured for that environment, an example of which is shown in Figure 2.

![Figure 2: Environment Fields for a Development Environment.](image)

**UPDATING K2 ENVIRONMENT DETAILS**

To update an environment definition, select an environment and click Edit. This opens the properties for the specified environment (as shown in Figure 3). Within this dialog you can change the name and description of the environment as well as whether the environment is set as the default environment. The default environment is selected when you start a deployment wizard. There can only be one default environment.
You can easily modify the current values of any Environment Field. Simply click the radio button next to the value you wish to change, and then click Edit. A dialog box, as shown in Figure 4, allows you to change the description and value for that field. When you update an environment value, it does not take effect until a process that uses it is deployed. Bear in mind that Environment variables are a design-time concept, while the String table is runtime. For more information, see the section of this document entitled ‘Environment values and the String Table’.

You can also add additional Environment Fields using the K2 Workspace. For example, it is necessary to manually specify the SharePoint site URL because it is not automatically configured at installation time. To do this, select the Environment Fields, and then click Add in the right-hand pane. You will see a dialog as displayed in Figure 5 where you will specify the new field. Specify a name for this new field, an optional description, the Field Type (a drop-down list shows all available Field Types), whether it is a default Field, and finally the value of the field.
Figure 5: Adding an environment field, with Field Type drop-down selected.

**Note:** Any changes made to an Environment within Workspace will not be reflected in the K2 Object Browser in Visual Studio until the user selects ‘Refresh’ on the K2 Object Browser (steps to do this in section titled ‘Optimizing Multiple K2 Environments’).

### ADDING A NEW K2 ENVIRONMENT

To create a new Environment, navigate to the Environments node in Workspace. The list of current environments is displayed in the right-hand pane as shown in Figure 1. Click the **Add** button to display the **Add Environment** dialog box as shown in Figure 6.

After typing a unique environment name and optional description, the new environment is immediately displayed in the right-hand pane within Workspace. To get the new Environment node to appear in the left-hand pane of Workspace, you may need to refresh that list. To do so, right-click the Environments node in the left-hand pane, and then click **Refresh**.

Environment names must be unique even across different templates.

When an environment is added, it is based on the Default Template, meaning it has placeholders for each of the Default Template Fields. As you can see in Figure 7, creating a new environment does not set the value for any of the Environment Fields. It is necessary to specify all relevant values for your newly-created environment.
DELETING A K2 ENVIRONMENT

To delete an existing Environment, navigate to the Environments node in Workspace. The list of current environments will show in the right-hand pane. Select the radio button next to the Environment you wish to remove, and then click the **Delete** button.

**Warning:** Clicking **Delete** within the Environments list immediately deletes the environment. There is no verification prompt or method to undo this action.

Be careful when deleting environments. Deleting an environment removes all database entries related to that environment. Even though live processes do not use the Environment Library (they use String Tables instead), you could be removing information which is necessary to developers.

DEFINING NEW ENVIRONMENT TEMPLATE FIELDS

You can also add new Template Fields. Template Fields are available on the default template. To add Fields to the Default Template within Workspace, navigate to the Template Fields node as shown in Figure 8, and then click **Add**. The **Edit Field** dialog box is displayed, as shown in Figure 9. If you specify ‘Set as Default’, this newly-defined Template Field is created in every environment, based on the default template.
ENVIRONMENT PERMISSIONS

By default, all users have full access to all defined environments. Users can select any defined environment from the K2 Object Browser drop-down and can make changes to any fields defined in that environment, including adding and removing fields. Best Practice is to implement a ‘least privileges’ model for the entire K2 server. In other words, you should restrict which user accounts have access to use data, and further restrict which accounts have access to modify that data. To that end, Environment permissions restrict the ways in which given users may interact with Environment Fields.

Environment permissions are set within the Workspace Management Console. Permissions can be set for individual environments or at template level. If you set permissions at the level of the default template, they will be propagated to every environment based on that template.

There are two levels of permissions you can set on environments: ‘Read Only’ and ‘Modify’. As their names imply, Modify permissions grant a user the ability to add or delete fields, and to change field values; Read Only permissions grant a user the ability to make use of that environment from a process-development perspective.
To set permissions on an environment, select it in the left-hand pane of Workspace, and then click **Add** in the right pane. Add the users or groups you want, and grant them ‘Read Only’ or ‘Modify’ permissions as shown in Figure 10. Once you are finished, click **Save**.

*Figure 10: Environment Permissions.*

The following table summarizes the levels of permissions you can set on environments, and the impact it has on associated groups and user accounts.

<table>
<thead>
<tr>
<th>Permission Level</th>
<th>User Rights in K2 Object Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>See Environment</strong></td>
</tr>
<tr>
<td>No Permissions Set</td>
<td>All</td>
</tr>
<tr>
<td>Read Only</td>
<td>X</td>
</tr>
<tr>
<td>Modify</td>
<td>X</td>
</tr>
<tr>
<td>Permissions Set, user not in list</td>
<td>--</td>
</tr>
</tbody>
</table>

Note that there are known issues with permission settings in versions of K2 blackpearl prior to 4.6.5. Please contact your regional support representative if you have concerns.

**ENVIRONMENT VALUES AND THE STRING TABLE**

Process deployment is undertaken in the following sequence:

1. Processes are deployed to a specific environment.
2. The String Table is populated at deployment by values from the specified environment in the Environment Library (the environment: Development, for example).
3. That version of the process is linked to those values.

Should you deploy the same process against ‘Production’, the new version of the process will use the values from the Production String Table entries.

**STRING TABLE STEP-BY-STEP SCENARIO**

1. Process A is built on the dev client:
   - Default Template
   - Development Environment
   - Development environment values
2. Process A is then packaged and deployed on a Production environment. At deployment, the environment values for the Production environment are written to the String Table of the production server.

There is only one String table, but because of the way its columns are used, all processes in a target environment do not have to share the same values. In fact, every process could use a different set of values from the String Table without interfering with each other (but the amount of governance and maintenance involved would make that scenario undesirable). Although there is only one String table, processes read values from the section of the table relevant to each process, so many processes can share the String Table.

![Figure 11: Example of String Table entries.](image)

Every process deployed with the same environment library name will share the same values at runtime.

The String Table is a runtime concept. Values are only populated on deployment of a process. This is an important point: if you change an Environment Field, and then deploy a process, the String Table affecting all processes that share the same environment template and environment will be updated to use this new value.

When blackpearl is installed, there is only one environment library template (‘Default Template’). Inside the default template you will see multiple Environment Definitions, usually for Development and Production. More definitions can be added to a template if more environments are needed.

When a process is deployed, the current Environment Fields in the current Environment Definition from the current template will be deployed to the target server’s String Table. If there are values for fields in the String Table, and then those values are changed in the Environment Field and a process deployed, that value in the String Table will be overwritten. The name of the Environment Definition within the template goes to the Root column in the example above (Figure 11). Every process on the server that uses Environment Definition ‘Development’ will share the same String Table values at runtime. In the example above there are also String Table entries for “HRDevelopment” and several processes share those values.
To have “HRDevelopment” in the Root column of the String Table, an Environment Definition may have been added to the default template. Another method is to add a new Environment Template in K2 Workspace. Each template can then have multiple Environment Definitions. As long as the name of each Definition is unique when deployed to the String Table, only processes using that same name will share the same String Table values. Using this technique in Workspace, a different Environment Template can be created for each development group so they don’t accidentally overwrite another group’s values. The development team that uses the Figure 11 ‘HRDevelopment’ Environment Definition can deploy to different environments and never interfere with the group using the ‘Development’ Environment Definition. Using Visual Studio or K2 Studio you can switch to a different Environment Template in the object browser.

Viewing the String Table on a K2 Server

Within Workspace, go to Management Console, expand the K2 server, expand the Workflow Server, and expand the String Table node. You will see a string table section for each environment deployed on the K2 server (as shown in Figure 12). This string table is what the processes currently running on the server use to resolve variables at run time.

While the string table is listed at both process and server levels within Workspace, it is important to note that this is the same information, surfaced in two places.

**Note:** The string table and environment library may not always contain the same information. It is possible to change the string table directly, as well as make changes in the environment library that have not yet been committed to the string table. Environment library changes are not moved to the string table of the target server until a process is deployed. Pay attention when changing environments for deployment and verify that your string table values are accurate for runtime use. Making a change in the string table will make the change to all processes which share the same environment. There is a one-to-one correlation of string table and environment. Each Environment Definition must be uniquely named.

Figure 12: String Table for a sample development environment.
INTERACTING PROGRAMMATICALLY WITH ENVIRONMENT AND STRING TABLE VALUES

It is also possible to interact programmatically with both the Environment fields and String Table from within blackpearl process code (server events).

- The string tables, containing data consumed by running processes, can be accessed programmatically using the SourceCode.Workflow.Management API. More specifically, the SourceCode.Workflow.Management namespace includes a StringTable class, a StringTableEntry class, and a StringTables class. This is the more common requirement of the two.

**Note**: the String Table contains runtime values. Therefore, it is not meant to be a global variable that can be updated at will.

---

**Example**: Using SourceCode.Workflow.Management.dll to retrieve the current StringTable for a specific Process and modify the value of a specific string table entry.

Note that it is better to edit String Table values from within Workspace than by utilizing code.

```csharp
private void EditStringTableForProcess(string strProcName, string strFieldName, string strNewValue)
{
    // create a connection string

    // force the connection string to a specific credential of k2 server admin
    connBldr.Authenticate = true;
    connBldr.Host = "localhost";
    connBldr.IsPrimaryLogin = true;
    connBldr.SecurityLabelName = "K2";
    connBldr.Port = 5555;
    connBldr.Integrated = true;
    connBldr.UserID = "BPService";
    connBldr.Password = "k2pass";
    connBldr.WindowsDomain = "K2Demo.local";

    // instantiate a connection with this connection string
        new SourceCode.Workflow.Management.WorkflowManagementServer("blackpearl", 5555);
    server.CreateConnection();
    server.Connection.Open(connBldr.ToString());

    {
        // find the specific process
        if (ps.FullName == strProcName)
        {
            // get the name of the stringtable
            string strStringTableName = ps.StringTable;
            SourceCode.Workflow.Management.StringTable st =
                server.GetStringTable(strStringTableName);

            // set the value of the entry
            st.SetString(strFieldName, strNewValue);
        }
    }
}
```
Here is how you can access the string table within a server event using code:

```csharp
public partial class EventItem_22c1f9c0ce724fffd8f8f8373e7f097d1 : ICodeExtender<hostContext>
{
    public void Main(Project_983fdc7a5a1542aabe7bb9daa3efdaae2.EventItemContext_22c1f9c0ce724fffd8f8f8373e7f097d1 K2)
    {
        string url = K2.StringTable["MyPortal"]; // access string table
    }
}
```

We recommend that you define and maintain environment-specific placeholders in the Environment Library and rely on the deployment mechanism to keep the String Table up-to-date.

**Note:** If it is vitally important to manually alter a String Table value in order to address a runtime issue, you can do so, but the corresponding value within the Environment Library must likewise be altered so that the change is remembered when a new workflow is deployed.

### OPTIMIZING MULTIPLE K2 ENVIRONMENTS

When you install K2 and the K2 database is created, the installer automatically creates a Development and Production set of Environment Definitions, with the Environment Fields all pointing to that instance of K2. If you have multiple instances of K2 in your enterprise (Development, QA/Test, and Production in our example), you will have three different SQL instances, with three different databases, and a total of six K2 Environment Definitions. Since you physically have three instances of K2, you should only need three K2 Environment Definitions.

The K2 Designer for Visual Studio is configured to point to a single K2 database. This is where it finds information regarding available environments and the Environment Fields for those environments.

**Tip:** See the section Changing Servers in Visual Studio on page 17 for step-by-step guidance.
To simplify and consolidate deployment and design information, you can combine all of your Environment definitions into a single SQL instance. With this approach you could have three different physical K2 instances and three different Environment Definitions, but all Environment Fields would be stored in a single K2 database. As illustrated in Figure 13, a single Developer client, referencing the DEV K2 server’s database, can retrieve environment fields for any configured environments. The client can then deploy to any of the available environments using these environment fields (assuming sufficient connectivity and permissions).

Note: In the consolidated approach mentioned above, all K2 environments must reside on the same version of K2 blackpearl. If, for example, you try to connect a 4.5 development client to a 4.6 Staging K2 server, the approach will not work.

Figure 13
With this approach, when your K2 Designer for Visual Studio installation is configured, it points to the K2 database where all fields for all your environments are stored. When a developer is designing a process, they would use the Development environment fields all through process design, and then make use of the Development Environment at deployment time. The process would then be deployed to the Development K2 server and the String Table on that Development K2 server used at runtime. Once the process is properly tested and ready to be moved to the QA environment, the developer simply deploys the latest process definition by selecting the QA environment. The deployment process reads the appropriate Workflow and SmartObject server settings, among others, as configured in the QA environment definition, and deploys the K2 artifacts to them accordingly. The process is now hosted on the QA K2 server using the QA version of String Table at runtime.

Finally, when all QA testing has been completed and the process is ready to go live, you can create a deployment package for the process, copy the package to the Production environment and deploy. Or you can simply deploy from the development client by selecting the Production Environment. Using a package is the best-practice method, as giving deploy (export) rights to developers on a Production server is not recommended. Moreover, availability of a staging server where the package is tested is an optional (but helpful) step before deploying to Production. Note that all of these environments must be defined in the Environment Library even though they may not be used during deployment from a K2 Designer (the package will still read from the Environment Library at deployment time).

Typically, developers will not have sufficient rights to deploy directly to another environment. Best practice is that developers will create the package, and then have an authorized administrator deploy it.

UPDATING THE K2 OBJECT BROWSER IN VISUAL STUDIO

It is necessary to refresh the K2 Object Browser before changes made to environment field values within K2 Workspace appear. To do this, click on the drop-down next to the environment in the K2 Object Browser as shown in Figure 14, and then click Refresh.

![K2 Object Browser](image)

Figure 14: Refreshing the K2 Object Browser in Visual Studio.

Changed values in Visual Studio reflect immediately within the Environment Library.
CHANGING SERVERS IN VISUAL STUDIO

To point your designer to a different K2 database, click on the drop-down next to the environment in the K2 Object Browser as shown in Figure 15., and then click Change Server.

A new dialog box is displayed as shown in Figure 16, allowing you to specify a new Connection String. Click on the ellipse (‘…’) button to build a new connection string as per Figure 17.
CHANGING ENVIRONMENTS IN VISUAL STUDIO

If you have your Environment Definitions consolidated within a single database, you can switch environments within the K2 Object Browser without changing Environment Servers. Click on the drop-down arrow next to the Environment in the K2 Object Browser, and then click Environments. A list expands to show you the defined environments and allow you to select a new environment (Figure 18). The K2 Object Browser then displays current values for the new environment.

![Figure 18: List of environments within K2 Object Browser.](image)

DISTRIBUTED VS. CONSOLIDATED ENVIRONMENT LIBRARY – WHICH IS RIGHT?

There are some factors to take into consideration when deciding whether to deploy directly from a Development client to a Production server. For instance, many enterprises do not provide network connectivity between Development clients and Production servers. It is of vital importance that development versions are not accidentally deployed to Production environments, and that development and QA tests do not have the capability to inadvertently overwrite String Table values used in Production.

**Best Practice** is to restrict who has export rights to the K2 Production server. By limiting these permissions to given accounts, developers cannot accidentally deploy to Production.

Many of the same factors must be considered when deciding whether to utilize this consolidated model of Environment field hosting. In addition to the concepts above, such decisions must take the following into consideration:

1. **Environment Field Maintainability**
   Hosting all Environment Definitions within a single database makes it easier to manage from a System Administrator’s point of view. The server administrator can easily manage all fields for all K2 environments from a single place, and developers need not know what any of these values are. If environments are hosted in physically different K2 installations, any addition or deletion of Environment Library fields in one hosted environment is not automatically propagated to the others. For example, if a new environment library field is added in the Development Environment, then it will need to be manually added to QA and Production as well.

2. **Developer/Client Machine Configuration**
   Hosting all Environment Definitions within a single database is also much more straightforward from a client configuration perspective. You can configure a single
development client on a once-off basis and be able to deploy to any of your K2 environments.

The best practice concerning hosting environment settings is to have them on one server, and then to deploy to others using a package. Assuming the correct permissions are in place, deployment-related accidents can thus be prevented. You need to carefully consider the costs and benefits of this consolidated model versus a model whereby each K2 server hosts its own definition in its local SQL databases, and such definitions are available after installation.

Figure 19: Pros and cons of distributed vs. consolidated Environment Library models

PACKAGED DEPLOYMENT


See the K2 blackpearl Developers Reference for info on the environment library architecture: [http://help.k2.com/helppages/k2blackpearlDevRef4.6.5/webframe.html#Architecture_Environment_Library.html](http://help.k2.com/helppages/k2blackpearlDevRef4.6.5/webframe.html#Architecture_Environment_Library.html)
APPENDIX

RECOMMENDED SETUP OF THE ENVIRONMENT LIBRARY

When using the Package and Deployment Wizard, it is recommended that the following approach be used:

**Development**: Only set up the Development Environment definition, and delete the other Environment definitions which are created by default.

**Staging**: Only set up the Staging Environment definition, and delete the other Environment definitions.

**Production**: Only set up the Production Environment definition and delete the other Environment definitions.

The reason for the above is to avoid possible complications caused by the fact that the Package and Deployment Wizard deploys to the environment set in the target system by default. (Should the target environment not be the default on that system, there will be issues.)

When deploying from K2 Studio (for example, an InfoPath process which the Package and Deployment Wizard doesn’t approach), instead of changing the environment in the K2 Object browser, you should switch the server you connect to, to the actual K2 server in that environment. This ensures that you will make use of the correct environment definition.

**Advantages**:

- One environment definition on each Development, Staging and Production environment to be managed.
- Permissions can be set in such a way that the environment definitions aren’t accidently changed, which can happen in a more loosely-secured Development environment.
- Works with the new Package and Deployment tool behavior.

**Disadvantages**:

- There is a need to manage different environment definitions in separate places.
- There is a need to manually synchronize the new environment definition values in each environment if using K2 Studio for InfoPath processes, since adding a new environment value in Development does not automatically add it to Staging and Production. K2 Studio deployments will fail if the environment definition entries aren’t present.
- There is a need to change servers before deploying to different environments from within K2 Studio.
- The ‘Create package’ functionality will not work from within K2 Studio unless you manually change the msbuild environment fields.