Contents

Configuration file schema for the .NET Framework

10/29/2019 • 2 minutes to read • Edit Online

Configuration files are standard XML files that you can use to change settings and set policies for your apps. The .NET Framework configuration schema consists of elements that you can use in configuration files to control the behavior of your apps. The table of contents for this section reflects the schema hierarchy for startup, runtime, network, and other types of configuration settings.

For information about the types, format, and location of configuration files, see Configure apps. Familiarize yourself with XML if you want to edit the configuration files directly.

> **IMPORTANT**
> XML tags and attributes in configuration files are case-sensitive.

In this section

<configuration> Element
The top-level element for all configuration files.

<assemblyBinding> Element
Specifies assembly binding policy at the configuration level.

<linkedConfiguration> Element
Specifies a configuration file to include.

Startup Settings Schema
Elements that specify which version of the common language runtime to use.

Runtime Settings Schema
Elements that configure assembly binding and runtime behavior.

Network Settings Schema
Elements that specify how the .NET Framework connects to the internet.

Cryptography Settings Schema
Elements that map friendly algorithm names to classes that implement cryptography algorithms.

Configuration Sections Schema
Elements used to create and use configuration sections for custom settings.

Trace and Debug Settings Schema
Elements that specify trace switches and listeners.

Compiler and Language Provider Settings Schema
Elements that specify compiler configuration for available language providers.

Application Settings Schema
Elements that enable a Windows Forms or ASP.NET application to store and retrieve application-scoped and user-scoped settings.

App Settings Schema
Contains custom application settings, such as file paths, XML Web service URLs, or any other custom

configuration information for an application.

Web Settings Schema

Elements for configuring how ASP.NET works with a host application such as IIS. Used in *Aspnet.config* files.

Windows Forms Configuration Schema

All elements in the Windows Forms application configuration section, which includes customizations such as multi-monitor and high-DPI support.

WCF Configuration Schema

All elements that enable you to configure WCF service and client applications.

WCF Directive Syntax

Describes the `@ServiceHost` directive, which defines page-specific attributes used by the .svc compiler.

WIF Configuration Schema

All elements of the Windows Identity Foundation (WIF) configuration schema.

Related sections

Remoting Settings Schema

Describes the elements that configure client and server applications that implement remoting.

ASP.NET Settings Schema

Describes the elements that control the behavior of ASP.NET Web applications.

Web Services Settings Schema

Describes the elements that control the behavior of ASP.NET Web services and their clients.

Configuring .NET Framework Apps

Describes how to configure security, assembly binding, and remoting in the .NET Framework.

<configuration> element

8/22/2019 • 2 minutes to read • Edit Online

The root element in every configuration file used by the common language runtime and .NET Framework applications.

<configuration>

Syntax

```
<configuration>
  <!-- Configuration settings -->
</configuration>
```

Attributes

None

Parent element

None

Child elements

	DESCRIPTION
<assemblyBinding>	Specifies assembly binding policy at the configuration level.
<startup> Settings Schema	All elements in the startup settings schema.
<runtime> Settings Schema	All elements in the runtime settings schema.
<system.runtime.remoting> Settings Schema	All elements in the remoting settings schema.
<system.Net> Settings Schema	All elements in the network settings schema.
<cryptographySettings> Settings Schema	All elements in the crypto settings schema.
<configuration> Sections Schema	All elements in the configuration section settings schema.
Trace and Debug Settings Schema	All elements in the trace and debug settings schema.
ASP.NET Configuration Settings Schema	All elements in the ASP.NET configuration schema, which includes elements for configuring ASP.NET Web sites and applications. Used in *Web.config* files.
<webServices> Settings Schema	All elements in the Web services settings schema.

	DESCRIPTION
Web Settings Schema	All elements in the Web settings schema, which includes elements for configuring how ASP.NET works with a host application such as IIS. Used in *aspnet.config* files.

Remarks

Each configuration file must contain exactly one **<configuration>** element.

See also

- Configuration file schema for the .NET Framework

<assemblyBinding> element for <configuration>

8/22/2019 • 2 minutes to read • Edit Online

Specifies assembly binding policy at the configuration level.

<configuration>
<assemblyBinding>

Syntax

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <!-- Configuration files to include. -->
</assemblyBinding>
```

Attribute

	DESCRIPTION
xmlns	Required attribute. Specifies the XML namespace required for assembly binding. Use the string "urn:schemas-microsoft-com:asm.v1" as the value.

Parent element

	DESCRIPTION
<configuration>	The root element in every configuration file used by the common language runtime and .NET Framework applications.

Child element

	DESCRIPTION
<linkedConfiguration>	Specifies a configuration file to include.

Remarks

The **<linkedConfiguration>** element simplifies the management of component assemblies by allowing application configuration files to include assembly configuration files in well-known locations, rather than duplicating assembly configuration settings.

> **NOTE**
> The **<linkedConfiguration>** element is not supported for applications with Windows side-by-side manifests.

Example

The following example shows how to include a configuration file on the local hard disk:

```
<configuration>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <linkedConfiguration href="file://c:\Program Files\Contoso\sharedConfig.xml" />
  </assemblyBinding>
</configuration>
```

See also

- Configuration file schema for the .NET Framework

<linkedConfiguration> element

11/14/2019 • 2 minutes to read • Edit Online

Specifies a configuration file to include.

<configuration>
<assemblyBinding>
<linkedConfiguration>

Syntax

```
<linkedConfiguration href="URL of linked configuration file" />
```

Attribute

	DESCRIPTION
href	Required attribute. The URL of the configuration file to include. The only format supported for the **href** attribute is `file://`. Local files and UNC files are supported.

Parent element

	DESCRIPTION
<assemblyBinding> Element	Specifies assembly binding policy at the configuration level.

Child elements

None

Remarks

The **<linkedConfiguration>** element simplifies servicing for component assemblies. If one or more applications use an assembly that has a configuration file residing in a well-known location, the configuration files of the applications that use the assembly can use the **<linkedConfiguration>** element to include the assembly configuration file, rather than including configuration information directly. When the component assembly is serviced, updating the common configuration file provides updated configuration information to all applications that use the assembly.

> **NOTE**
> The **<linkedConfiguration>** element is not supported for applications with Windows side-by-side manifests.

The following rules govern the use of linked configuration files:

- The settings in included configuration files only affect loader binding policy and are used only by the loader. The included configuration files can have settings other than binding policies, but those settings don't have any effect.
- The only format supported for the `href` attribute is `file://`. Local files and UNC files are supported.
- There is no constraint on the number of linked configurations per configuration file.
- All linked configuration files are merged to form one file, similar to the behavior of the `#include` directive in C/C++.
- The **<linkedConfiguration>** element is allowed only in application configuration files; it's ignored in *Machine.config*.
- Circular references are detected and terminated. That is, if the **<linkedConfiguration>** elements of a series of configuration files form a loop, the loop is detected and stopped.

Example

The following example shows how to include configuration file from the local hard disk:

```
<configuration>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <linkedConfiguration href="file://c:\Program Files\Contoso\sharedConfig.xml"/>
  </assemblyBinding>
</configuration>
```

See also

- **<assemblyBinding>** Element
- Configuration file schema for the .NET Framework

<configSections> element for <configuration>

10/30/2019 • 2 minutes to read • Edit Online

Contains configuration section and namespace declarations.

<configuration>
<configSections>

Attributes

None

Parent element

	DESCRIPTION
<configuration>	The root element in every configuration file used by the common language runtime and .NET Framework applications.

Child elements

	DESCRIPTION
<section>	Contains a configuration section declaration.
<sectionGroup>	Defines a namespace for configuration sections.
<remove>	Removes a predefined section or section group.
<clear>	Clears all previously defined sections and section groups.

Remarks

If this element is in a configuration file, it must be the first child element of the **<configuration>** element.

Example

The following example shows how to define a configuration section and define settings for that section:

```
<configuration>
  <configSections>
    <section name="sampleSection"
             type="System.Configuration.SingleTagSectionHandler" />
  </configSections>
  <sampleSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<clear> element for <configSections>

10/30/2019 • 2 minutes to read • Edit Online

Clears all previously defined sections and section groups.

<configuration>
<configSections>
<clear>

Syntax

```
<clear/>
```

Attribute

	DESCRIPTION
name	Required attribute. Specifies the name of the section or section group to remove.

Parent element

	DESCRIPTION
<configSections> Element	Contains configuration section and namespace declarations.

Child elements

None

Remarks

The **<clear>** element removes all sections and section groups from your application that were defined earlier in the current configuration file or at a higher level in the configuration file hierarchy.

Example

This example defines a machine configuration file and an application configuration file and shows how to use the **<clear>** element in an application configuration file to clear sections previously defined in the machine configuration file.

The following machine configuration file code declares two sections, **<sampleSection>** and **<anotherSampleSection>**, which are read before the application configuration file:

```
<!-- Machine.config file -->
<configuration>
  <configSections>
    <section name="sampleSection"
             type="System.Configuration.SingleTagSectionHandler" />
    <section name="anotherSampleSection"
             type="System.Configuration.NameValueSectionHandler" />
  </configSections>
  <sampleSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

The following application configuration file code clears all previously declared sections. The application cannot use or retrieve settings in either of the sections that were declared in the machine configuration file. However, it can use settings from **<anotherSection>** because it comes after the **<clear>** element.

```
<!-- Application configuration file -->
<configuration>
  <configSections>
    <clear/>
    <section name="anotherSection"
             type="System.Configuration.NameValueSectionHandler" />
  </configSections>
  <anotherSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<remove> element for <configSections>

11/14/2019 • 2 minutes to read • Edit Online

Removes a predefined section or section group.

<configuration>
<configSections>
<remove>

Syntax

```
<remove name="section name or section group name" />
```

Attribute

	DESCRIPTION
name	Required attribute. Specifies the name of the section or section group to remove.

Parent element

	DESCRIPTION
<configSections> Element	Contains configuration section and namespace declarations.

Child elements

None

Remarks

You can use the **<remove>** element to remove sections and section groups from your application that were defined at a higher level in the configuration file hierarchy.

Example

The following example shows how to use the **<remove>** element in an application configuration file to remove a section previously defined in the machine configuration file.

The following machine configuration file code declares the section **<sampleSection>**:

```
<!-- Machine.config file -->
<configuration>
  <configSections>
    <section name="sampleSection"
             type="System.Configuration.SingleTagSectionHandler" />
  </configSections>
  <sampleSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

The following application configuration file code removes the **<sampleSection>** section. After removal, the application cannot retrieve the settings in **<sampleSection>**.

```
<!-- Application configuration file -->
<configuration>
  <configSections>
    <remove name="sampleSection"/>
  </configSections>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<sectionGroup> element for <configSections>

11/14/2019 • 2 minutes to read • Edit Online

Defines a namespace for configuration sections.

<configuration>
<configSections>
<sectionGroup>

Syntax

```
<sectionGroup name="section group name">
  <!-- Configuration sections -->
</sectionGroup>
```

Attribute

	DESCRIPTION
name	Required attribute. Specifies the name of the section group you are defining.

Parent element

	DESCRIPTION
<configSections> Element	Contains configuration section and namespace declarations.

Child elements

	DESCRIPTION
<section>	Contains a configuration section declaration.

Remarks

Declaring a section group creates a container tag for configuration sections and ensures that there are no naming conflicts with configuration sections defined by someone else. You can nest **<sectionGroup>** elements within each other.

Example

The following example shows how to declare a section group and declare sections within a section group:

```
<configuration>
  <configSections>
    <sectionGroup name="mySectionGroup">
      <section name="mySection"
               type="System.Configuration.NameValueSectionHandler,System" />
    </sectionGroup>
  </configSections>
  <mySectionGroup>
    <mySection>
      <add key="key1" value="value1" />
    </mySection>
  </mySectionGroup>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

Custom element for SingleTagSectionHandler

10/30/2019 • 2 minutes to read • Edit Online

Defines settings in a custom configuration section that is defined by a <section> element and uses the SingleTagSectionHandler class.

<configuration>
<sectionName>

Syntax

```
<sectionName key="value" key2="value2" ... />
```

Attributes

Attributes and attribute values are user defined.

Parent element

	DESCRIPTION
<configuration>	The root element in every configuration file used by the common language runtime and .NET Framework applications.

Child elements

None

Remarks

The **<sectionName>** element is a custom element defined by a **<section>** tag in the **<configSections>** element. The configuration system returns a IDictionary object when you call Configuration.GetSection(String).

Example

The following example declares a custom element called **<sampleSection>** that contains settings read by the SingleTagSectionHandler class:

```
<configuration>
  <configSections>
    <section name="sampleSection"
             type="System.Configuration.SingleTagSectionHandler" />
  </configSections>
  <sampleSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

Custom element for NameValueSectionHandler and DictionarySectionHandler

10/30/2019 • 2 minutes to read • Edit Online

Defines settings for custom configuration sections that use the NameValueSectionHandler and DictionarySectionHandler classes.

<configuration>
<sectionName>

Attributes

None

Parent element

	DESCRIPTION
<configuration>	The root element in every configuration file used by the common language runtime and .NET Framework applications.

Child elements

	DESCRIPTION
<add> for NameValueSectionHandler and DictionarySectionHandler	Adds custom application settings.
<remove> for NameValueSectionHandler and DictionarySectionHandler	Removes a previously defined setting.
<clear> for NameValueSectionHandler and DictionarySectionHandler	Clears all previously defined settings in a section.

Remarks

The **<sectionName>** element is a custom element defined by a **<section>** tag in the **<configSections>** element.

The following table shows the type of object the ConfigurationSettings.GetConfig method returns for each configuration section handler:

CONFIGURATION SECTION HANDLER	RETURN TYPE
NameValueSectionHandler	NameValueCollection
DictionarySectionHandler	IDictionary

Example

The following example shows how to declare sections that use the DictionarySectionHandler and NameValueSectionHandler classes.

The first custom element is **<dictionarySample>**, which contains settings read by the DictionarySectionHandler class in the `System.dll` assembly. The second custom element is **<mySection>**, which contains settings read by the NameValueSectionHandler class in the `System.dll` assembly.

```
<configuration>
  <configSections>
    <section name="dictionarySample" type="System.Configuration.DictionarySectionHandler,System" />
    <sectionGroup name="mySectionGroup">
      <section name="mySection" type="System.Configuration.NameValueSectionHandler,System" />
    </sectionGroup>
  </configSections>
  <dictionarySample>
    <add key="myKey" value="myValue" />
  </dictionarySample>
  <mySectionGroup>
    <mySection>
      <add key="key1" value="value1" />
    </mySection>
  </mySectionGroup>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<add> element for NameValueSectionHandler and DictionarySectionHandler

11/14/2019 • 2 minutes to read • Edit Online

Adds custom application settings. Each **<add>** tag contains a key/value pair.

<configuration>
<sectionName>
<add>

Syntax

```
<add key="key" value="value" />
```

Attributes

ATTRIBUTE	DESCRIPTION
key	Required attribute. Specifies the name of the setting.
value	Required attribute. Specifies the value of the setting.

Parent element

ELEMENT	DESCRIPTION
<sectionName> Element	Defines settings for custom configuration sections that use the NameValueSectionHandler and DictionarySectionHandler classes.

Child elements

None

Example

The following example shows how to define a custom configuration section and use the **<add>** element to put settings into the section:

```
<configuration>
  <configSections>
    <section name="dictionarySample" type="System.Configuration.DictionarySectionHandler,System" />
  </configSections>
  <dictionarySample>
    <add key="myKey" value="myValue" />
  </dictionarySample>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<clear> element for NameValueSectionHandler and DictionarySectionHandler

11/14/2019 • 2 minutes to read • Edit Online

Clears all previously defined settings in a section.

<configuration>
<sectionName>
<clear>

Syntax

```
<clear />
```

Attributes

None

Parent element

	DESCRIPTION
<sectionName> Element	Defines settings for custom configuration sections that use the NameValueSectionHandler and DictionarySectionHandler classes.

Child elements

None

Remarks

You can use the **<clear>** element to remove all settings from your application that were defined at a higher level in the configuration file hierarchy.

Example

This example defines a machine configuration file and an application configuration file and shows how to use the **<clear>** element in an application configuration file to clear sections previously defined in the machine configuration file.

The following machine configuration file code declares the section **<mySection>**:

```
<!-- Machine.config file -->
<configuration>
  <configSections>
    <section name="mySection" type="System.Configuration.NameValueSectionHandler,System" />
  </configSections>
  <mySection>
    <add key="key1" value="value1" />
    <add key="key2" value="value2" />
  </mySection>
</configuration>
```

The following application configuration file code removes all settings from **<mySection>**. The application cannot retrieve any of the settings that were declared in the in the **<mySection>** section of the machine configuration file.

```
<!-- Application configuration file -->
<configuration>
  <mySection>
    <clear/>
  </mySection>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<remove> element for NameValueSectionHandler and DictionarySectionHandler

11/14/2019 • 2 minutes to read • Edit Online

Removes a previously defined setting.

<configuration>
<sectionName>
<remove>

Syntax

```
<add remove="key" />
```

Attribute

	DESCRIPTION
key	Required attribute. Specifies the name of the setting to remove.

Parent element

ELEMENT	DESCRIPTION
<sectionName> Element	Defines settings for custom configuration sections that use the NameValueSectionHandler and DictionarySectionHandler classes.

Child elements

None

Remarks

You can use the **<remove>** element to remove settings from your application that were defined at a higher level in the configuration file hierarchy.

Example

The following example shows how to use the **<remove>** element in an application configuration file to remove settings previously defined in the machine configuration file.

The following machine configuration file code declares the section **<mySection>** and adds two settings, `key1` and `key2`, to it:

```
<!-- Machine.config file -->
<configuration>
  <configSections>
    <section name="mySection" type="System.Configuration.NameValueSectionHandler,System" />
  </configSections>
  <mySection>
    <add key="key1" value="value1" />
    <add key="key2" value="value2" />
  </mySection>
</configuration>
```

The following application configuration file code removes the `key2` setting from **<mySection>**:

```
<!--Application configuration file -->
<configuration>
  <mySection>
    <remove key="key2" />
  </mySection>
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

<section> element

11/14/2019 • 2 minutes to read • Edit Online

Contains a configuration section declaration.

<configuration>
<configSections>
<section>

<configuration>
<configSections>
<sectionGroup>
<section>

Syntax

```
<section name="section name"
         type="configuration section handler class, assembly"
         allowDefinition="Everywhere|MachineOnly|MachineToApplication"
         allowLocation="true|false" />
```

Required attributes

	DESCRIPTION
name	Specifies the name of the configuration section.
type	Specifies the name of the configuration section handler class that reads the section from the configuration file. The type value has the syntax "fully-qualified-section-handler-class-name, simple-assembly-name". The simple assembly name is the root filename without the *.dll* file extension.

Optional attributes

The following attributes are applicable only for ASP.NET applications. The configuration system ignores these attributes for other application types.

	DESCRIPTION

	DESCRIPTION
allowDefinition	Specifies which configuration file the section can be used in. Use one of the following values: **Everywhere** Allows the section to be used in any configuration file. This is the default. **MachineOnly** Allows the section to be used only in the machine configuration file (*Machine.config*). **MachineToApplication** Allows the section to be used in the machine configuration file or the application configuration file.
allowLocation	Determines whether the section can be used within the **<location>** element. Use one of the following values: **true** Allows the section to be used within the **<location>** element. This is the default. **false** Does not allow the section to be used within the **<location>** element.

Parent elements

	DESCRIPTION
<configSections> Element	Contains configuration section and namespace declarations.
<sectionGroup> Element	Defines a namespace for configuration sections.

> **NOTE**
> A **<section>** element is a child element of either **<configSections>** or **<sectionGroup>** but not both.

Child elements

None

Remarks

Declaring a configuration section essentially defines a new element for the configuration file. The new element contains settings that a configuration section handler (that is, a class that implements the IConfigurationSectionHandler interface) reads. The attributes and child elements of a section you define depend on the section handler you use to read your settings.

Declaring a configuration section handler in the *Machine.config* file enables you to use the configuration section in any application configuration file on that computer, unless the **allowDefinition** attribute specifies otherwise.

Example

The following example shows how to define a configuration section and define settings for that section:

```
<configuration>
  <configSections>
    <section name="sampleSection"
             type="System.Configuration.SingleTagSectionHandler"
             allowLocation="false" />
  </configSections>
  <sampleSection setting1="Value1"
                 setting2="value two"
                 setting3="third value" />
</configuration>
```

Configuration file

This element can be used in the application configuration file, machine configuration file (*Machine.config*), and *Web.config* files that are not at the application directory level.

See also

- Configuration file schema for the .NET Framework

App Settings schema

11/14/2019 • 2 minutes to read • Edit Online

Contains custom application settings, such as file paths, XML Web service URLs, or any other custom configuration information for an application.

```
<configuration>
 <appSettings>
  <add>
  <clear>
  <remove>
```

ELEMENT	DESCRIPTION
<appSettings>	Contains **<add>**, **<clear>**, and **<remove>** tags to control application settings. Has an optional **file** attribute.
<add>	Defines a setting. Child of **<appSettings>**. Requires **key** and **value** attributes.
<clear>	Clears all settings. Child of **<appSettings>**. Has no attributes.
<remove>	Removes a setting. Child of **<appSettings>**. Requires a **key** attribute.

<appSettings> element

This element contains **<add>**, **<clear>**, and **<remove>** tags to control application settings. It defines an optional attribute for **file**.

<add> element

Adds a custom application setting as a name/value pair to the application settings collection. It defines attributes for **key** and **value**.

<clear> element

Removes all references to inherited custom application settings and allows only the references that are added by **<add>** elements following the **<clear>** element. It defines no attributes.

<remove> element

Removes a reference to an inherited custom application setting from the application settings collection. It defines an attribute for **key**.

Example

The following example shows an external application settings file (*custom.config*) that defines a custom application setting:

```
<?xml version="1.0" encoding="utf-8" ?>
<appSettings>
  <add key="MyCustomSetting" value="MyCustomSettingValue" />
</appSettings>
```

The following example shows an application configuration file that consumes the setting in the external settings file and sets an application setting of its own:

```
<configuration>
  <appSettings file="custom.config">
    <add key="ApplicationName" value="MyApplication" />
  </appSettings>
</configuration>
```

See also

- Application Settings Overview
- Application Settings Architecture

Application Settings schema

8/22/2019 • 2 minutes to read • Edit Online

Application settings allow a Windows Forms or ASP.NET application to store and retrieve application-scoped and user-scoped settings. In this context, a *setting* is any piece of information that may be specific to the application or specific to the current user — anything from a database connection string to the user's preferred default window size.

By default, application settings in a Windows Forms application uses the LocalFileSettingsProvider class, which uses the .NET configuration system to store settings in an XML configuration file. For more information about the files used by application settings, see Application Settings Architecture.

Application settings defines the following elements as part of the configuration files it uses.

ELEMENT	DESCRIPTION
<applicationSettings>	Contains all **<setting>** tags specific to the application.
<userSettings>	Contains all **<setting>** tags specific to the current user.
<setting>	Defines a setting. Child of either **<applicationSettings>** or **<userSettings>**.
<value>	Defines a setting's value. Child of **<setting>**.

<applicationSettings> element

This element contains all **<setting>** tags that are specific to an instance of the application on a client computer. It defines no attributes.

<userSettings> element

This element contains all **<setting>** tags that are specific to the user who is currently using the application. It defines no attributes.

<setting> element

This element defines a setting. It has the following attributes.

ATTRIBUTE	DESCRIPTION
name	Required. The unique ID of the setting. Settings created through Visual Studio are saved with the name `ProjectName.Properties.Settings`.

ATTRIBUTE	DESCRIPTION
serializedAs	Required. The format to use for serializing the value to text. Valid values are: - `string`. The value is serialized as a string using a TypeConverter. - `xml`. The value is serialized using XML serialization. - `binary`. The value is serialized as text-encoded binary using binary serialization. - `custom`. The settings provider has inherent knowledge of this setting and serializes and de-serializes it.

<value> element

This element contains the value of a setting.

Example

The following example shows an application settings file that defines two application-scoped settings and two user-scoped settings:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
      <section name="WindowsApplication1.Properties.Settings" type="System.Configuration.ClientSettingsSection,
System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
    </sectionGroup>
    <sectionGroup name="userSettings" type="System.Configuration.UserSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">
      <section name="WindowsApplication1.Properties.Settings" type="System.Configuration.ClientSettingsSection,
System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" />
    </sectionGroup>
  </configSections>
  <applicationSettings>
    <WindowsApplication1.Properties.Settings>
      <setting name="Cursor" serializeAs="String">
        <value>Default</value>
      </setting>
      <setting name="DoubleBuffering" serializeAs="String">
        <value>False</value>
      </setting>
    </WindowsApplication1.Properties.Settings>
  </applicationSettings>
  <userSettings>
    <WindowsApplication1.Properties.Settings>
      <setting name="FormTitle" serializeAs="String">
        <value>Form1</value>
      </setting>
      <setting name="FormSize" serializeAs="String">
        <value>595, 536</value>
      </setting>
    </WindowsApplication1.Properties.Settings>
  </userSettings>
</configuration>
```

See also

- Application Settings Overview
- Application Settings Architecture

Compiler and Language Provider Settings Schema

11/14/2019 • 2 minutes to read • Edit Online

Compiler and language provider settings specify compiler configuration elements for available language providers. Each compiler configuration element specifies the code provider type name, compiler parameters, supported language names, and supported file extensions.

The .NET Framework defines the initial compiler settings in the machine configuration file (Machine.config). Developers and compiler vendors can add configuration settings for a new CodeDomProvider implementation. Use the CodeDomProvider.GetAllCompilerInfo method to programmatically enumerate language provider and compiler configuration settings on a computer.

<configuration>
<system.codedom>
<compilers>
<compiler>

ELEMENT	DESCRIPTION
<system.codedom>	Specifies compiler configuration settings for available language providers.
<compilers>	Container for compiler configuration elements; contains zero or more <compiler> elements.
<compiler>	Specifies the compiler configuration attributes for a language provider.

Example

The following example illustrates a typical compiler configuration element.

```
<configuration>
   <system.codedom>
     <compilers>
       <!-- zero or more compiler elements -->
       <compiler
          language="c#;cs;csharp"
          extension=".cs"
          type="Microsoft.CSharp.CSharpCodeProvider, System, Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
          compilerOptions=""
          warningLevel="1" />
     </compilers>
   </system.codedom>
</configuration>
```

See also

- CompilerInfo
- CodeDomProvider
- Configuration File Schema

- <compiler> Element

Configuration sections schema

10/30/2019 • 2 minutes to read • Edit Online

The configuration sections schema contains elements that define custom settings in configuration files. For general information on configuration files and schemas, see Configuration file schema for the .NET Framework.

<configuration>
<configSections>
<clear>
<remove>
<section>
<sectionGroup>

	DESCRIPTION
<clear> for **<configSections>**	Clears all previously defined sections and section groups.
<clear>	Clears all previously defined sections and section groups.
<configSections>	Contains configuration section and namespace declarations.
<remove> for **<configSections>**	Removes a predefined section or section group.
<section> for **<configSections>** and **<sectionGroup>**	Contains a configuration section declaration.
<sectionGroup> for **<configSections>**	Defines a namespace for configuration sections.

Cryptography Settings Schema

11/14/2019 • 2 minutes to read • Edit Online

The cryptography settings schema contains elements that specify how to map friendly algorithm names to classes that implement cryptography algorithms.

```
<configuration>
 <mscorlib>
  <cryptographySettings>
   <cryptoNameMapping>
    <cryptoClasses>
     <cryptoClass>
    <nameEntry>
   <oidMap>
    <oidEntry>
```

ELEMENT	DESCRIPTION
<cryptoClasses>	Contains a list of cryptography classes that have a mapping to a friendly name in the **<nameEntry>** element.
<cryptoClass>	Contains a cryptography class that has a mapping to a friendly name in the **<nameEntry>** element.
<cryptographySettings>	Contains cryptography settings.
<cryptoNameMapping>	Contains mappings of classes to friendly names.
<mscorlib> element for cryptography settings	Contains the **<cryptographySettings>** element.
<nameEntry>	Maps a class name to a friendly algorithm name, which allows one class to have many friendly names.
<oidEntry>	Maps an ASN.1 object identifier (OID) to a friendly name.
<oidMap>	Contains ASN.1 OID mappings to classes.

See also

- Configuration File Schema
- Cryptographic Services

Network Settings Schema

10/1/2019 • 2 minutes to read • Edit Online

Network settings specify how the .NET Framework connects to the Internet.

The <system.net> settings specify how the .NET Framework connects to the network. The following table describes the function of each child configuration element under the <system.Net> Element (Network Settings).

ELEMENT	DESCRIPTION
<authenticationModules> Element (Network Settings)	Specifies the modules used to authenticate Internet requests.
<connectionManagement> Element (Network Settings)	Specifies the maximum number of connections to Internet hosts.
<defaultProxy> Element (Network Settings)	Specifies the proxy server used for HTTP requests to the Internet.
<mailSettings> Element (Network Settings)	Contains settings for mail sending options.
<requestCaching> Element (Network Settings)	Controls the caching mechanism for network requests.
<webRequestModules> Element (Network Settings)	Specifies the modules used to request information from Internet hosts.

The <uri> settings specify how the .NET Framework handles web addresses expressed using uniform resource identifiers (URIs). The following table describes the function of each child configuration element under the <uri> Element (Uri Settings).

ELEMENT	DESCRIPTION
<idn> Element (Uri Settings)	Specifies if Internationalized Domain Name (IDN) parsing is applied to domain names.
<iriParsing> Element (Uri Settings)	Specifies if International Resource Identifier (IRI) parsing is applied to a Uri and whether IRI parsing rules should be applied.
<schemeSettings> Element (Uri Settings)	Specifies how a Uri will be parsed for specific schemes.

See also

- Configuring Internet Applications
- Configuration File Schema

Run-time settings schema

11/22/2019 • 4 minutes to read • Edit Online

Run-time settings are used by the common language runtime to configure applications that target the .NET Framework.

The <runtime> section and its parent and child elements

```
<configuration>
 <runtime>
  <alwaysFlowImpersonationPolicy>
  <AppContextSwitchOverrides>
  <appDomainManagerAssembly>
  <appDomainManagerType>
  <appDomainResourceMonitoring>
  <assemblyBinding>
   <dependentAssembly>
    <assemblyIdentity>
    <bindingRedirect>
    <codeBase>
    <publisherPolicy>
   <probing>
   <qualifyAssembly>
   <supportPortability>
  <bypassTrustedAppStrongNames>
  <CompatSortNLSVersion>
  <developmentMode>
  <disableCachingBindingFailures>
  <disableCommitThreadStack>
  <disableFusionUpdatesFromADManager>
  <EnableAmPmParseAdjustment>
  <enforceFIPSPolicy>
  <etwEnable>
  <forcePerformanceCounterUniqueSharedMemoryReads>
  <gcAllowVeryLargeObjects>
  <gcConcurrent>
  <GCCpuGroup>
  <GCHeapAffinitizeMask>
  <GCHeapCount>
  <GCLOHThreshold>
  <GCNoAffinitize>
  <gcServer>
  <generatePublisherEvidence>
  <legacyCorruptedStateExceptionsPolicy>
  <legacyImpersonationPolicy>
  <loadfromRemoteSources>
  <NetFx40_LegacySecurityPolicy>
  <NetFx40_PInvokeStackResilience>
  <NetFx45_CultureAwareComparerGetHashCode_LongStrings>
```

<PreferComInsteadOfManagedRemoting>
<relativeBindForResources>
<shadowCopyVerifyByTimeStamp>
<Thread_UseAllCpuGroups>
<ThrowUnobservedTaskExceptions>
<TimeSpan_LegacyFormatMode>
<useLegacyJit>
<UseRandomizedStringHashAlgorithm>
<UseSmallInternalThreadStacks>
<system.runtime.caching>
<memoryCache>
<namedCaches>
<add>
<clear>
<remove>

Alphabetical list of <runtime> elements

ELEMENT	DESCRIPTION
<add>	Adds a named cache to the `namedCaches` collection for a memory cache.
<alwaysFlowImpersonationPolicy>	Specifies that the Windows identity always flows across asynchronous points, regardless of how impersonation was performed.
<AppContextSwitchOverrides>	Defines one or more switches used by the AppContext class to provide an opt-out mechanism for new functionality.
<appDomainManagerAssembly>	Specifies the assembly that provides the application domain manager for the default application domain in the process.
<appDomainManagerType>	Specifies the type that serves as the application domain manager for the default application domain.
<appDomainResourceMonitoring>	Instructs the runtime to collect statistics on all application domains in the process for the life of the process.
<assemblyBinding>	Contains information about assembly version redirection and the locations of assemblies.
<assemblyIdentity>	Contains identifying information about an assembly.
<bindingRedirect>	Redirects one assembly version to another.
<bypassTrustedAppStrongNames>	Specifies whether strong name verification for trusted assemblies should be bypassed.
<clear>	Clears the `namedCaches` collection for a memory cache.
<codeBase>	Specifies where the runtime can find an assembly.

ELEMENT	DESCRIPTION
<CompatSortNLSVersion>	Specifies that the runtime should use legacy sorting behavior when performing string comparisons
<dependentAssembly>	Encapsulates binding policy and assembly location for each assembly.
<developmentMode>	Specifies whether the runtime searches for assemblies in directories specified by the DEVPATH environment variable.
<disableCachingBindingFailures>	Specifies whether the caching of binding failures, which is the default behavior in the .NET Framework 2.0, is disabled.
<disableCommitThreadStack>	Specifies whether the full thread stack is committed when a thread starts.
<disableFusionUpdatesFromADManager>	Specifies whether the default behavior, which is to allow the runtime host to override configuration settings for an application domain, is disabled.
<EnableAmPmParseAdjustment>	Determines whether date and time parsing methods use an adjusted set of rules to parse date strings that contain only a day, month, hour, and AM/PM designator.
<enforceFIPSPolicy>	Specifies whether to enforce a computer configuration requirement that cryptographic algorithms must comply with the Federal Information Processing Standards (FIPS).
<etwEnable>	Specifies whether to enable event tracing for Windows (ETW) for common language runtime events.
<forcePerformanceCounterUniqueSharedMemoryReads>	Specifies whether PerfCounter.dll uses the CategoryOptions registry setting in a .NET Framework version 1.1 application to determine whether to load performance counter data from category-specific shared memory or global memory.
<gcAllowVeryLargeObjects>	On 64-bit platforms, enables arrays that are greater than 2 gigabytes (GB) in total size.
<gcConcurrent>	Specifies whether the runtime runs garbage collection concurrently.
<GCCpuGroup>	Specifies whether garbage collection supports multiple CPU groups.
<GCHeapAffinitizeMask>	Defines the affinity between GC heaps and individual processors.
<GCHeapCount>	Specifies the number of heaps/threads to use for server garbage collection.
<GCLOHThreshold>	Specifies the threshold size that causes objects to go on the large object heap (LOH).

ELEMENT	DESCRIPTION
<GCNoAffinitize>	Specifies whether or not to affinitize server GC threads with CPUs.
<gcServer>	Specifies whether the common language runtime runs server garbage collection.
<generatePublisherEvidence>	Specifies whether the runtime uses code access security (CAS) publisher policy.
<legacyCorruptedStateExceptionsPolicy>	Specifies whether the runtime allows managed code to catch access violations and other corrupted state exceptions.
<legacyImpersonationPolicy>	Specifies that the Windows identity does not flow across asynchronous points, regardless of the flow settings for the execution context on the current thread.
<loadfromRemoteSources>	Specifies whether assemblies from remote sources are loaded as full trust.
<memoryCache>	Defines an element that is used to configure a cache that is based on the MemoryCache class.
<namedCaches>	Contains a collection of configuration settings for the `namedCache` instance.
<NetFx40_LegacySecurityPolicy>	Specifies whether the runtime uses legacy code access security (CAS) policy.
<NetFx40_PInvokeStackResilience>	Specifies whether the runtime automatically fixes incorrect platform invoke declarations at run time, at the cost of slower transitions between managed and unmanaged code.
<NetFx45_CultureAwareComparerGetHashCode_LongStrings>	Specifies whether the runtime uses a fixed amount of memory to calculate hash codes for the StringComparer.GetHashCode method.
<PreferComInsteadOfManagedRemoting>	Specifies that the runtime will use COM interop instead of remoting across application domain boundaries.
<probing>	Specifies subdirectories that the runtime searches when loading assemblies.
<publisherPolicy>	Specifies whether the runtime applies publisher policy.
<qualifyAssembly>	Specifies the full name of the assembly that should be dynamically loaded when a partial name is used.
<relativeBindForResources>	Optimizes the probe for satellite assemblies.
<remove>	Removes a named cache entry from the `namedCaches` collection for a memory cache.

ELEMENT	DESCRIPTION
<runtime>	Contains information about assembly binding and the behavior of garbage collection.
<shadowCopyTimeStampVerification>	Specifies whether shadow copying uses the default startup behavior introduced in the .NET Framework 4, or reverts to the startup behavior of earlier versions of the .NET Framework.
<supportPortability>	Specifies that an application can reference the same assembly in two different implementations of the .NET Framework, by disabling the default behavior that treats the assemblies as equivalent for application portability purposes.
<system.runtime.caching>	Provides configuration information for the default in-memory object cache.
<Thread_UseAllCpuGroups>	Specifies whether the runtime distributes managed threads across all CPU groups.
<ThrowUnobservedTaskExceptions>	Specifies whether unhandled task exceptions should terminate a running process.
<TimeSpan_LegacyFormatMode>	Specifies whether the runtime uses legacy formatting for TimeSpan values.
<useLegacyJit>	Determines whether the common language runtime uses the legacy 64-bit JIT compiler for just-in-time compilation.
<UseRandomizedStringHashAlgorithm>	Specifies whether the runtime calculates hash codes for strings on a per application domain basis.
<UseSmallInternalThreadStacks>	Requests that the runtime use explicit stack sizes when it creates certain threads that it uses internally, instead of the default stack size.

See also

- Configuration File Schema
- To disable concurrent garbage collection
- Redirecting Assembly Versions

Startup settings schema

1/26/2019 • 2 minutes to read • Edit Online

Startup settings specify the version of the common language runtime that should run the application.

ELEMENT	DESCRIPTION
<requiredRuntime>	Specifies that the application supports only version 1.0 of the common language runtime. Applications built with runtime version 1.1 should use the **<supportedRuntime>** element.
<supportedRuntime>	Specifies which versions of the common language runtime the application supports.
<startup>	Contains the **<requiredRuntime>** and **<supportedRuntime>** elements.

See also

- Configuration File Schema
- How to: Configure an app to support .NET Framework 4 or later versions

Trace and Debug Settings Schema

8/22/2019 • 2 minutes to read • Edit Online

Trace and debug settings specify trace listeners that collect, store, and route messages, and the level where a trace switch is set.

The following table describes the function of each trace and debug settings element.

ELEMENT	DESCRIPTION
<add>	Adds a listener to the `Listeners` collection for a trace source.
<add>	Adds a listener to the `Listeners` collection.
<add>	Adds a listener to the `sharedListeners` collection.
<add>	Specifies the level where a trace switch is set.
<assert>	Specifies whether to display a message box when you call the Debug.Assert method; also specifies the name of the file to write messages to.
<clear>	Clears the `Listeners` collection for a trace source.
<clear>	Clears the `Listeners` collection for trace.
<filter>	Adds a filter to a listener in the `Listeners` collection for a trace source.
<filter>	Adds a filter to a listener in the `Listeners` collection for trace.
<filter>	Adds a filter to a listener in the `sharedListeners` collection.
<listeners>	Specifies listeners for the `Listeners` collection for a trace source.
<listeners>	Specifies listeners for the `Listeners` collection for trace.
<performanceCounters>	Specifies the size of the global memory shared by performance counters.
<remove>	Removes a listener from the `Listeners` collection for trace.
<remove>	Removes a listener from the `Listeners` collection for a trace source.
<sharedListeners>	Contains listeners that any source or trace element can reference.

ELEMENT	DESCRIPTION
<sources>	Contains trace sources that initiate tracing messages.
<source>	Specifies a trace source that initiates tracing messages.
<switches>	Contains trace switches and the level where the trace switches are set.
<system.diagnostics>	Specifies trace listeners that collect, store, and route messages and the level where a trace switch is set.
<trace>	Contains listeners that collect, store, and route tracing messages.

See also

- Trace
- TraceSource
- Debug
- Configuration File Schema

Web Settings Schema

11/14/2019 • 2 minutes to read • Edit Online

Web settings specify CPU and execution-level ASP.NET settings that apply to process-wide behavior managed by the ASP.NET hosting layer. These settings differ from application domain-type settings that are specified in the Web.config file of an ASP.NET application.

Web settings are contained in Aspnet.config files, which are located in the installation folders for versions of the .NET Framework. For example, the Aspnet.config file for .NET Framework 2.0 is in the following folder:

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\
```

Web settings are not used in any other configuration files such as the machine.config file, the root Web.config, or application-level Web.config files.

<configuration>
<system.web>
<applicationPool>

ELEMENT	DESCRIPTION
<system.web>	Contains information that the ASP.NET hosting layer uses.
<applicationPool>	Specifies CPU and execution-level ASP.NET settings that apply to process-wide behavior managed by the ASP.NET hosting layer.

See also

- Configuration File Schema

WCF Configuration Schema

8/27/2019 • 2 minutes to read • Edit Online

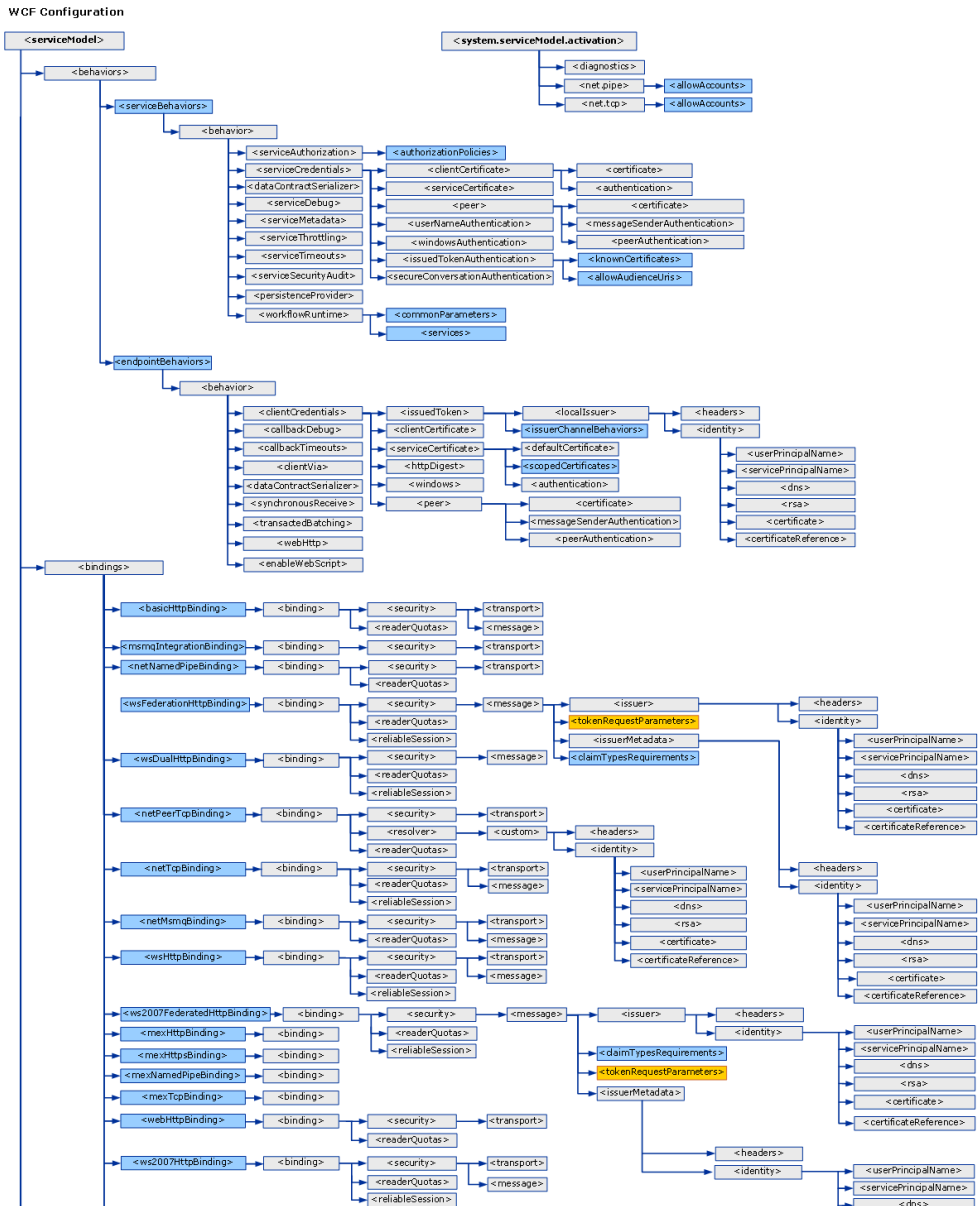
Windows Communication Foundation (WCF) configuration elements enable you to configure WCF service and client applications. You can use the Configuration Editor Tool (SvcConfigEditor.exe) to create and modify configuration files for clients and services. Since the configuration files are formatted as XML, you must be familiar with XML if you want to manually edit them using a text editor. Otherwise, you may run into issues such as an unfound XML element tag or attribute. This is because XML element tags and attributes are case-sensitive.

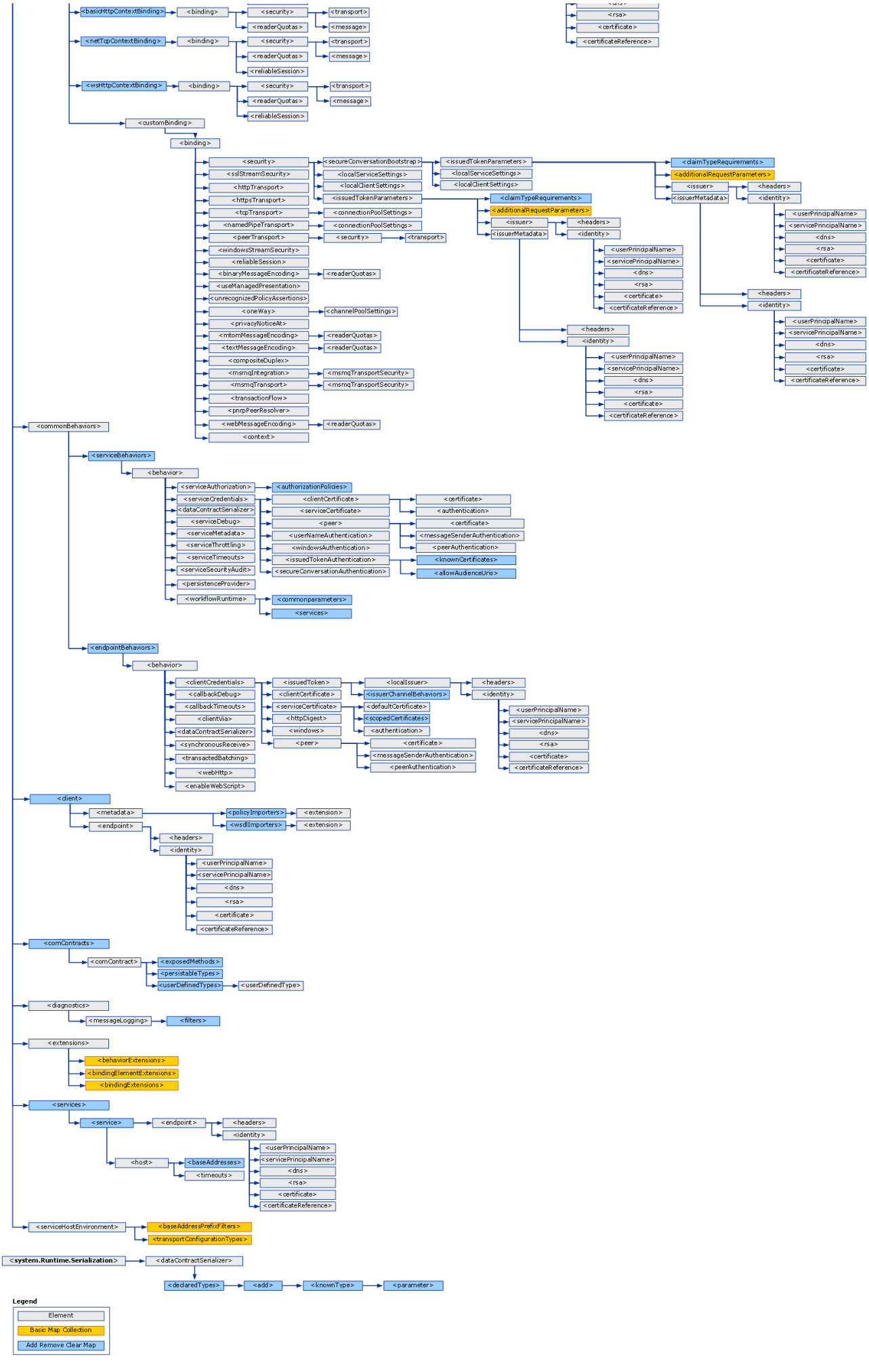
The WCF configuration system is based on the System.Configuration namespace. Therefore, you can use all the standard features provided by the System.Configuration namespace, such as configuration locking, encryption and merging to increase the security of your application and its configuration. For more information on these concepts, see the following topics.

Encrypting Configuration Information

Locking Configuration Settings

This section describes all possible values of each configuration item, and how it interacts with other WCF configuration elements. The following map illustrates the WCF configuration schema:





Caution

You should protect WCF configuration sections in your application configuration files (app.config) with appropriate

Access Control Lists (ACL) to prevent any potential security threats. For example, you should make sure that only the appropriate people can access or modify the security settings on application bindings, or the service model section of the configuration file for a service.

In This Section

<system.serviceModel>
Describes the `ServiceModel` element.

<system.serviceModel.activation>
Configures the SMSvcHost.exe tool.

<system.runtime.serialization>
The top-level element for setting options when using serializers such as the DataContractSerializer.

Related Sections

Configuring Windows Communication Foundation Applications
Describes how to configure WCF services and clients.

WCF Directive Syntax

8/22/2019 • 2 minutes to read • Edit Online

Specifies settings used by the Windows Communication Foundation (WCF) directives in the .svc files to direct the compilers. Each directive can contain one or more attributes (paired with values) that are specific to that directive. WCF has only the @ServiceHost directive.

In This Section

@ServiceHost
Defines page-specific attributes used by the .svc compiler. Can be included only in .svc files.

Related Sections

How to: Host a WCF Service in IIS
Describes how the @ServiceHost directive is used when hosting a service in Internet Information Services (IIS)

How to: Host a WCF Service in WAS
Describes how the @ServiceHost directive is used when hosting a service in Windows Process Activation Service (WAS)

See also

- Hosting
- Hosting in Internet Information Services
- Hosting in Windows Process Activation Service

Windows Identity Foundation Configuration Schema

9/4/2019 • 2 minutes to read • Edit Online

The topics in this section provide information about the Windows Identity Foundation (WIF) configuration schema. You can also configure an application to use WIF through classes exposed by the framework. These classes are noted in the sections that treat relevant elements in the schema. The following shows the basic XML tag structure exposed by the WIF configuration schema. Attributes are omitted. Highlighted comments indicate major components of the schema.

```
<configuration>
    <system.identityModel>
        <!-- Service Configuration -->
        <identityConfiguration>
            <caches>
                <sessionSecurityTokenCache />
                <tokenReplayCache />
            </caches>

            <certificateValidation>
                <certificateValidator />
            </certificateValidation>

            <claimsAuthenticationManager />

            <claimsAuthorizationManager>
                <optionalConfigurationElement>
            </claimsAuthorizationManager>

            <claimTypeRequired>
                <claimType />
            </claimTypeRequired>

            <tokenReplayDetection />

            <!-- Security Token Handler Collection Configuration -->
            <securityTokenHandlers>
                <add>
                    <!-- Can take an optional configuration element which can be one of
                         the following or a custom element -->
                    <samlSecurityTokenHandlerRequirement>
                        <nameClaimType>
                        <roleClaimType>
                    </samlSecurityTokenHandlerRequirement>

                    <sessionSecurityTokenHandlerRequirement />
                    <x509SecurityTokenHandlerRequirement />
                    <userNameSecurityTokenHandlerRequirement />
                </add>
                <clear />
                <remove />
                <securityTokenHandlerConfiguration>
                    <audienceUris>
                        <add>
                        <clear>
                        <remove>
                    </audienceUris>

                    <caches>
                        <sessionSecurityTokenCache />
                        <tokenReplayCache />
                    </caches>
```

```
                    <certificateValidation>
                        <certificateValidator>
                    </certificateValidation>

                    <issuerNameRegistry>
                        <!-- Can take an optional configuration element which can be
                             the <trustedIssuers> element to configure a configuration-based
                             issuer name registry or can be a custom element -->
                        <trustedIssuers>
                            <add>
                            <clear>
                            <remove>
                        </trustedIssuers>
                    </issuerNameRegistry>

                    <issuerTokenResolver />
                    <serviceTokenResolver />
                    <tokenReplayDetection />
                </securityTokenHandlerConfiguration>
            </securityTokenHandlers>
        </identityConfiguration>
    </system.identityModel>

    <system.identityModel.services>
        <!-- Federation Authentication Configuration -->
        <federatedAuthentication>
            <cookieHandler>
                <chunkedCookieHandler />
                <customCookieHandler />
            </cookieHandler>

            <serviceCertificate>
                <certificateReference>
            </serviceCertificate>

            <wsFederation />
        </federatedAuthentication>
    </system.identityModel.services>
</configuration>
```

In This Section

<system.identityModel> Provides configuration for enabling WIF options in applications.

<system.identityModel.services> Provides configuration for passive federation using WIF. Configures the Session Authentication Module (SAM) and the Federated Authentication Module (WSFAM).

Windows Forms Configuration Section

10/30/2019 • 2 minutes to read • Edit Online

Windows Forms configuration settings allow a Windows Forms app to store and retrieve information about customized application settings such as multi-monitor support, high DPI support, and other predefined configuration settings.

Windows Forms application configuration settings are stored in an application configuration file's `System.Windows.Forms.ApplicationConfigurationSection` element.

Syntax

```
<configuration>
  <System.Windows.Forms.ApplicationConfigurationSection>
  ...
  </System.Windows.Forms.ApplicationConfigurationSection>
</configuration>
```

Attributes and elements

The following sections describe attributes, child elements, and parent elements.

Attributes

None.

Child elements

ELEMENT	DESCRIPTION
`<add>`	Adds a configuration setting key with a specified value

Parent elements

ELEMENT	DESCRIPTION
<configuration>	The root element in every configuration file used by the common language runtime and Windows Forms applications

Remarks

Starting with the .NET Framework 4.7, the `<System.Windows.Forms.ApplicationConfigurationSection>` element allows you to configure Windows Forms applications to take advantage of features added in recent releases of the .NET Framework.

The `<System.Windows.Forms.ApplicationConfigurationSection>` element can include one or more child `<add>` elements, each of which defines a specific configuration setting.

See also

- Configuration File Schema

- High DPI Support in Windows Forms

Windows Workflow Foundation Configuration Schema

8/22/2019 • 2 minutes to read • Edit Online

Windows Workflow Foundation (WF) configuration elements enable you to configure workflow applications. For a workflow application, you can configure among other things, tracking and tracing. For more information about tracking and tracing, see Workflow Tracking and Tracing. For workflow services, you can also use Windows Communication Foundation (WCF) configuration elements. For more details about WCF, see WCF Configuration Schema.

Because configuration files are formatted as XML, you must be familiar with XML if you want to manually edit them using a text editor. Otherwise, you may run into issues such as an unfound XML element tag or attribute. This is because XML element tags and attributes are case-sensitive.

In This Section