Perceptive Document Composition

Configuration

Version: 5.3

perceptivesoftware

Written by: Product Documentation, R&D
Date: May 2013

Table of Contents

General Information

The Environment Variable PDCdir

The Composition Studio setup creates the environment variable **PDCdir**.

This is a system variable and as such is visible to all Windows users.

The value of the variable is the path to the Composition Studio installation directory.

Example

If the Studio was installed in the directory

D:\PDC

the environment variable has the value

D:\PDC (ends with a backslash).

Configuring Document Composition Client

General Information

The standard name of the Document Composition Client configuration file is

PDC.Client.exe.config.

The configuration file is located in the same directory as the application file **PDC.Client.exe**.

The Configuration File PDC.Client.exe.config

Example Configuration

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="CompositeUI"
type="Microsoft.Practices.CompositeUI.Configuration.SettingsSection,
Microsoft.Practices.CompositeUI" allowExeDefinition="MachineToLocalUser" />
    <sectionGroup name="userSettings"
type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section name="ModusSuite.Studio.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false" />
    </sectionGroup>
    <section name="PropertyEditor"
type="ModusSuite.Common.PropertyEditors.PropertyEditorConfiguration,
ModusSuite.Common.PropertyEditors" />
  </configSections>

  <appSettings>
```

```
    <add key="sts" value="http://localhost:8000/sts" />
    <add key="login" value="http://localhost:8010/mur/login" />
    <add key="mur" value="http://localhost:8010/mur/data" />
    <add key="mwsrws" value="http://localhost:8010/mws/mwsrepository" />
    <add key="mws" value="http://localhost:8011/mws/mwsprocess" />
    <add key="dpws" value="http://localhost:8010/dataprovider" />
    <add key="ows" value="http://localhost:8010/ows/owsrepository"/>
    <add key="odinviewsystemoid" value="dm" />
    <add key="odinviewdbalias" value="odin_mws" />
    <add key="useInternalDocumentViewer" value="False" />
    <add key="license" value="http://localhost:8010/mur/license" />
    <add key="enableOnlinePrinterPrintTimeChanging" value="False" />
    <add key="blockExpandLevel" value="1" />
    <add key="shortProcDescLength" value="0" />
    <add key="loadRootFolderOnBundleAutoStart" value="False" />
    <add key="loadFolderTreeOnBundleAutoStart" value="False" />
    <add key="showMessageBoxOnMwsError" value="True" />
    <add key="showInfoBoxOnAutomaticForwarding" value="True" />
    <add key="WordUpdateTOC" value="true" />
    <add key="showProcessOptions" value="true" />
  </appSettings>

  <CompositeUI>
    <services>
      <add
serviceType="Microsoft.Practices.CompositeUI.Services.IAuthenticationService,
Microsoft.Practices.CompositeUI"
instanceType="ModusSuite.Common.CAB.AuthenticationService,
ModusSuite.Common.CAB.ClientAuthenticationService" />
    </services>
  </CompositeUI>

  <PropertyEditor configSource="ModusSuite.Common.PropertyEditors.dll.config"
/>

  <system.diagnostics>
    <sources>
      <source name="mws" switchValue="Information" >
        <listeners>
          <add name="tracefile_listener"/>
          <remove name="Default"/>
        </listeners>
      </source>

      <source name="modusstudio" switchValue="Information" >
        <listeners>
```

```
          <add name="outputmanager_listener"/>
          <remove name="Default"/>
        </listeners>
      </source>
      <source name="mwsclient" switchValue="Verbose" >
        <listeners>
          <add name="outputmanager_listener"/>
          <add name="tracefile_listener"/>
          <remove name="Default"/>
        </listeners>
      </source>
      <!-- This source is used by run time processes -->
      <source name="runtime_process" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch" >
        <listeners>
          <add name="outputmanager_listener"/>
          <add name="tracefile_listener"/>
          <remove name="Default"/>
        </listeners>
      </source>
      <source name="odin" switchValue="Information" >
        <listeners>
          <add name="tracefile_listener"/>
          <add name="outputmanager_listener"/>
          <remove name="Default"/>
        </listeners>
      </source>
    <sources>

    <sharedListeners>
      <add name="outputmanager_listener"
type="ModusSuite.Studio.OutputManager.Types.OutputWindowTraceListener,
ModusSuite.Studio.OutputManager, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null"/>
      <!-- <add name="tracefile_listener"
type="System.Diagnostics.DelimitedListTraceListener" delimiter=";"
initializeData="PDC_Suite_Log.txt" traceOutputOptions="DateTime, ProcessId,
ThreadId"/> -->
      <add name="tracefile_listener"
type="ModusSuite.Common.SystemFramework.DailyTraceListener,
ModusSuite.Common.SystemFramework" delimiter=";"
initializeData="PDC.Client_Log.txt" traceOutputOptions="DateTime, ProcessId,
ThreadId"/>
    </sharedListeners>

    <trace autoflush="true"/>
  </system.diagnostics>
```

```
  <userSettings>
    <ModusSuite.Studio.Properties.Settings>
      <setting name="Layout" serializeAs="String">
        <value />
      </setting>
      <setting name="Filter" serializeAs="String">
        <value />
      </setting>
    </ModusSuite.Studio.Properties.Settings>
  </userSettings>

  <system.serviceModel>
    <!--For debugging purposes set the includeExceptionDetailInFaults
attribute to true-->
    <bindings>
      <wsHttpBinding>
        <binding name="WSHttpBinding_IDataProvider" >
          <security mode="Message">
            <message establishSecurityContext="false" />
          </security>
        </binding>
      </wsHttpBinding>
      <!-- Binding for Debugger -->
      <wsDualHttpBinding>
        <binding name="DebugClientBinding"
          clientBaseAddress="http://localhost:4711/DebugClient/"
          sendTimeout="Infinite"
          receiveTimeout="Infinite"
          maxBufferPoolSize="2147483600" maxReceivedMessageSize="2147483600">
          <readerQuotas maxStringContentLength="1003741824"
 maxArrayLength="2147483647" />
          <security mode="None"/>
        </binding>
      </wsDualHttpBinding>
    </bindings>

    <client>
      <!--
        <endpoint address="http://127.0.0.1/DPWS/DPWS.svc"
binding="wsHttpBinding"
            bindingConfiguration="WSHttpBinding_IDataProvider"
contract="IDataProvider"
            name="WSHttpBinding_IDataProvider">
            <identity>
                <userPrincipalName value="127.0.0.1\ASPNET" />
```

```
            </identity>
          </endpoint>
          -->
        <endpoint name="ModusSuite.Runtime.DebugService"
          address="http://127.0.0.1:8012/RemoteDebugger"
          binding="wsDualHttpBinding"
          bindingConfiguration="DebugClientBinding"
          contract="ModusSuite.Runtime.Types.IDebugService" />
      </client>
    </system.serviceModel>
</configuration>
```

appSettings

```
<appSettings>
   <add key="sts" value="http://localhost:8000/sts" />
   <add key="login" value="http://localhost:8010/mur/login" />
   <add key="mur" value="http://localhost:8010/mur/data" />
   <add key="mwsrws" value="http://localhost:8010/mws/mwsrepository" />
   <add key="mws" value="http://localhost:8011/mws/mwsprocess" />
   <add key="dpws" value="http://localhost:8010/dataprovider" />
   <add key="ows" value="http://localhost:8010/ows/owsrepository"/>
   <add key="odinviewsystemoid" value="dm" />
   <add key="odinviewdbalias" value="odin_mws" />
   <add key="useInternalDocumentViewer" value="False" />
   <add key="license" value="http://localhost:8010/mur/license" />
   <add key="enableOnlinePrinterPrintTimeChanging" value="False" />
   <add key="blockExpandLevel" value="1" />
   <add key="shortProcDescLength" value="0" />
   <add key="loadRootFolderOnBundleAutoStart" value="False" />
   <add key="loadFolderTreeOnBundleAutoStart" value="False" />
   <add key="showMessageBoxOnMwsError" value="True" />
   <add key="showInfoBoxOnAutomaticForwarding" value="True" />
   <add key="WordUpdateTOC" value="true" />
   <add key="showProcessOptions" value="true" />
</appSettings>
```

Possible Keys

Attribute	Definition
sts	See Client Configuration sts
login	See Client Configuration mur
mur	See Client Configuration mur

<table>
<tr><td>mwsrws</td><td>See Client Configuration mwsrws</td></tr>
<tr><td>mws</td><td>See Client Configuration mws</td></tr>
<tr><td>dpws</td><td>See Client Configuration dpws</td></tr>
<tr><td>ows</td><td>See Client Configuration ows</td></tr>
<tr><td>odinviewsystemoid</td><td>See Client Configuration Settings</td></tr>
<tr><td>odinviewdbalias</td><td>See Client Configuration Settings</td></tr>
<tr><td>useInternalDocumentViewer</td><td><table><tr><td>True</td><td>Generated documents are displayed using the internal viewer.</td></tr><tr><td>False</td><td>Generated documents are displayed using Microsoft Word.</td></tr></table></td></tr>
<tr><td>license</td><td>internal</td></tr>
<tr><td>enableOnlinePrinterPrintTimeChanging</td><td><table><tr><td>True</td><td>Allow change print time for online printers</td></tr><tr><td>False</td><td>Do not allow change print time for online printers</td></tr></table>Standard value is False, when no specification made.</td></tr>
<tr><td>blockExpandLevel</td><td>Allows you to define the level to which text blocks are automatically expanded when a document is selected.
Possible settings are:<table><tr><td>-1</td><td>expand all levels</td></tr><tr><td>0</td><td>no levels expanded</td></tr><tr><td>x</td><td>expand all blocks to level x</td></tr></table>Standard value is 1, when no</td></tr>
</table>

<table>
<tr><td></td><td>specification made.</td></tr>
<tr><td>shortProcDescLength</td><td>Show Process Description in Process List.
0 none (default), x>0 the first x characters of the process description<table><tr><td>0</td><td>Process description is not returned.</td></tr><tr><td>x</td><td>The first x characters of the process description are shown.</td></tr></table>Standard value is 0, when no specification made.</td></tr>
<tr><td>loadRootFolderOnBundleAutoStart</td><td>This setting only applies when the start parameter objectname or processid are used to load a bundle.<table><tr><td>True</td><td>Loads the content of the root folder and the folder structure.</td></tr><tr><td>False</td><td>Only loads the folder structure.
The content of the root folder can be loaded by pressing the button Refresh.</td></tr></table>Standard value is True, when no specification made.</td></tr>
<tr><td>loadFolderTreeOnBundleAutoStart</td><td>This setting only applies when the start parameter objectname or processid are used to load a bundle.
Standard value is True, when no specification made.
If the setting is set to False the system is not loaded (neither content of the root folder, nor folder structure).</td></tr>
</table>

<table>
<tr><td></td><td>The setting for loadRootFolderOnBundleAutoStart is ignored.

A system must be defined by a valid SystemOID in the start parameter startsystem.

Selecting a system is only possible when the current process has been closed or saved.

Please note:

If the start parameter closeonexitprocess is specified, the Document Composition Client will close automatically when a process is closed or saved!</td></tr>
<tr><td>showMessageBoxOnMwsError</td><td><table><tr><td>True</td><td>Displays MWS server error messages in a separate window.</td></tr><tr><td>False</td><td>Displays MWS server error messages in output window only.</td></tr></table>
Standard value is True, when no specification made.</td></tr>
<tr><td>showInfoBoxOnAutomaticForwarding</td><td><table><tr><td>True</td><td>Displays the pop-up message box on automatic process forwarding.</td></tr><tr><td>False</td><td>Pop-up message box is not displayed.</td></tr></table>
Standard value is True, when no specification made.</td></tr>
<tr><td>WordUpdateTOC</td><td>See Updating the Table of Contents in Documents</td></tr>
<tr><td>showProcessOptions</td><td><table><tr><td>True</td><td>The dialog box Process Options is displayed when a bundle is opened.</td></tr><tr><td>False</td><td>The dialog box Process Options is not displayed when a bundle is opened.</td></tr></table></td></tr>
</table>

	Passes an empty string as value for Process Title and Process Type.

Standard value is **True** when the attribute is not specified.

sources

```
<sources>
   <source name="mws" switchValue="Information" >
      <listeners>
         <add name="tracefile_listener"/>
         <remove name="Default"/>
      </listeners>
   </source>
   <source name="modusstudio" switchValue="Information" >
      <listeners>
         <add name="outputmanager_listener"/>
         <remove name="Default"/>
      </listeners>
   </source>
   <source name="mwsclient" switchValue="Verbose" >
      <listeners>
         <add name="outputmanager_listener"/>
         <add name="tracefile_listener"/>
         <remove name="Default"/>
      </listeners>
   </source>
   <!-- This source is used by run time processes -->
   <source name="runtime_process" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch" >
      <listeners>
         <add name="outputmanager_listener"/>
         <add name="tracefile_listener"/>
         <remove name="Default"/>
      </listeners>
   </source>
   <source name="odin" switchValue="Information" >
      <listeners>
         <add name="tracefile_listener"/>
         <add name="outputmanager_listener"/>
         <remove name="Default"/>
      </listeners>
   </source>
<sources>
```

See:

Configuration Files for Logging

sharedListeners

```
<sharedListeners>
   <add name="outputmanager_listener"
type="ModusSuite.Studio.OutputManager.Types.OutputWindowTraceListener,
ModusSuite.Studio.OutputManager, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null"/>
   <!-- <add name="tracefile_listener"
type="System.Diagnostics.DelimitedListTraceListener" delimiter=";"
initializeData="PDC_Suite_Log.txt" traceOutputOptions="DateTime, ProcessId,
ThreadId"/> -->
   <add name="tracefile_listener"
type="ModusSuite.Common.SystemFramework.DailyTraceListener,
ModusSuite.Common.SystemFramework" delimiter=";"
initializeData="PDC.Client_Log.txt" traceOutputOptions="DateTime, ProcessId,
ThreadId"/>
</sharedListeners>
```

See:

Configuration Files for Logging

Updating the Table of Contents in Documents

The key **WordUpdateTOC** controls whether the table of contents is updated when the document is opened from the Client.

If no entry is found, the value is set to **True**.

Document Composition Client

The key for Document Composition Client is defined in the file **PDC.Client.exe.config**.

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

Configuration of Odin Views for Clients

Plug-In for Odin Views

To activate Odin Views in a Document Composition Client, the plug-in configuration file **clientprofile.xml** must contain the XML element

```
<ModuleInfo AssemblyFile="ModusSuite.Odin.OdinViews.dll" />.
```

Example:

```
<?xml version="1.0" encoding="utf-8" ?>
<SolutionProfile xmlns="http://schemas.microsoft.com/pag/cab-profile" >
   <Modules>
      …
      <ModuleInfo AssemblyFile="ModusSuite.Odin.OdinViews.dll" />
      ...
   </Modules>
</SolutionProfile>
```

If the element is not present, the plug-in for Odin Views is not loaded when the Client is started and the functionality is not available.

Client Configuration Settings

For Odin Views to function correctly in a Document Composition Client they require the following settings in the section **appSettings**:

1. Specification of the system the Views and DB Alias are loaded from.
   Uses the key **odinviewsystemoid**.
   The value used is the SystemObjectID.
2. Specification of the name of the DB Alias.
   Uses the key odinviewdbalias
   The value used is the name of the DB Alias.
   The DB Alias is loaded from the system specified in the key **odinviewsystemoid**.

```
<add key="odinviewsystemoid" value="m5_e" />
<add key="odinviewdbalias" value="OWSMWS" />
```

Specification of the system that Odin View Definitions and database connection information are loaded from.

Example:

```
<configuration>
   …
   <appSettings>
      <add key="odinviewsystemoid" value="dm" />
      <add key="odinviewdbalias" value="odin" />
   </appSettings>
   …
</configuration>
```

Configuring Document Composition Web Client

General Information

The standard name of the WebClient configuration file is

PDC.WebClient.xml

The configuration file is located in the same directory as the application file **PDC.Client.exe**.

The Configuration File PDC.WebClient.xml

Example Configuration

```
<configuration>
   <appSettings>
      <add key="mwsProcessServiceRest"
value="http://megatron:8011/mwsrest/mwsprocess" />
      <add key="mwsProcessServiceBasic"
value="http://megatron:8011/mwsbasic/mwsprocess" />
      <add key="mwsRepositoryServiceBasic"
value="http://megatron:8010/mwsbasic/mwsrepository" />
      <add key="MURServiceBasic" value="http://megatron:8010/murbasic/data"
/>
      <add key="blockExpandLevel" value="1" />
      <add key="showMessageBoxOnMwsError" value="true" />
      <add key="showInfoBoxOnAutomaticForwarding" value="true" />
      <add key="shortProcDescLength" value="0" />
      <add key="useInternalDocumentViewer" value="false" />
      <add key="enableOnlinePrinterPrintTimeChanging" value="false" />
      <add key="loadRootFolderOnBundleAutoStart" value="true" />
      <add key="loadFolderTreeOnBundleAutoStart" value="true" />
      <add key="wordUpdateTOC" value="true" />
      <add key="showProcessOptions" value="true" />
   </appSettings>
</configuration>
```

appSettings

```
<appSettings>
   <add key="mwsProcessServiceRest"
value="http://megatron:8011/mwsrest/mwsprocess" />
   <add key="mwsProcessServiceBasic"
value="http://megatron:8011/mwsbasic/mwsprocess" />
   <add key="mwsRepositoryServiceBasic"
value="http://megatron:8010/mwsbasic/mwsrepository" />
   <add key="MURServiceBasic" value="http://megatron:8010/murbasic/data" />
   <add key="blockExpandLevel" value="1" />
   <add key="showMessageBoxOnMwsError" value="true" />
   <add key="showInfoBoxOnAutomaticForwarding" value="true" />
   <add key="shortProcDescLength" value="0" />
   <add key="useInternalDocumentViewer" value="false" />
   <add key="enableOnlinePrinterPrintTimeChanging" value="false" />
   <add key="loadRootFolderOnBundleAutoStart" value="true" />
   <add key="loadFolderTreeOnBundleAutoStart" value="true" />
   <add key="wordUpdateTOC" value="true" />
   <add key="showProcessOptions" value="true" />
</appSettings>
```

Possible Keys

<table>
<tr><th>Attribute</th><th>Definition</th></tr>
<tr><td>mwsProcessServiceRest</td><td>internal</td></tr>
<tr><td>mwsProcessServiceBasic</td><td>internal</td></tr>
<tr><td>mwsRepositoryServiceBasic</td><td>intern</td></tr>
<tr><td>MURServiceBasic</td><td>internal</td></tr>
<tr><td>mws</td><td>See Client Configuration mws</td></tr>
<tr><td>dpws</td><td>See Client Configuration dpws</td></tr>
<tr><td>ows</td><td>See Client Configuration ows</td></tr>
<tr><td>odinviewsystemoid</td><td>See Client Configuration Settings</td></tr>
<tr><td>odinviewdbalias</td><td>See Client Configuration Settings</td></tr>
<tr><td>blockExpandLevel</td><td>Allows you to define the level to which text blocks are automatically expanded when a document is selected.
Possible settings are:<table><tr><td>-1</td><td>expand all levels</td></tr><tr><td>0</td><td>no levels expanded</td></tr><tr><td>x</td><td>expand all blocks to level x</td></tr></table>Standard value is 1, when no specification made.</td></tr>
<tr><td>showMessageBoxOnMwsError</td><td><table><tr><td>True</td><td>Displays MWS server error messages in a separate window.</td></tr><tr><td>False</td><td>Displays MWS server error messages in output window only.</td></tr></table>Standard value is True, when no specification made.</td></tr>
</table>

<table>
<tr><td>showInfoBoxOnAutomaticForwarding</td><td><table><tr><td>True</td><td>Displays the pop-up message box on automatic process forwarding.</td></tr><tr><td>False</td><td>Pop-up message box is not displayed.</td></tr></table>Standard value is True, when no specification made.</td></tr>
<tr><td>WordUpdateTOC</td><td>See Updating the Table of Contents in Documents</td></tr>
<tr><td>showProcessOptions</td><td><table><tr><td>True</td><td>The dialog box Process Options is displayed when a bundle is opened.</td></tr><tr><td>False</td><td>The dialog box Process Options is not displayed when a bundle is opened. Passes an empty string as value for Process Title and Process Type.</td></tr></table>Standard value is True when the attribute is not specified.</td></tr>
<tr><td>shortProcDescLength</td><td>Show Process Description in Process List.
0 none (default), x>0 the first x characters of the process description<table><tr><td>0</td><td>Process description is not returned.</td></tr><tr><td>x</td><td>The first x characters of the process description are shown.</td></tr></table>Standard value is 0, when no specification made.</td></tr>
<tr><td>useInternalDocumentViewer</td><td><table><tr><td>True</td><td>Generated documents are displayed using the internal viewer.</td></tr></table></td></tr>
</table>

	False: Generated documents are displayed using Microsoft Word.
enableOnlinePrinterPrintTimeChanging	True: Allow change print time for online printers False: Do not allow change print time for online printers Standard value is **False**, when no specification made.
loadRootFolderOnBundleAutoStart	This setting only applies when the start parameter **objectname** or **processid** are used to load a bundle. True: Loads the content of the root folder and the folder structure. False: Only loads the folder structure. The content of the root folder can be loaded by pressing the button **Refresh**. Standard value is True, when no specification made.
loadFolderTreeOnBundleAutoStart	This setting only applies when the start parameter **objectname** or **processid** are used to load a bundle. Standard value is True, when no specification made. If the setting is set to **False** the system is not loaded (neither content of the root folder, nor folder structure). The setting for **loadRootFolderOnBundleAutoStart** is ignored. A system must be defined by a valid

<table>
<tr><td></td><td>SystemOID in the start parameter startsystem.

Selecting a system is only possible when the current process has been closed or saved.

Please note:

If the start parameter closeonexitprocess is specified, the Client will close automatically when a process is closed or saved!</td></tr>
<tr><td>wordUpdateTOC</td><td>See Updating the Table of Contents in Documents</td></tr>
<tr><td>showProcessOptions</td><td><table><tr><td>True</td><td>The dialog box Process Options is displayed when a bundle is opened.</td></tr><tr><td>False</td><td>The dialog box Process Options is not displayed when a bundle is opened. Passes an empty string as value for Process Title and Process Type.</td></tr></table>
Standard value is True when the attribute is not specified.</td></tr>
</table>

Updating the Table of Contents in Documents

The key **WordUpdateTOC** controls whether the table of contents is updated when the document is opened from the Client.

True	The table of contents is updated.
False	The table of contents is not updated.

If no entry is found, the value is set to **True**.

Web Client

The key for the WebClient is defined in the file **PDC.WebClient.xml**.

```
<configuration>
  <appSettings>
    <!--<add key="odinViewSystemoid" value="" />
```

```
    <add key="odinViewDbAlias" value="" />
    <add key="mwsProcessServiceRest" value="" />-->
    <add key="mwsProcessServiceBasic"
value="http://localhost:8011/mwsbasic/mwsprocess" />
    <add key="mwsRepositoryServiceBasic"
value="http://localhost:8010/mwsbasic/mwsrepository" />
    <add key="MURServiceBasic" value="http://localhost:8010/murbasic/data" />
    <add key="blockExpandLevel" value="1" />
    <add key="showMessageBoxOnMwsError" value="true" />
    <add key="showInfoBoxOnAutomaticForwarding" value="true" />
    <add key="shortProcDescLength" value="0" />
    <add key="useInternalDocumentViewer" value="false" />
    <add key="enableOnlinePrinterPrintTimeChanging" value="false" />
    <add key="loadRootFolderOnBundleAutoStart" value="true" />
    <add key="loadFolderTreeOnBundleAutoStart" value="true" />
    <add key="wordUpdateTOC" value="true" />
  </appSettings>
</configuration>
```

Configuring Composition Studio

General Information

The standard name of the Composition Studio configuration file is

PDC.Studio.exe.config

The configuration file is located in the same directory as the application file PDC.Client.exe.

The Configuration File PDC.Studio.exe.config

Example Configuration

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
      <section name="mlsdatabase"
type="ModusSuite.Common.SystemFramework.AliasConfigurationSection,
ModusSuite.Common.SystemFramework"/>
      <section name="CompositeUI"
type="Microsoft.Practices.CompositeUI.Configuration.SettingsSection,
Microsoft.Practices.CompositeUI" allowExeDefinition="MachineToLocalUser" />
      <section name="modusruntime"
type="ModusSuite.Runtime.SystemFramework.RuntimeConfiguration,
ModusSuite.Runtime.SystemFramework" />
      <section name="mws" type="ModusSuite.MWS.SystemFramework.Configuration,
ModusSuite.MWS.SystemFramework" />
      <section name="xdata"
type="ModusSuite.Xdata.SystemFramework.XdataConfiguration,
ModusSuite.Xdata.SystemFramework" />
```

```
      <section name="remotecontrol"
type="ModusSuite.Runtime.RemoteControlSystemFramework.RemoteControlConfigurat
ion, ModusSuite.Runtime.RemoteControlSystemFramework" />
      <section name="inputmaskpresets"
type="ModusSuite.Studio.ModusEditor.Types.PresetConfigurationSection,
ModusSuite.Studio.ModusEditor" />
      <sectionGroup name="userSettings"
type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" >
         <section name="ModusSuite.Studio.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false" />
         <section name="ModusSuite.Studio.Navigator.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false" />
      </sectionGroup>
      <section name="dbalias"
type="ModusSuite.Common.ConnectionStringEditor.SqlTemplateConfigurationSectio
n, ModusSuite.Common.ConnectionStringEditor" />
      <section name="PropertyEditor"
type="ModusSuite.Common.PropertyEditors.PropertyEditorConfiguration,
ModusSuite.Common.PropertyEditors" />
      <section name="odinSettings"
type="ModusSuite.Odin.SystemFramework.OdinConfigurationSection,
ModusSuite.Odin.SystemFramework" />
   </configSections>
   <appSettings>
      <add key="loadDbAliasFromLocalFile" value="False" />
      <add key="wordtemplate"
value="%APPDATA%\Microsoft\Templates\normal.dotm"/>
      <add key="sts" value="http://localhost:8000/sts"/>
      <add key="login" value="http://localhost:8010/mur/login"/>
      <add key="license" value="http://localhost:8010/mur/license"/>
      <add key="rws" value="http://localhost:8010/rws"/>
      <add key="mwsrws" value="http://localhost:8010/mws/mwsrepository"/>
      <add key="dpws" value="http://localhost:8010/dataprovider"/>
      <add key="odinparameterfile" value="odinparameter.config"/>
      <add key="ows" value="http://localhost:8010/ows/owsrepository"/>
      <add key="useInternalDocumentViewer" value="False" />
   </appSettings>
   <modusruntime>
      <runtimeservices>
         <service name="repository"
assembly="ModusSuite.Runtime.RepositoryService"
type="ModusSuite.Runtime.RepositoryService"/>
```

```
         <service name="threading"
assembly="ModusSuite.Runtime.ThreadingService"
type="ModusSuite.Runtime.ThreadingService"/>
         <service name="timer" assembly="ModusSuite.Runtime.TimerService"
type="ModusSuite.Runtime.TimerService"/>
         <service name="debugger"  assembly="ModusSuite.Runtime.DebugService"
type="ModusSuite.Runtime.DebugService"/>
         <service name="mws" assembly="ModusSuite.Runtime.MWSRuntimeService"
type="ModusSuite.Runtime.MWSRuntimeService"/>
      </runtimeservices>
   </modusruntime>
   <CompositeUI>
      <services>
         <add
serviceType="Microsoft.Practices.CompositeUI.Services.IAuthenticationService,
Microsoft.Practices.CompositeUI"
instanceType="ModusSuite.Common.CAB.AuthenticationService,
ModusSuite.Common.CAB.ClientAuthenticationService" />
      </services>
   </CompositeUI>
   <odinSettings configSource="odin.config"/>
   <mws systemoid="DEBUG OID" dbalias="mws" process="mws_standard"
client_runtime_mode="true"> </mws>
   <remotecontrol configSource="remotecontrol.config"/>
   <xdata configSource="xdata.config" />
   <dbalias configSource="dbalias.config" />
   <inputmaskpresets configSource="inputmasks.config" />
   <PropertyEditor
configSource="ModusSuite.Common.PropertyEditors.dll.config" />
   <runtime>
      <generatePublisherEvidence enabled="false"/>
   </runtime>
   <system.diagnostics>
      <sources>
         <source name="mws" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <source name="xdata" switchName="xdata_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="outputmanager_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
```

```
         <source name="modusstudio" switchName="studio_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="outputmanager_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <source name="monalisa" switchName="monalisa_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <!-- This source is used by run time processes -->
         <source name="runtime_process" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <source name="runtime_components" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <source name="odin" switchName="odin_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <add name="outputmanager_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <source name="textsystem" switchName="textsystem_switch"
switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <add name="outputmanager_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
         <!-- this source is used by the class EventTopic of the CAB, e.g. to
log exceptions caused when
```

```
         calling Fire
         -->
         <source
name="Microsoft.Practices.CompositeUI.EventBroker.EventTopic"
switchName="cab_switch" switchType="System.Diagnostics.SourceSwitch" >
            <listeners>
               <add name="tracefile_listener"/>
               <remove name="Default"/>
            </listeners>
         </source>
      <sources>
      <switches>
         <!--<add name="test_switch" value="1" />-->
         <!-- You can set the level of tracing -->
         <!-- You can turn tracing off -->
         <!--add name="test_switch" value="Off" -->
         <add name="runtime_switch" value="Information" />
         <add name="odin_switch" value="Information" />
         <add name="xdata_switch" value="Information" />
         <add name="studio_switch" value="Information" />
         <add name="monalisa_switch" value="Information" />
         <add name="textsystem_switch" value="Information" />
         <add name="cab_switch" value="Off" /> <!-- set to "Information" to
log EventTopic exceptions -->
      </switches>
      <sharedListeners>
         <add name="outputmanager_listener"
type="ModusSuite.Studio.OutputManager.Types.OutputWindowTraceListener,
ModusSuite.Studio.OutputManager, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=null"/>
         <add name="tracefile_listener"
type="ModusSuite.Common.SystemFramework.DailyTraceListener,
ModusSuite.Common.SystemFramework" delimiter=";"
initializeData="PDC.Studio_Log.txt" traceOutputOptions="DateTime"/>
      </sharedListeners>
      <trace autoflush="true"/>
   </system.diagnostics>
   <userSettings>
      <ModusSuite.Studio.Properties.Settings>
         <setting name="Layout" serializeAs="String">
            <value />
         </setting>
      </ModusSuite.Studio.Properties.Settings>
      <ModusSuite.Studio.Navigator.Properties.Settings>
         <setting name="NavigatorMaxRowsPerPage" serializeAs="String">
            <value>-1</value>
         </setting>
```

```
      </ModusSuite.Studio.Navigator.Properties.Settings>
   </userSettings>
   <system.serviceModel>
      <client>
         <endpoint name="RemoteControlServiceXWS"
                   address="http://localhost:4723/RemoteControlService"
                   binding="wsDualHttpBinding"
                   bindingConfiguration="Binding1"
                   contract="ModusSuite.Runtime.RemoteControlContract.IRuntim
eControlService" />
         <endpoint name="RemoteControlServiceOWS"
                   address="http://localhost:4722/RemoteControlService"
                   binding="wsDualHttpBinding"
                   bindingConfiguration="Binding1"
                   contract="ModusSuite.Runtime.RemoteControlContract.IRuntim
eControlService" />
         <endpoint name="RemoteControlServiceMWS"
                   address="http://localhost:4721/RemoteControlService"
                   binding="wsDualHttpBinding"
                   bindingConfiguration="Binding1"
                   contract="ModusSuite.Runtime.RemoteControlContract.IRuntim
eControlService" />
         <endpoint name="RemoteControlServiceCore"
                   address="http://localhost:4720/RemoteControlService"
                   binding="wsDualHttpBinding"
                   bindingConfiguration="Binding1"
                   contract="ModusSuite.Runtime.RemoteControlContract.IRuntim
eControlService" />
      </client>
      <bindings>
         <wsDualHttpBinding>
            <binding name="Binding1"
                     clientBaseAddress="http://localhost:8000/RemoteControlCl
ient/"
                     maxBufferPoolSize="2147483600"
maxReceivedMessageSize="2147483600"
                     receiveTimeout="Infinite"
                     sendTimeout="Infinite">
               <readerQuotas
                maxStringContentLength="2147483647"
                maxArrayLength="2147483647" />
               <security mode="None"/>
            </binding>
         </wsDualHttpBinding>
         <wsHttpBinding>
         <!--  Settings for communication with repository (rws)
```

```
               DO NOT DELETE the binding element. Element is read when the
channel
               is created! --> -->
            <binding name="rws" sendTimeout="00:05:00" />
         </wsHttpBinding>
      </bindings>
      <!--For debugging purposes set the includeExceptionDetailInFaults
attribute to true-->
      <behaviors>
         <serviceBehaviors>
            <behavior name="DebugServiceBehavior">
               <serviceThrottling maxConcurrentCalls="1000000"
maxConcurrentInstances="1000000" maxConcurrentSessions="1000000"/>
               <serviceMetadata httpGetEnabled="True"/>
               <serviceDebug includeExceptionDetailInFaults="False" />
            </behavior>
         </serviceBehaviors>
      </behaviors>
   </system.serviceModel>
</configuration>
```

appSettings

```
<appSettings>
   <add key="loadDbAliasFromLocalFile" value="False" />
   <add key="wordtemplate"
value="%APPDATA%\Microsoft\Templates\normal.dotm"/>
   <add key="sts" value="http://localhost:8000/sts"/>
   <add key="login" value="http://localhost:8010/mur/login"/>
   <add key="license" value="http://localhost:8010/mur/license"/>
   <add key="rws" value="http://localhost:8010/rws"/>
   <add key="mwsrws" value="http://localhost:8010/mws/mwsrepository"/>
   <add key="dpws" value="http://localhost:8010/dataprovider"/>
   <add key="odinparameterfile" value="odinparameter.config"/>
   <add key="ows" value="http://localhost:8010/ows/owsrepository"/>
   <add key="useInternalDocumentViewer" value="False" />
   <add key="compilewordml2007only" value="true"/>
</appSettings>
```

Possible Keys

Attribute	Definition	
loadDbAliasFromLocalFile	True	DB Alias is loaded from a configuration file. See Load DB Alias from a File

	False: DB Alias is loaded from the repository.
wordtemplate	Specifies which normal.dotm Studio uses. For more information, see the following chapters of the PDC.Studio manual: • Perceptive Document Composition Studio > Administration > System Administration > Assign a Word Template • Perceptive Document Composition Studio > Word Templates
sts	See sts – Security Token Service
login	internal
license	internal
rws	See rws – RepositoryWebService
mwsrws	See mwsrws and mwsrwsbasic – ModusRepositoryWebService
dpws	See dpws – DataProviderService
odinparameterfile	Specifies the name of the configuration file that defines the parameter used for importing print jobs in the Odin database
ows	See ows and owsbasic – OdinWebService
useInternalDocumentViewer	True: Generated documents are displayed using the internal viewer. False: Generated documents are displayed using Microsoft Word.

compilewordml2007only	True	The key must be set to **True** for installations using both Word 2010 and Word 2003 with compatibility pack.
	False	The key must be set to **False** for installations using only Word 2010. Standard value is **False**, if key is missing.

mws

<mws systemoid="DEBUG OID" dbalias="mws" process="mws_standard" client_runtime_mode="true"> </mws>

Attributes of the Element mws

Attribute	Definition
systemoid	ObjectID of the system where the MWS process and DB Alias are stored.
dbalias	Name of the DB Alias used to access the MWS database.
process	The name of the MWS process that must reside in the system specified in the attribute **systemoid**.
client_runtime_mode	Specifies whether the MWS runs on a Client (Composition Studio) or on the Server

True	MWS runs in Composition Studio (on the client). **Please note:** It is not the process defined in the above system that is executed here. Instead, the application Composition Studio creates an internal process to test documents. The value of the attribute **systemoid** should be set to **DEBUG OID**. The attribute **dbalias** is not interpreted.
False	MWS runs on the server.

protect_document	This attribute is only interpreted when the attribute

<table>
<tr><td></td><td>client_runtime_mode has the value false.

The attribute applies to documents with password protection. It specifies whether document protection is activated directly by the server after documents have been generated.

Default value is false when the attribute is not specified.
<table><tr><td>True</td><td>Document protection is activated by the MWS Server immediately after generation</td></tr><tr><td>False</td><td>Document protection is activated by the Client (Document Composition clients or third-party clients).</td></tr></table></td></tr>
<tr><td>set_processtitle_using_poolvar</td><td>Specifies the name of a variable in the Systempool to be read when a process is closed or forwarded and set as the value for Process Title.

Default value is Empty string when the attribute is not specified.</td></tr>
<tr><td>enableWord2003</td><td>Default value is false when the attribute is not specified.

Specifies whether documents are made compatible with Word 2003 before being downloaded and displayed by a Client using Word 2003 with compatibility pack.
<table><tr><td>True</td><td>Documents are made compatible with Word 2003

Please note:

When these documents are downloaded all CopyLabel ContentControls are replaced with Bookmarks. When they are uploaded back to the Server these Bookmarks are replaced with the CopyLabel ContentControls that were removed on download.</td></tr><tr><td>False</td><td>Documents remain unchanged.</td></tr></table></td></tr>
<tr><td>result_error_xml_path</td><td>This attribute is only interpreted when the attribute client_runtime_mode has the value false.

If a path is specified here, a text file containing the result XML is written to the specified path for every process call where the result code is not null (non-zero values).

The name of the text file is composed as follows:

<code>[processId]_[timestamp in the form of yyyyMMdd_HHmmssffffff].txt</code>

Example:

<code>29111f09-e33f-4782-b071-</code></td></tr>
</table>

	88a665d23b67_20110922_1522384641250.txt **Example Configurations:** gmail<mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="\\ServerName\ShareName\FolderName .... />" or <mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="c:\temp\mws" .... />" or <mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="%MyResultErrorXmlFolder%" .... />" Default value is **Empty string** when the attribute is not specified. No text files are written when the attribute is not specified or the value passed is an empty string. If the specified path is not found or could not be created when the service is started, the **MWSServiceHost** cannot start and throws the following error: "runtime_process";Error;0;;"ModusSuite.MWS.Types.MwsException: Category: <mws>, Token: <ResultErrorXmlPathNotFound> Message: The file "\\ServerName\ShareName\FolderName" was not found. The specified path was not found. Please check the settings in the configuration file for the attribute 'result_error_xml_path' in the element 'mws' If the specified path is no longer available at run time, a corresponding error message is traced containing the contents of the file that could not be written.
throw_error_on_missing_content_control	Specifies whether document generation is aborted with an error or a document is generated with warnings when ContentControls of text block variables are missing. Default value is **true** when the attribute is not specified. True: Document generation is aborted with an error message. False: A warning is traced and document generation continued.

Load DB Alias from a File

Requirements

The following requirements must be met for a DB Alias to be loaded from a file:

In the section appSettings the value **loadDbAliasFromLocalFile** must be set to **True**.

The DB Alias object to be loaded from a local file must reside in the repository system even though it is not actually loaded from the repository.

The following configuration settings must be present:

```
<configuration>
   <configSections>
      …
      <section name="mlsdatabase"
type="ModusSuite.Common.SystemFramework.AliasConfigurationSection,
ModusSuite.Common.SystemFramework"/>
      ……
   </configSections>
   <appSettings>
      …
      <add key="loadDbAliasFromLocalFile" value="true" />
      …
   </appSettings>
   …
   <mlsdatabase configSource="mlsdatabase.config" />
…
</configuration>
```

Structure of the local file

The file specified in the attribute **configSource** must have the following structure:

```
<mlsdatabase>
   <aliaslist>
      <alias name="DataDbAlias" code="uwc…==" connectionstring="Persist
Security Info=True; …. " dbtype="SQLMS" />
      <alias name="JOBDATA" code="uwc…==" connectionstring="User ID=sa;…"
dbtype="SQLMS" />
      </aliaslist>
</mlsdatabase>
```

A simple way of creating a file with the structure required is to use the export function in the DB Alias administration tools.

Logging

If a DB Alias is loaded from a file, the action is traced by a corresponding message logged at the **Information** level.

Configuring Windows Services

Configuring Wait Time for Starting / Stopping a Windows Service

When Windows services are started or stopped using the service control manager (SCM or **net start/stop**) the operating systems waits for a specific period of time to allow the action to be completed.

If the services does not start / stop within this set time, the services controller aborts the action with an error.

Depending on the process being run as a Windows service, the time needed to start / stop the service may be longer than the standard operating system time out allows.

If required, the following settings in the respective service configuration files can extend the standard time out:

1. The number of millisecond(s) to be added for starting the service.
   Uses the key **AdditionalTimeOnStart**.
   The value is the additional time required expressed in milliseconds
2. The number of millisecond(s) to be added for stopping the service.
   Uses the key **AdditionalTimeOnStop**.
   The value is the additional time required expressed in milliseconds

Example:

In the following example start and stop require 10 seconds longer.

```
<configuration>
…
  <appSettings>
    <add key="AdditionalTimeOnStart" value="10000" />
    <add key="AdditionalTimeOnStop" value="10000" />
  </appSettings>
…
</configuration>
```

Accessing Network Drives

General

If a process needs access to a network drive, this can be implemented in one of two ways:

1. Using a UNC path, e.g.

```
\\NetData\odin
```

This option needs no further configuration.

2. Using a drive letter

This option requires that the necessary drive mapping be defined in the relevant service configuration file.

Using Drive Letters with Network Drives

Using Drive Letters with Network Drives

Drive mappings apply for a specific Windows account.

If a Windows service is executed by a Windows account the operating system expects the service to know the drive letters of the account used. This, however, is not the case.

The reason is that drives connected using **net use** are only available to a Windows account in an interactive Windows session.

A login session is created for the Windows account when the Windows service is started. This is not an interactive session, but a special login session to execute services. As a result, connected drives are not available in this session.

Document Composition service applications have been enhanced to enable the use of drive letters despite this restriction.

When a Document Composition service starts up, it checks their configuration file.

If the configuration file contains information on drive letter mapping to a network drive, it makes a connection to it.

If the network drive cannot be accessed, the service will not start.

Any information on this action can then be found in the log file and the event log.

When a service stops, the connection to the network drive is closed.

To map a letter to a network drive, the configuration file of the respective Document Composition service must contain the following elements:

- The section **driveMapping** in the XML element **configSections**.
- The element **driveMapping**.

Numerous connections can be configured.

In the following example, the network drive **\\NetData\odin** is mapped to the drive letter **X**:.

The network drive **\\NetData\mws** is mapped to the drive letter **Y**.

```
<configuration>
...
   <configSections>
      ...
      <section name="driveMapping"
type="ModusSuite.Runtime.SystemFramework.DriveMappingConfiguration,
ModusSuite.Runtime.SystemFramework"/>
      …
   </configSections>
   ...
   <xdata configSource="xdata.config"/>
   <odinSettings configSource="odin.config"/>
   <driveMapping>
      <drives>
         <drive localName="x:" remoteName="\\NetData\odin" />
         <drive localName="y:" remoteName="\\NetData \mws" />
      </drives>
```

```
   </driveMapping>
   ...
<configuration>
```

Configuring MWS Services

The Configuration File Modus_MWS.exe.config

Example Configuration

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="modusruntime"
type="ModusSuite.Runtime.SystemFramework.RuntimeConfiguration,
ModusSuite.Runtime.SystemFramework" />
    <section name="mws" type="ModusSuite.MWS.SystemFramework.Configuration,
ModusSuite.MWS.SystemFramework" />
    <section name="xdata"
type="ModusSuite.Xdata.SystemFramework.XdataConfiguration,
ModusSuite.Xdata.SystemFramework" />
    <section name="remotecontrol"
type="ModusSuite.Runtime.SystemFramework.RemoteControlConfiguration,
ModusSuite.Runtime.SystemFramework" />
  </configSections>
  <appSettings>
    <add key="wordtemplate" value="%PDCdir%Normal.dotm" />
    <add key="rws_config" value="repository.config" />
    <add key="odinparameterfile" value="odinparameter.config" />
    <add key="disableDebugNamedPipe" value="1" />
    <add key="rws" value="http://localhost:8010/rws" />
    <add key="ows" value="http://localhost:8010/ows/owsrepository" />
    <add key="sts" value="http://localhost:8000/sts" />
    <add key="login" value="http://localhost:8010/mur/login" />
    <add key="mur" value="http://localhost:8010/mur/data" />
    <add key="Authenticate" value="0" />
    <add key="license" value="http://localhost:8010/mur/license" />
    <add key="compilewordml2007only" value="true"/>
  </appSettings>
  <modusruntime>
    <runtimeservices>
      <service name="repository"
assembly="ModusSuite.Runtime.RepositoryService"
type="ModusSuite.Runtime.RepositoryService" />
      <service name="threading"
assembly="ModusSuite.Runtime.ThreadingService"
type="ModusSuite.Runtime.ThreadingService" />
```

```
      <service name="timer" assembly="ModusSuite.Runtime.TimerService"
type="ModusSuite.Runtime.TimerService" />
      <service name="debuging" assembly="ModusSuite.Runtime.DebugService"
type="ModusSuite.Runtime.DebugService" />
      <service name="remotecontrol"
assembly="ModusSuite.Runtime.RemoteControlService"
type="ModusSuite.Runtime.RemoteControlService" />
      <service name="mws" assembly="ModusSuite.Runtime.MWSRuntimeService"
type="ModusSuite.Runtime.MWSRuntimeService" />
      <service name="mwsrws"
assembly="ModusSuite.Runtime.MWSRepositoryRuntimeService"
type="ModusSuite.Runtime.MWSRepositoryRuntimeService" />
      <service name="mwsbasic"
assembly="ModusSuite.Runtime.MWSRuntimeService"
type="ModusSuite.Runtime.MWSRuntimeServiceBasic" />
      <service name="mwsrwsbasic"
assembly="ModusSuite.Runtime.MWSRepositoryRuntimeService"
type="ModusSuite.Runtime.MWSRepositoryRuntimeServiceBasic" />
    </runtimeservices>
  </modusruntime>
  <remotecontrol address="http://127.0.0.1:4721/RemoteControlService" />
  <mws systemoid="dm" dbalias="odin_mws" process="MWS_Standard"
client_runtime_mode="false" protect_document="false" />
  <xdata configSource="xdata.config" />
  <system.diagnostics>
    <sources>
      <source name="mws" switchName="mws_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <remove name="Default" />
          <add name="tracefile_listener" />
        </listeners>
      </source>
      <source name="rws" switchName="rws_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <remove name="Default" />
          <add name="tracefile_listener" />
        </listeners>
      </source>
      <source name="runtime_process" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
```

```
      <source name="runtime_components" switchName="runtime_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
      <source name="monalisa" switchName="monalisa_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
      <source name="odin" switchName="odin_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
      <!-- This source is used by WCF -->
      <source name="System.ServiceModel" switchValue="Critical,Error"
propagateActivity="true">
        <listeners>
          <add name="sdt"
type="ModusSuite.Common.SystemFramework.XmlDailyTraceListener,
ModusSuite.Common.SystemFramework" initializeData="modusMWS.svclog" />
        </listeners>
      </source>
      <source name="textsystem" switchName="textsystem_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
      <source name="xdata" switchName="xdata_switch"
switchType="System.Diagnostics.SourceSwitch">
        <listeners>
          <add name="tracefile_listener" />
          <remove name="Default" />
        </listeners>
      </source>
    <sources>
    <switches>
      <!--value="Error": only log error messages  -->
```

```
      <!--value="Off": disable logging -->
      <add name="rws_switch" value="Error" />
      <add name="mws_switch" value="Error" />
      <add name="runtime_switch" value="Error" />
      <add name="monalisa_switch" value="Error" />
      <add name="odin_switch" value="Error" />
      <add name="textsystem_switch" value="Information" />
      <add name="xdata_switch" value="Error" />
    </switches>
    <sharedListeners>
      <add name="eventlog_listener"
type="System.Diagnostics.EventLogTraceListener" initializeData="Application">
        <!-- Filter makes sure only error messages are written to the event
log -->
        <filter type="System.Diagnostics.EventTypeFilter"
initializeData="Error" />
      </add>
      <add name="tracefile_listener"
type="ModusSuite.Common.SystemFramework.DailyTraceListener,
ModusSuite.Common.SystemFramework" delimiter=";"
initializeData="modus_MWS_log.txt" traceOutputOptions="DateTime" />
    </sharedListeners>
    <trace autoflush="true" />
  </system.diagnostics>
  <system.serviceModel>
    <diagnostics wmiProviderEnabled="false">
      <messageLogging logMalformedMessages="false"
logMessagesAtServiceLevel="false" logEntireMessage="false"
logMessagesAtTransportLevel="false" />
    </diagnostics>
    <services>
      <service name="MWSRepositoryService"
behaviorConfiguration="ModusSuiteServiceBehaviour">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8010/mws" />
          </baseAddresses>
        </host>
        <endpoint address="mwsrepository" binding="wsHttpBinding"
bindingConfiguration="TokenBinding" contract="IMWSRepositoryService" />
      </service>
      <service name="MWSRepositoryServiceBasic"
behaviorConfiguration="BasicBehaviour">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8010/mwsbasic" />
          </baseAddresses>
```

```
        </host>
        <endpoint address="mwsrepository" binding="basicHttpBinding"
bindingConfiguration="BasicBinding" contract="IMWSRepositoryService" />
      </service>
      <service name="ModusSuite.MWS.MWSProcessService"
behaviorConfiguration="ModusSuiteServiceBehaviour">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8011/mws" />
          </baseAddresses>
        </host>
        <endpoint address="mwsprocess" binding="wsHttpBinding"
bindingConfiguration="TokenBinding"
contract="ModusSuite.MWS.Types.IMWSProcessService" />
      </service>
      <service name="ModusSuite.MWS.MWSProcessServiceBasic"
behaviorConfiguration="BasicBehaviour">
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8011/mwsbasic" />
          </baseAddresses>
        </host>
        <endpoint address="mwsprocess" binding="basicHttpBinding"
bindingConfiguration="BasicBinding"
contract="ModusSuite.MWS.Types.IMWSProcessServiceBasic" />
      </service>
      <service name="ModusSuite.Runtime.RemoteControlService">
        <endpoint address="http://127.0.0.1:4721/RemoteControlService"
binding="wsDualHttpBinding" bindingConfiguration="RemoteControlBinding"
contract="ModusSuite.Runtime.RemoteControlContract.IRuntimeControlService" />
      </service>
      <service name="ModusSuite.Runtime.DebugService">
        <endpoint address="http://127.0.0.1:8012/RemoteDebugger"
binding="wsDualHttpBinding" bindingConfiguration="DebugBinding"
contract="ModusSuite.Runtime.Types.IDebugService" />
      </service>
    </services>
    <bindings>
      <wsHttpBinding>
        <binding name="TokenBinding" maxReceivedMessageSize="2147483600"
receiveTimeout="Infinite" maxBufferPoolSize="2147483600">
          <readerQuotas maxStringContentLength="2147483647"
maxArrayLength="2147483647" />
          <security mode="Message">
            <message clientCredentialType="IssuedToken" />
          </security>
        </binding>
        <!--  Settings for communication with repository (rws)
```

```
              DO NOT DELETE the binding element. Element is read when the
channel
              is created -->
        <binding name="rws" sendTimeout="00:05:00" />
      </wsHttpBinding>
      <wsDualHttpBinding>
        <binding name="DebugBinding" maxReceivedMessageSize="2147483600"
receiveTimeout="Infinite" sendTimeout="Infinite"
maxBufferPoolSize="2147483600">
          <security mode="None" />
          <readerQuotas maxStringContentLength="2147483647"
maxArrayLength="2147483647" />
        </binding>
        <binding name="RemoteControlBinding"
clientBaseAddress="http://127.0.0.1:8000/RemoteControlClient/"
receiveTimeout="Infinite" sendTimeout="Infinite">
          <security mode="None" />
        </binding>
      </wsDualHttpBinding>
      <basicHttpBinding>
        <binding name="BasicBinding" maxBufferPoolSize="2147483600"
maxReceivedMessageSize="2147483600" receiveTimeout="Infinite">
          <security mode="None" />
          <readerQuotas maxStringContentLength="2147483647"
maxArrayLength="2147483647" />
        </binding>
      </basicHttpBinding>
    </bindings>
    <behaviors>
      <serviceBehaviors>
        <behavior name="ModusSuiteServiceBehaviour">
          <serviceDebug includeExceptionDetailInFaults="true" />
          <!--
          The serviceMetadata behavior enables metadata (e.g. WSDL, Policy)
publishing.
          This configuration enables publishing of such data over HTTP GET.
          -->
          <serviceMetadata httpGetEnabled="true" />
          <serviceCredentials>
            <!--
            The serviceCredentials behavior enables definition of validation
parameters for issued tokens.
            This configuration adds the "modusOne.server" certificate to a
list of known certificates. This
            means that the service will accept tokens issued by "STS".
            -->
            <issuedTokenAuthentication>
```

```
              <knownCertificates>
                <add storeLocation="LocalMachine" storeName="TrustedPeople"
x509FindType="FindBySubjectDistinguishedName" findValue="CN=modusOne.server"
/>
              </knownCertificates>
            </issuedTokenAuthentication>
            <!--
            The serviceCredentials behavior enables definition of a service
certificate.
            A service certificate is used by a client to authenticate the
service and provide message protection.
            This configuration references the "modusOne.service" certificate
installed during setup of the sample.
            -->
            <serviceCertificate storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectDistinguishedName" findValue="CN=modusOne.service"
/>
          </serviceCredentials>
          <serviceThrottling maxConcurrentCalls="1000000"
maxConcurrentInstances="1000000" maxConcurrentSessions="1000000" />
        </behavior>
        <behavior name="BasicBehaviour">
          <serviceDebug includeExceptionDetailInFaults="true" />
          <serviceMetadata httpGetEnabled="true" />
          <serviceThrottling maxConcurrentCalls="1000000"
maxConcurrentInstances="1000000" maxConcurrentSessions="1000000" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
  </system.serviceModel>
</configuration>
```

appSettings

```
<appSettings>
   <add key="wordtemplate"
value="%APPDATA%\Microsoft\Templates\normal.dotm"/>
   <add key="compilewordml2007only" value="true"/>
</appSettings>
```

Possible Keys

Attribute	Definition
wordtemplate	Specifies which normal.dotm the server uses. For more information, see the following chapters of the PDC.Studio manual:

	• Perceptive Document Composition Studio > Administration > System Administration > Assign a Word Template • Perceptive Document Composition Studio > Word Templates	
compilewordml2007only	True	The key must be set to **True** for installations using both Word 2010 and Word 2003 with compatibility pack.
	False	The key must be set to **False** for installations using only Word 2010. Standard value is **False**, if key is missing.

mws

```
<mws systemoid="dm" dbalias="odin_mws" process="MWS_Standard"
client_runtime_mode="false" protect_document="false" />
```

Attributes of the Element mws

Attribute	Definition	
systemoid	ObjectID of the system where the MWS process and DB Alias are stored.	
dbalias	Name of the DB Alias used to access the MWS database.	
process	The name of the MWS process that must reside in the system specified in the attribute **systemoid**.	
client_runtime_mode	Specifies whether the MWS runs on a Client (Composition Studio) or on the Server	
	True	MWS runs in Composition Studio (on the client). Please note: It is not the process defined in the above system that is executed here. Instead, the application Composition Studio creates an internal process to test documents. The value of the attribute **systemoid** should be set to **DEBUG OID**. The attribute **dbalias** is not interpreted.
	False	MWS runs on the server.

<table>
<tr><td>protect_document</td><td>This attribute is only interpreted when the attribute client_runtime_mode has the value false.
The attribute applies to documents with password protection. It specifies whether document protection is activated directly by the server after documents have been generated.
Default value is false when the attribute is not specified.<table><tr><td>True</td><td>Document protection is activated by the MWS Server immediately after generation</td></tr><tr><td>False</td><td>Document protection is activated by the Client (Document Composition clients or third-party clients).</td></tr></table></td></tr>
<tr><td>set_processtitle_using_poolvar</td><td>Specifies the name of a variable in the Systempool to be read when a process is closed or forwarded and set as the value for Process Title.
Default value is Empty string when the attribute is not specified.</td></tr>
<tr><td>enableWord2003</td><td>Default value is false when the attribute is not specified.
Specifies whether documents are made compatible with Word 2003 before being downloaded and displayed by a Client using Word 2003 with compatibility pack.<table><tr><td>True</td><td>Documents are made compatible with Word 2003.
Please note:
When these documents are downloaded all CopyLabel ContentControls are replaced with Bookmarks. When they are uploaded back to the Server these Bookmarks are replaced with the CopyLabel ContentControls that were removed on download.</td></tr><tr><td>False</td><td>Documents remain unchanged.</td></tr></table></td></tr>
<tr><td>result_error_xml_path</td><td>This attribute is only interpreted when the attribute client_runtime_mode has the value false.
If a path is specified here, a text file containing the result XML is written to the specified path for every process call where the result code is not null (non-zero values).
The name of the text file is composed as follows:
<code>[processId]_[timestamp in the form of yyyyMMdd_HHmmssffffff].txt</code>
Example:</td></tr>
</table>

	29111f09-e33f-4782-b071-88a665d23b67_20110922_1522384641250.txt **Example Configurations:** gmail<mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="\\ServerName\ShareName\FolderName .... />" or <mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="c:\temp\mws" .... />" or <mws systemoid="mws" dbalias="MWS" process="MWS_Standard" client_runtime_mode="false" result_error_xml_path="%MyResultErrorXmlFolder%" .... />" Default value is **Empty string** when the attribute is not specified. No text files are written when the attribute is not specified or the value passed is an empty string. If the specified path is not found or could not be created when the service is started, the **MWSServiceHost** cannot start and throws the following error: "runtime_process";Error;0;;"ModusSuite.MWS.Types.MwsException: Category: <mws>, Token: <ResultErrorXmlPathNotFound> Message: The file "\\ServerName\ShareName\FolderName" was not found. The specified path was not found. Please check the settings in the configuration file for the attribute 'result_error_xml_path' in the element 'mws' If the specified path is no longer available at run time, a corresponding error message is traced containing the contents of the file that could not be written.
throw_error_on_missing_content_control	Specifies whether document generation is aborted with an error or a document is generated with warnings when ContentControls of text block variables are missing. Default value is **true** when the attribute is not specified. True: Document generation is aborted with an error message. False: A warning is traced and document generation continued.

Configuring Communication

General

The product Perceptive Document Composition consists of a number of server-side services and clients.

Client and server services communicate via SOAP web service interfaces made available on the basis of the Microsoft Windows Communication Foundation (WCF).

To ensure that client and server communication can take place, clients need to know the network addresses of server-side services.

As communication is done using the HTTP protocol, a server address looks like that of a typical web page:

http://Hostname:Port/sts

The following sections look at each individual Web Service interface and explain in detail the relevant configuration settings for client and server.

Important:

Communication between server and client can only be established when the same network name and port is defined in the respective client and server settings!

sts – Security Token Service

Client Configuration sts

Specify the service address using the XML element

```
<add key="sts" value="http://localhost:8000/sts" />
```

inside the XML element **<appSettings>.**

The **value** entered must be the **host base address** of the service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8000:

```
<appSettings>
  <add key="sts" value="http://Perceptive:8000/sts" />
...
</appSettings>
```

Server Configuration sts

The specific functions of this service are provided by the application **modus_core**.

Individual settings for the server endpoint are stored in the file **modus_core.exe.config** and configured in the following XML element:

```
<system.serviceModel>
  <services>
    ...
    <service behaviorConfiguration ="STSBehaviour" name="STS" >
```

```
      <host>
        <baseAddresses>
          <add baseAddress ="http://Perceptive:8000/sts" />
        </baseAddresses>
      </host>
      <endpoint address ="windows" ...  />
      <endpoint address ="username" ...  />
    </service>
    ...
  </services>
<system.serviceModel>
```

mur - ModusUserRepository

Client Configuration mur

Specify service addresses using the following XML elements:

```
    <add key="login" value="http://localhost:8010/mur/login" />
    <add key="mur" value="http://localhost:8010/mur/data" />
```

inside the XML element **<appSettings>.**

The **value** of the key **login** is the login address. The **value** of the key **mur** is the data address.

These addresses are created from the host base address and the endpoint address of the corresponding service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="login" value="http://Perceptive:8010/mur/login" />
  <add key="mur" value="http://Perceptive:8010/mur/data" />
</appSettings>
```

Server Configuration mur

The specific functions of this service are provided by the application **modus_core**.

Individual settings for the server endpoint are stored in the file **modus_core.exe.config** and configured in the following XML element:

```
<system.serviceModel>
  <services>
    ...
    <service name="ModusUserRepositoryService"
             behaviorConfiguration ="ModusSuiteServiceBehaviour" >
      <host>
        <baseAddresses>
          <add baseAddress ="http://Perceptive:8010/mur" />
        </baseAddresses>
      </host>
      <endpoint address ="login".../>
```

```
      <endpoint address ="data".../>
    </service>
    ...
  </services>
<system.serviceModel>
```

dpws – DataProviderService

Client Configuration dpws

Specify the service address using the XML element

```
<add key="dpws" value="http://localhost:8010/dataprovider" />
```

inside the XML element **<appSettings>.**

The value entered must be the host base address of the service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="dpws" value="http://Perceptive:8010/dataprovider" />
...
</appSettings>
```

Server Configuration dpws

The specific functions of this service are provided by the application **modus_core**.

Individual settings for the server endpoint are stored in the file **modus_core.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
    <service name="ModusSuite.DataProvider.DataProviderService"
             behaviorConfiguration ="ModusSuiteServiceBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/dataprovider" />
         </baseAddresses>
       </host>
       <endpoint address ="dpws" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

rws – RepositoryWebService

Client Configuration rws

Specify the service address using the XML element

```
<add key="rws" value="http://localhost:8010/rws" />
```

inside the XML element <appSettings>.

The value entered must be the host base address of the service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="rws" value="http://Perceptive:8010/rws" />
...
</appSettings>
```

Server Configuration rws

The specific functions of this service are provided by the application **modus_core**.

Individual settings for the server endpoint are stored in the file **modus_core.exe.config** and configured in the following XML element:

```
<system.serviceModel>
  <services>
    ...
    <service name="ModusSuite.Repository.RepositoryWebService"
              behaviorConfiguration ="ModusSuiteServiceBehaviour" >
      <host>
        <baseAddresses>
          <add baseAddress ="http://Perceptive:8010/rws" />
        </baseAddresses>
      </host>
      <endpoint address ="repository" .../>
      <endpoint address ="transferjobmanager" .../>
      <endpoint address ="systemmanager" .../>
    </service>    ...
    ...
  </services>
<system.serviceModel>
```

ows and owsbasic – OdinWebService

Client Configuration ows

Specify the service address using the XML element

```
<add key="ows" value="http://localhost:8010/ows/owsrepository" />
```

inside the XML element **<appSettings>.**

The value for the key ows must be the owsrepository address.

This address is created from the host base address and the endpoint address of the corresponding service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="ows" value="http://Perceptive:8010/ows/owsrepository" />
...
</appSettings>
```

Server Configuration ows

The specific functions of this service are provided by the application **modus_ows**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_ows.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
   <service name="ModusSuite.OWS.OWSService"
            behaviorConfiguration ="ModusSuiteServiceBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/ows" />
         </baseAddresses>
      </host>
      <endpoint address ="owsrepository" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

Client Configuration owsbasic

The following WSDL URL can be used by a third party client (a JAVA client, for example) to retrieve the service definition and to consume the service.

```
http://localhost:8010/owsbasic?wsdl
```

This address is created from the host base address followed by ?wsdl.

The following example illustrates how to specify the WSDL URL of the service hosted on the computer with the network name Perceptive using IP port 8010:

```
http://Perceptive:8010/owsbasic?wsdl
```

Server Configuration owsbasic

The specific functions of this service are provided by the application **modus_ows**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_ows.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
     <service name="ModusSuite.OWS.OWSServiceBasic"
```

```
              behaviorConfiguration ="BasicBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/owsbasic" />
         </baseAddresses>
       </host>
        <endpoint address ="owsrepository" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

xws and xwsbasic – XdataWebService

Client Configuration xws

Specify the service address using the XML element

```
<add key="xws" value=" http://localhost:8010/xws/selection" />
```

inside the XML element **<appSettings>.**

The value for the key xws must be the selection address. This address is created from the host base address and the endpoint address of the corresponding service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="xws" value="http://Perceptive:8010/xws/selection" />
...
</appSettings>
```

Server Configuration xws

The specific functions of this service are provided by the application **modus_xws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_xws.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
   <service name="ModusSuite.Xdata.SelectionService"
            behaviorConfiguration ="ModusSuiteServiceBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/xws" />
         </baseAddresses>
       </host>
       <endpoint address ="selection" ... />
     </service>
```

```
   ...
 </services>
<system.serviceModel>
```

Client Configuration xwsbasic

The following WSDL URL can be used by a third party client (a JAVA client, for example) to retrieve the service definition and to consume the service.

```
http://localhost:8010/xwsbasic?wsdl
```

This address is created from the host base address followed by ?wsdl.

The following example illustrates how to specify the WSDL URL of the service hosted on the computer with the network name Perceptive using IP port 8010:

```
http://Perceptive:8010/xwsbasic?wsdl
```

Server Configuration xwsbasic

The specific functions of this service are provided by the application **modus_xws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_xws.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
     <service name="ModusSuite.Xdata.SelectionServiceBasic"
              behaviorConfiguration ="BasicBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/xwsbasic" />
         </baseAddresses>
      </host>
       <endpoint address ="selection" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

mws and mwsbasic – ModusWebService

Client Configuration mws

Specify the service address using the XML element

```
<add key="mws" value=" http://localhost:8011/mws/mwsprocess" />
```

inside the XML element **<appSettings>.**

The value for the key mws must be the mwsprocess address.

This address is created from the host base address and the endpoint address of the corresponding service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8011:

```
<appSettings>
  <add key="mws" value="http://Perceptive:8011/mws/mwsprocess" />
...
</appSettings>
```

Server Configuration mws

The specific functions of this service are provided by the application **modus_mws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_mws.exe.config** and configured in the following XML element:

<system.serviceModel>

```
 <services>
   ...
   <service name="ModusSuite.MWS.MWSProcessService"
            behaviorConfiguration ="ModusSuiteServiceBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8011/mws" />
         </baseAddresses>
      </host>
      <endpoint address ="mwsprocess" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

Client Configuration mwsbasic

The following WSDL URL can be used by a third party client (a JAVA client, for example) to retrieve the service definition and to consume the service.

```
http://localhost:8011/mwsbasic?wsdl
```

This address is created from the host base address followed by ?wsdl.

The following example illustrates how to specify the WSDL URL of the service hosted on the computer with the network name Perceptive using IP port 8011:

```
http://Perceptive:8011/mwsbasic?wsdl
```

Server Configuration mwsbasic

The specific functions of this service are provided by the application **modus_mws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_mws.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
```

```
   ...
     <service name="ModusSuite.MWS.MWSProcessServiceBasic"
              behaviorConfiguration ="BasicBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8011/mwsbasic" />
         </baseAddresses>
       </host>
        <endpoint address ="mwsprocess" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

mwsrws and mwsrwsbasic – ModusRepositoryWebService

Client Configuration mwsrws

Specify the service address using the XML element

```
<add key="mwsrws" value=" http://localhost:8010/mws/mwsrepository" />
```

inside the XML element **<appSettings>.**

The value for the key **mwsrws** must be the **mwsrepository** address.

This address is created from the host base address and the endpoint address of the corresponding service.

In the following example, the client communicates with the service hosted on the computer with the network name Perceptive using IP port 8010:

```
<appSettings>
  <add key="mwsrws" value="http://Perceptive:8010/mws/mwsrepository" />
...
</appSettings>
```

Server Configuration mwsrws

The specific functions of this service are provided by the application **modus_mws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_mws.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
   <service name="MWSRepositoryService"
            behaviorConfiguration ="ModusSuiteServiceBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/mws" />
         </baseAddresses>
```

```
      </host>
      <endpoint address ="mwsrepository" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

Client Configuration mwsrwsbasic

The following WSDL URL can be used by a third party client (a JAVA client, for example) to retrieve the service definition and to consume the service.

```
http://localhost:8010/mwsbasic?wsdl
```

This address is created from the host base address followed by ?wsdl.

The following example illustrates how to specify the WSDL URL of the service hosted on the computer with the network name Perceptive using IP port 8010:

```
http://Perceptive:8010/mwsbasic?wsdl
```

Server Configuration mwsrwsbasic

The specific functions of this service are provided by the application **modus_mws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_mws.exe.config** and configured in the following XML element:

```
<system.serviceModel>
 <services>
   ...
     <service name="MWSRepositoryServiceBasic"
              behaviorConfiguration ="BasicBehaviour" >
       <host>
         <baseAddresses>
           <add baseAddress ="http://Perceptive:8010/mwsbasic" />
         </baseAddresses>
      </host>
       <endpoint address ="mwsrepository" ... />
     </service>
   ...
 </services>
<system.serviceModel>
```

RemoteControlService

General

The standard setup routine automatically adds settings for **Remote Control Client** for the following Windows services

- modus_Core
- modus_mws

- modus_ows
- modus_xws

in the files **PDC.Studio.exe.config and RemoteControl.config** and also adds default settings for Remote Control Services in the corresponding service configuration files (modus_Core.exe.config, modus_MWS.exe.config, etc.). These only require modification with regard to host name of servers and clients.

The following example illustrates the settings required for an additional Windows service:

Service Name: PDCProcess_SVC

ServiceHost: WindowsServiceHostModus.exe (renamed copy of PDC.WindowsServiceHost.exe)

Client Configuration (Composition Studio)

The Remote Control configuration file is specified in the configuration source **remotecontrol** in the configuration file **PDC.Studio.exe.config**.

```
<remotecontrol configSource="remotecontrol.config"/>
```

The actual configuration file **RemoteControl.config** defines which Remote Control Services are accessed on which server.

The following example uses a server with the name Perceptive and a Remote Control Service with the name RemoteCotrolServiceModus.

This is defined in the corresponding endpoint.

The Windows service is accessed via the name of the service (ModProcess_SVC).

```
<remotecontrol>
 <computers>
   <computer name="Perceptive" display="PDC Server">
     <services>
       ...
       <service name="ModProcess_SVC"
                display="PDC Process Service"
                runtime="true"
                endpoint="RemoteControlServiceModus"/>
       ...
     </services>
   </computer>
  </computers>
</remotecontrol>
```

In the configuration file **PDC.Studio.exe.config**, the XML element **<client>** defines the endpoints used.

In the following example, the client uses IP port 4712 to communicate with the service DebugService hosted on the computer with the network name Perceptive.

The Remote Control Service is defined by its endpoint address.

```
<system.serviceModel>
  ...
  <client>
    <endpoint name="RemoteControlServiceModus"
```

```
              address="http://Perceptive:4712/RemoteControlService"
              binding="wsDualHttpBinding"
              bindingConfiguration="Binding1"
              contract="ModusSuite.Runtime.RemoteControlContract.IRuntimeCont
rolService"/>
   ...
  </client>
  ...
</system.serviceModel>
```

The associated binding (here Binding1) is used to specify the client base address the server uses to communicate with the client.

In the following example, the client (DebugClient in Studio) runs on the computer with the network name Perceptive_Client and uses the IP port 8000:

```
</system.serviceModel>
 ...
 <bindings>
   <wsDualHttpBinding>
     <binding name="Binding1"
        clientBaseAddress="http://Perceptive_Client:8000/RemoteControlClient"
        ...
    </binding>
...
   </wsDualHttpBinding>
...
 </bindings>
...
</system.serviceModel>
```

Server Configuration

The following example illustrates the settings required for an additional Windows service:

Service Name: ModProcess_SVC

ServiceHost: WindowsServiceHostModus.exe (renamed copy of PDC.WindowsServiceHost.exe)

This service is then provided by the Windows service **WindowsServiceHostModus**.

Individual settings for the server endpoint are stored in the file **WindowsServiceHostModus.config** and configured in the following XML element:

```
<remotecontrol address="http://Perceptive:4712/RemoteControlService" />
<system.serviceModel>
 <services>
   ...
   <service name="ModusSuite.Runtime.RemoteControlService">
       <!-- use base address provided by host -->
     <endpoint
       address="http://Perceptive:4712/RemoteControlService"
       binding="wsDualHttpBinding"
```

```
       bindingConfiguration ="RemoteControlBinding"
       contract="ModusSuite.Runtime.RemoteControlContract.IRuntimeControlServ
ice"
     />
   </service>
   ...
 </services>
<system.serviceModel>
```

Debug Service (RemoteDebugger)

General

The module

```
<ModuleInfo AssemblyFile="ModusSuite.Studio.RemoteDebugger.dll" />
```

must be loaded to enable debugging of scripts and conditions in a Document Composition Client.

The modules a Document Composition Client loads are defined in the configuration file **clientprofile.xml**.

Client Configuration (Document Composition Client)

The XML element **<client>** in the configuration file **PDC.Studio.exe.config** is used to define the endpoint.

A client only uses the endpoint name to connect to the debug service **ModusSuite.Runtime.DebugService**.

In the following example, the client uses IP port 8012 to communicate with the service DebugService hosted on the computer with the network name Perceptive.

The **Remote Control Service** is defined by its endpoint address.

```
<system.serviceModel>
 ...
 <client>
    <endpoint name="ModusSuite.Runtime.DebugService"
       address=”http://Perceptive:8012/RemoteDebugger”
       binding="wsDualHttpBinding"
       bindingConfiguration="DebugClientBinding"
       contract="ModusSuite.Runtime.Types.IDebugService" />
   </client>
...
 </client>
 ...
</system.serviceModel>
```

The **DebugClientBinding** specifies the client base address the server uses to communicate with the client.

In the following example, the client (DebugClient in Document Composition Client) runs on the computer with the network name Perceptive_Client and uses the IP port 4711:

```
</system.serviceModel>
 ...
 <bindings>
   <wsDualHttpBinding>
     <binding name="DebugClientBinding"
              clientBaseAddress="http://Perceptive_Client:4711/DebugClient"
              ...
    </binding>
...
   </wsDualHttpBinding>
...
 </bindings>
...
</system.serviceModel>
```

Server Configuration

The specific functions of this service are provided by the application **modus_mws**. The application is installed as a Windows service as part of the standard set up.

Individual settings for the server endpoint are stored in the file **modus_mws.exe.config**.

The following entry is required in the appSettings section:

```
<appSetting>
...
 <add key="disableDebugNamedPipe" value="1"/>
...
</appSetting>
```

In the following example, the Debug Service runs on the computer with the network name Perceptive and uses the IP port 8012:

```
<system.serviceModel>
 <services>
   ...
   <service name="ModusSuite.Runtime.DebugService">
     <endpoint
         address="http://Perceptive:8012/RemoteDebugger"
         binding="wsDualHttpBinding"
         bindingConfiguration ="DebugBinding"
         contract="ModusSuite.Runtime.Types.IDebugService"/>
   </service>
  ...
 <services>
<system.serviceModel>
```

Configuring Logging

General

This chapter examines various uses of logging that are possible. Logging can be understood as the function of capturing details of operative processes and any errors encountered.

The first section looks at the basic principles of logging. The second section uses an sample XML configuration file to explain how to configure logging.

Basics of Logging

Trace Sources

A trace source refers to something that is able to generate messages for logging.

In this sense, it refers to all functional units of Document Composition, including Studio and Clients, that independent of each other and at a different level of detail for each instance, can generate information for logging.

Available trace sources

Trace Sources are available for functional components.

For each of the following components listed here, the level of detail generated for each respective log can be configured independently from the others.

Component	Trace Source
Composition Studio	modusstudio
Document Composition Client	mwsclient
Monalisa	monalisa
Runtime	runtime_process
Runtime-Components	runtime_components
Textsystem	textsystem
Xdata	Xdata
MWS	mws
RWS	rws
Odin	Odin
DPWS	dpws

User Repository	mur

Log Level

Depending on where it is used and on specific requirements, logging can be configured to provide various levels of detail.

Configuration in this sense means you can specify the log level from **Error** (logs all errors) to **Information** (logs details of entire process flows).

Available log levels

Log Level	Definition
Off	Logging is deactivated.
Error	This level only logs errors
Warning	Use the log level Warning to log warnings and errors. Warnings are not errors but they indicate near-error states such as missing data. Missing data is not necessarily an error, but may need to be monitored.
Information	Logs information messages in addition to warnings and errors.
Verbose	Writes detailed information on all process activities to the log file.

Trace Listeners

Trace listeners are agents responsible for outputting log messages and for making logging possible in the first place.

They listen for messages from the various trace sources and output them as text messages to an individually defined output medium, e.g. in log files or to an output window.

Available Trace Listeners

Trace Listeners	Definition
DailyTraceListener	This trace listener creates a new log file every day at midnight. The content of this file is displayed in list form. Fields are separated by an individually defined separator character.
OutputWindowTraceListener	As Studio and Client also have their own output window, the trace listener can be configured to send messages to this window.

Logging Settings

Configuration Files for Logging

All settings relating to logging are made in the corresponding XML configuration files:

Application / Service	Configuration File
Composition Studio	PDC.Studio.exe.config
Document Composition Client	PDC.Client.exe.config
Core Service	modus_Core.exe.config
MWS Service	modus_MWS.exe.config
OWS Service	modus_OWS.exe.config
XWS Service	modus_XWS.exe.config

Additional XML configuration files that use logging, but where actual application usage is more optional.

Application	Configuration File
Repository Setup	RepositorySetup.exe.config
PDC Object Tool	PDC.Tool.exe.config
DocX-Compiler	DocXCompiler.exe.config
Runtime Console Host	PDC.ConsoleHost.exe.config
Windows Service Host	PDC.WindowsServiceHost.exe.config
WCF-Console	ModusSuite.WCF.Console.exe.config
Transfer Console	TransferConsole.exe.config
Monalisa Engine	MonalisaEngine.exe.config

Log File Storage Location

The storage location of log files is specified along with the definition of the listener in the section **<sharedListeners>**.

The attribute **initializeData**, which is normally only the name of the log file itself, can also be used to specify a qualified path to the file.

Example

```
c:\temp\PDC.Studio_Log.txt
```

If only the file name is specified, i.e. no path defined, the log file is written by default to the folder

```
%userprofile%\PDC
```

This ensures that the user has sufficient rights to create and write to the log file.

Exception

The only exception to this is the log file of a system service.

In this case, the log file is created in the Perceptive Document Composition installation directory.

Activating Logging

XML configuration files contain a section **<system.diagnostics>** followed by the subsection **<sources>**.

The elements below **<sources>** list the relevant trace sources for the respective functions and also their individual trace listeners and log levels.

The usual structure of a trace source entry is as follows:

```
<source name="…" switchName="…" switchType="System.Diagnostics.SwitchType">
   <listeners …>
      <add name="…"/>
      <remove name="Default"/>
   </listeners>
</source>
```

An interesting attribute of the element **<source>** is **switchName**.

This attribute refers to the switch used to turn logging on and off and to the log level each source is set to.

Switch Definition

The section **<switches>** is directly below **<sources>** and contains **<add>** elements. Here, the attribute **name** contains the switch name (this must be identical to the attribute **switchName** in **<source>**). The attribute **value** refers to the log level.

The same switch can, of course, be used for more than one trace source.

Listener Definition

The section **<switches>** is followed by the section **<sharedListeners>**. This is where trace listeners are specified.

Listeners are also added using an **<add>** element. Here, the attribute **name** must correspond to the value of the attribute **name** of the respective **add** entry in the section **<listeners>**.

Example Logging Configuration

The following is an example of a logging configuration in the file **PDC.Studio.exe.config** used to log MWS activities:

```
<sources>
   <source name="mws"
           switchName="runtime_switch"
           switchType="System.Diagnostics.SourceSwitch">
      <listeners>
         <add name="tracefile_listener"/>
         <remove name="Default"/>
```

```
      </listeners>
   </source>
<sources>

<switches>
   <add name="runtime_switch" value="Information" />
</switches>

<sharedListeners>
   <add name="tracefile_listener"
        type="ModusSuite.Common.SystemFramework.DailyTraceListener,
              ModusSuite.Common.SystemFramework"
              delimiter=";"
              initializeData="PDC.Studio_Log.txt"
              traceOutputOptions="DateTime"/>
</sharedListeners>
```

This example illustrates how the trace source **mws** is configured.

The switch **runtime_switch** sets the log level to **Information**. The listener with the name **tracefile_listener** uses the DailyTraceListener to write the output to a text file.

The name of the log file is **PDC.Studio_log.txt**.

Configuring User Management

UserRepository.config

This file contains general settings for the user repository.

Example Configuration

```
<?xml version="1.0" encoding="utf-8" ?>
   <userRepository
    systemOId ="dm"
    roleMapper = "Std_Mapping"
    userProfile = "Std_Profil"
    profileReadOption = "None"
    userStore = "Windows"
    />
```

Attributes of the Element userRepository

Attribute	Definition
systemOId	ObjectID of the system the RoleMapper and UserProfiles are loaded from after a user has been authorized successfully.

<table>
<tr><td>roleMapper</td><td>Name of the RoleMapper object used to link groups to Roles.</td></tr>
<tr><td>userProfile</td><td>Name of the UserProfile object used to link LDAP attributes to Document Composition attributes.</td></tr>
<tr><td>profileReadOption</td><td>Specifies whether user profiles and/or roles are read from LDAP.
Possible values:<table><tr><td>None</td><td>No profiles read from LDAP</td></tr><tr><td>User</td><td>User profiles are read from LDAP</td></tr><tr><td>Role</td><td>Role profiles are read from LDAP</td></tr><tr><td>All</td><td>All profiles are read from LDAP</td></tr></table>This setting is dependent on the parameter userStore. In this way, for example, it is possible to read users from Windows user management, but read profiles form LDAP.</td></tr>
<tr><td>userStore</td><td>Specifies where user and group information is obtained (LDAP or Windows user management).
Possible values:<table><tr><td>Windows</td><td>Information is read from Windows user management.</td></tr><tr><td>LDAP</td><td>Information is read from LDAP.</td></tr></table></td></tr>
</table>

UserRepository_Ldap.config

When LDAP is used, the actual access to LDAP is defined using this file.

Example Configuration

```
<?xml version="1.0" encoding="utf-8" ?>
   <userRepository_Ldap
      connectionString = "LDAP://delphi:389/DC=DMS-PE,DC=DE"
      connectionProtection = "None"
      adminUser = "CN=Administrator,CN=Users,DC=dms-pe,DC=de"
      adminPassword = "uwcQ9+OQ/SYqqmFWQNckIw=="
      groupContainerDN = ""
      userContainerDN = ""
      groupFilter = "(&amp;(objectCategory=group)(objectClass=group){0})"
      userFilter = "(&amp;(objectCategory=person)(objectClass=user){0})"
      attributeMapGroupName = "member"
      attributeMapUserName = "sAMAccountName"
      includeLocalGroups = "true"
  />
```

Attributes of the Element userRepository_LDAP

<table>
<tr><th>Attribute</th><th>Definition</th></tr>
<tr><td>connectionString</td><td>Connection information to LDAP Server
Syntax:
<code>LDAP://Host:Port/BaseDN</code>
<table>
<tr><td>Host</td><td>LDAP-Server</td></tr>
<tr><td>Port</td><td>Standard: 389
Using SSL: 636</td></tr>
<tr><td>BaseDN</td><td>Root-Node in LDAP where user and group containers are stored</td></tr>
</table>
Example:
<code>"LDAP://localhost:389/DC=DMS-PE,DC=DE"</code></td></tr>
<tr><td>connectionProtection</td><td>Possible values:
<table>
<tr><td>None</td><td>Standard Value</td></tr>
<tr><td>SignAndSeal</td><td>Secures the connection by digital signage and encryption of all packets sent to the Server.</td></tr>
<tr><td>SSL</td><td>Connection is made via SSL connection.</td></tr>
</table></td></tr>
<tr><td>adminUser</td><td>Admin-User account read from LDAP</td></tr>
</table>

	This user requires read rights for all user and group containers. **Example:** `"CN=Administrator,CN=Users,DC=dms-pe,DC=de"`
adminPassword	Password for this user - encrypted using the tool **Encoder.exe**.
groupContainerDN	Name of the container where groups are defined in LDAP. If this parameter is not empty, the search for the group starts in this container. If this parameter is empty, the search for the user starts in the BaseDN specified in the parameter **ConnectionString**. We recommend leaving this parameter empty ("").
userContainerDN	Name of the container where users are defined in LDAP. If this parameter is not empty, the search for the user starts in this container. If this parameter is empty, the search for the user starts in the BaseDN specified in the parameter **ConnectionString**. We recommend leaving this parameter empty ("").

groupFilter	Search filter to identify the LDAP class for groups.
	Example of Microsoft Active Directory:
	`(&(objectCategory=group)(objectClass=group){0})`
	The LDAP class for groups has the attribute **objectCategory=group** and **objectClass=group**.
	{0} is automatically replaced with the group name specified in the element **AttributeMapGroupName** e.g. (member=CN=Test User,CN=Users,DC=dms-pe,DC=de).
	The following filter can be used to find the group **Admin**:
	(&(objectCategory=group)(objectClass=group)(member=Admin)
	If, for example, the LDAP class only has the attribute **objectClass**, the definition looks as follows
	`(&(objectClass=group){0})`
	Note:
	The character "&" must be replaced by & in the configuration file.

userFilter	Search filter to identify the LDAP class for users. **Example of Microsoft Active Directory:** `(&(objectCategory=person)(objectClass=user){0})`

<table>
<tr><td></td><td>The LDAP class for users has the attribute objectCategory=person and objectClass=user.
{0} is automatically replaced with the user name specified in the element AttributeMapUserName e.g. (sAMAccountName=Miller).
The following filter can be used to find the user Miller:
(&(objectCategory=person)(objectClass=person)(sAMAccountName=Miller)

If, for example, the LDAP class only has the attribute objectClass, the definition looks as follows
<code>(&(objectClass=person){0})</code>
Note:
The character "&" must be replaced by & in the configuration file.</td></tr>
<tr><td>attributeMapGroupName</td><td>Attribute used to identify the groups a user belongs to, e.g. "member".</td></tr>
<tr><td>attributeMapUserName</td><td>Attribute used to identify a user, e.g. "sAMAccountName".</td></tr>
<tr><td>includeLocalGroups</td><td><table><tr><td>True</td><td>Include local groups belonging to global groups.</td></tr><tr><td>False</td><td>Does not include local groups.</td></tr></table></td></tr>
</table>

Rights and Permissions in Studio and Client

General Rights	No.	Definition	Studio Right	Client Right	Note
Start Studio	1	Start Studio	●		
Show Navigator	2	Show Navigator	●		
Show Client Navigator	3	Show Navigator in Client Navigator is not displayed when Client is started externally		●	
General Settings	4	User can activate / deactivate paging in Navigator. (Main menu->Options->General Settings)	●		
Change System	5	Allows a user to change	●		

		system.			
Client Change System	6	Allows a user to change system in the Client.		●	The right **Client Change System** requires right #3 **Show Client Navigator**
Change Folder	7	Authorization only checked when a start folder has been defined. Allows changing to a child folder but not to a parent folder. Navigator is empty if a start folder is not defined and a message follows that the start folder could not be set.	●		
Filter & Search Function	8	Allows filtering of object names as well as Full-Text Search in current system. If the right is missing, the **Filter** and **Search** Navigator tabs are hidden.	●		
Show Object Type Filter	9	Show object filter for the various object types in the navigator	●		
Transfer Systems	10	Systems can be transferred	●		
Synchronize Systems	11	Systems can be synchronized	●		
Administration	12	Perform administration tasks. If the right is missing, the Administration tab is not visible in the ribbon. Administration modules cannot be accessed.	●		

Download Created Documents	13	Download created documents	●	●	
Process List	14	Display process list		●	
List Forwarded Processes	15	Display list of forwarded processes		●	
Insert Documents Manually	16	Allow manual insertion of additional documents		●	
Insert Text Block Manually	17	Allow manual insertion of additional text blocks		●	
Assign Printer to all Documents in the Bundle	18	Assign / change printer for all documents in the bundle		●	
Assign Printer to Document	19	Assign / change printer for individual documents		●	
Print Parameter Assistant	20	Allows setting of output parameters		●	
Forward Process Manually	21	Forward processes to other users / user groups manually		●	
Create Documents	22	Create documents		●	
Data Retrieval	23	Execute data retrieval		●	
Print and Archive	24	Print and archive generated bundles		●	
Client Administration	25	Perform administration tasks. If the right is missing, the Administration tab is not visible in the ribbon. Administration modules cannot be accessed.		●	
Process List for Specified User	26	Allows the list of processes of a user set in the start parameter		●	

		listuser to be displayed			
Client Change Folder	27	Authorization only checked when a start folder has been defined. Allows changing to a child folder but not to a parent folder. Navigator is empty if a start folder is not defined and a message follows that the start folder could not be set.		•	
Client View Odin Processes	28	Display Odin processes		•	
Client View Odin Stacks	29	Display Odin stacks		•	
Client View Odin Open Envelopes	30	Displays open envelopes		•	
Client View Open Jobs	31	Displays open jobs		•	
Clear MWS Server Cache	32	Clear MWS Server Cache		•	
Assign / Remove External Document File	33	Replace bundle documents with external document files.		•	
Load / Remove External Text Block File	34	Replace document text blocks with external document files.		•	
Save Created Document As...	35	Save created documents to a directory. The user also required the right **Download created documents (13)**.		•	
Create Release Version	36	Create new release versions	•		
Permanently	37	Permanently delete objects from the recycle	•		

Delete Object		bin			
Test Bundles and Documents in Studio	3509	User can test documents and bundles in the Client integrated in the Studio.	●		
UI Rights					
Change font	2000	Change font		●	
Change Font Size	2001	Change font size		●	
Change Font Color	2002	Change font color		●	
Font Style Bold	2003	Change font style bold		●	
Font Style Italic	2004	Change font style italic		●	
Font Style Underlined	2005	Change font style underlined		●	
Change Text Alignment	2006	Change text alignment		●	
Print Document Preview	2007	Print the document preview		●	
Rights for Odin Processes					
Unlock Odin process	3000	User can unlock a locked process	●	●	
Reset Odin process	3001	User can change a process from **Locked** to **Rendition Created** or **Imported**.	●	●	
Create Odin Stack for Deleting/Printing in the Odin View **Process**	3002	User can create a stack for print / delete from process jobs	●	●	
Delete Odin process	3003	User can set processes to status **Delete**.	●	●	
Pause Odin	3004	User can lock a process	●	●	

process					
Create Odin Stack for Deleting/Printing in the Odin View **Open Jobs**.	3005	User can create a stack for print / delete from open jobs	●	●	
Rights for Odin View Stack					
Edit Odin stack	3500	User can edit stacks	●	●	
Release Odin stack	3501	User can release stacks	●	●	
Delete Odin stack	3502	User can delete stacks	●	●	
Reset Odin stack	3503	User can set status of stacks to **Wait**.	●	●	
Reset Odin stack to 'Streaming'	3504	User can reset stacks to **Streaming**.	●	●	
Lock Odin stack	3505	User can lock a stack	●	●	
Unlock Odin stack	3506	User can unlock a stack	●	●	
Reset Odin stack to	3507	User can reset a stack to a specific stack type	●	●	
Set Odin Job State	3508	User can reset jobs to a specific status	●	●	
Set Odin job printer	3510	User can change the printer of a job	●	●	
Remove Odin Envelope from Stack	3511	User can remove an envelope from a stack	●	●	
Remove Odin Job from Stack/Envelope	3512	User can remove a job from an envelope or stack	●		

Cache Management Configuration

The following configuration settings are required if objects are to be automatically removed from the cache when they are saved or transferred to another system.

1. Add the following entries to the section **<appSettings>** in the file **Modus_Core.exe.config** :

```
<appSettings>
...
   <add key="rwsMulticastIP" value="224.100.0.1" />
   <add key="rwsMulticastPort" value="9050" />
   <add key="rwsMulticastTTL" value="50" />
   <add key="rws_sendudpmessage" value="y" />
...
</appSettings>
```

Key	Definition
rwsMulticastIP	IP address of the multicast group
Table text	Table text The IP must be within the range of 224.0.0.0 - 239.255.255.255. Routers must support UDP multicasting. Objects will not be removed from the cache if the entry is missing
rwsMulticastPort	The port to be used rwsMulticastTTL Defines the **Time To Live** (TTL), i.e. the maximum number of routers a packet can pass before the router deletes the packet.
rws_sendudpmessage	y: Cache Management is activated. n: Cache Management is not activated.

2. Add the following lines to the section **<appSettings>** for every service whose cache is to be emptied, e.g. the file **Modus_MWS.exe.config**:

```
<appSettings>
...
   <add key="rwsMulticastIP" value="224.100.0.1" />
   <add key="rwsMulticastPort" value="9050" />
   <add key="rwsMulticastTTL" value="50" />
...
</appSettings>
```

DocXCompiler / PDC.Tool

If objects modified by the DocXCompiler or PDC.Tool should also be removed from the cache, enter the Keys **rwsMulticastIP**, **rwsMulticastPort** and **rwsMulticastTTL** ,as described above ,in the file **DocXCompiler.exe.config** or **PDC.Tool.exe.config**.

The command to remove an object from the cache is sent on the following events:

- An object is modified and saved in Studio
- After successfully completing a transfer
- After successfully revoking a transfer

Entries in the log file:

If the protocol switch **rws_switch** is set to the value **Information** or **Verbose** , the rws writes the following message in the Modus_Core log file for each object when the notification is sent:

```
"rws";Information;0;"sending udp command 'RemoveItemsFromCache' for item oid: [object oid] / system-oid: [system oid]
```

If the runtime cache should log receipt of the notification in the log file, set the **runtime_switch** of the corresponding EXE to the value **Information** or **Verbose**.

Receipt of notification to remove a cached object is logged as follows:

```
"runtime_process";Information;0;"[UDP Request (10.184.105.65)] Remove 1 item/s from cache, systemOId: [system oid]
```

Prepare Databases for Full-Text Search

Oracle

The required **Oracle Text** index is part of a standard Oracle configuration.

Depending on the operating system in use, it might be necessary to add an appropriate XPS extension.

MS SQL Server

Check the availability of the **SQL Full Text** index on your system.

Information on full-text search can be found here:

http://msdn.microsoft.com/en-us/library/ms403375%28SQL.90%29.aspx

http://technet.microsoft.com/en-us/library/ms142497.aspx

http://technet.microsoft.com/en-us/library/ms142499.aspx

IFilters

The search inside binary files requires IFilters, e.g. for .pdf and MS Office files:

Adobe PDF IFilter 9 (64-bit)	http://www.adobe.com/support/downloads/detail.jsp?ftpID=4025
Adobe PDF IFilter v6.0 (32-bit)	http://www.adobe.com/support/downloads/detail.jsp?ftpID=2611

Microsoft Office 2010 Filter Packs	http://www.microsoft.com/download/en/details.aspx?id=17062

1. Download and install the appropriate filters.
2. Execute this code on your SQL Server:

```
exec sp_fulltext_service 'load_os_resources', 1;
exec sp_fulltext_service 'verify_signature', 0;
go
```

3. Restart SQL server and check if IFilters are installed (.pdf, .docx for example):

```
exec sp_help_fulltext_system_components 'filter';
```

XPS Extension

Depending on the operating system in use, it might be necessary to add an appropriate XPS extension.

Index