Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠️ DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.

⚠️ WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.

⚠️ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠️ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.
Preface

General information

Note
The standard applications are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. The standard applications do not represent specific customer solutions, but are only intended to provide support for typical tasks. You are responsible for the proper operation of the described products. These standard applications do not relieve you of your responsibility regarding the safe handling when using, installing operating and maintaining the equipment. By using these standard applications, you agree that Siemens cannot be made liable for possible damage beyond the mentioned liability clause. We reserve the right to make changes and revisions to these standard applications at any time without prior notice. In the case of any differences between the suggestions made in these standard applications and other publications from Siemens, such as catalogs, the contents of the other documentation have priority.

Warranty conditions, liability, and support

If the application has been made available free of charge, the following applies:

We do not provide a warranty for any of the information contained in this document.

All other rights and claims against Siemens AG irrespective of legal basis are excluded. In particular claims for damages against Siemens AG in the case of product outage, downtime, loss of profit, either directly, indirectly or consequential damage are excluded.

This does not apply when liability is compulsory by law, e. g. in the case of the Product Liability Act, premeditation, an act of gross negligence by superiors and managerial staff of Siemens AG or in cases of fraudulent concealment of defects.

This limitation of liability also applies to sub-contractors, suppliers, delegates, superiors and managerial staff of Siemens AG.

German law shall apply to this agreement for customers with head offices in Germany; Swiss law for customers with head offices outside Germany. Application of the United Nations Convention on Contracts for the International Sale of Goods as of 11.04.1980 (CISG) is excluded.
If the application has been made available against payment, the appropriate alternative applies for the respective business transaction:

- **Alternative 1: (Internal business)**
  If nothing else has been negotiated, then the "Conditions for the supply and services in Siemens internal business" applies in the version that is valid at the time that the equipment is purchased.

- **Alternative 2: (Domestic business of Siemens AG)**
  If nothing else was negotiated, the "General License Conditions for Software for Automation and Drives for Customers with a Registered Office in Germany" valid at the time of sale are applicable.

- **Alternative 3: (Direct export business of Siemens AG)**
  If nothing else has been negotiated, then the "General License Conditions for Software Products for Automation and Drives for Customers with a Seat or Registered Office outside Germany", valid at the time of sale, are applicable.

It is not permitted to distribute or duplicate these application examples in any form including excerpts thereof without the express consent of Siemens Industry Sector.

**Notice regarding export identification codes**

AL: N
ECCN: N

**About this document**

**Objective**

This manual describes the commands of the XML interface available with the ProjectGenerator.

The ProjectGenerator is an extension to the SIMATIC system.

The manual is an addition to the SIMATIC standard documentation.

**Note**

This document does not claim to contain all details on devices in any version or to take all conceivable operational cases and applications into account.

Should you require further information or encounter specific problems not covered in enough detail for your field of application, please contact your local Siemens office.

**Target group**

This document is intended for programmers.
Table of contents

Preface ............................................................................................................................................. 3

1 Command overview ..................................................................................................................... 7

1.1 Templates and examples ........................................................................................................ 7
1.2 Overview of parameters ......................................................................................................... 10
1.3 General commands ................................................................................................................ 14
1.3.1 ChangeForm .................................................................................................................... 14
1.3.1.1 Button .................................................................................................................... 15
1.3.1.2 CheckBox ................................................................................................................ 16
1.3.1.3 ComboBox ............................................................................................................... 17
1.3.1.4 Label ....................................................................................................................... 19
1.3.1.5 ListBox .................................................................................................................... 20
1.3.1.6 ListView .................................................................................................................. 22
1.3.1.7 Picture ..................................................................................................................... 24
1.3.1.8 RadioButton ............................................................................................................ 25
1.3.1.9 TextBox ................................................................................................................... 26
1.3.2 ActivateLogging .............................................................................................................. 27
1.3.3 BrowseProject ................................................................................................................. 28
1.3.4 CheckPath ....................................................................................................................... 28
1.3.5 CheckProjectPath ............................................................................................................. 29
1.3.6 DestroyForm .................................................................................................................... 29
1.3.7 FolderBrowser ............................................................................................................... 30
1.3.8 NextCommand ................................................................................................................. 30
1.3.9 OpenFile ......................................................................................................................... 31
1.3.10 PingToIPAddress ............................................................................................................ 32
1.3.11 ReadNextEquipmentModuleConfig .............................................................................. 32
1.3.12 SetColor ....................................................................................................................... 33
1.3.13 Step7DefaultPath ......................................................................................................... 34
1.3.14 StartupPath ................................................................................................................... 34
1.3.15 Step7DefaultPath ......................................................................................................... 34
1.3.16 SetTemporaryVariable ................................................................................................. 35
1.4 Commands for inserting and deleting objects ........................................................................ 36
1.4.1 CreateObject .................................................................................................................. 36
1.4.2 DeleteObject ................................................................................................................... 37
1.4.3 ImportSimaticLibrary .................................................................................................... 38
1.4.4 ImportSimaticSource ..................................................................................................... 39
1.5 Information commands ......................................................................................................... 40
1.5.1 GetAvailableSimaticDevices ......................................................................................... 40
1.5.2 GetAvailableSimaticDeviceVersions ............................................................................. 41
1.5.3 GetDevicesInProject ...................................................................................................... 42
1.5.4 GetSimaticDeviceName ................................................................................................. 43
1.5.5 GetSimaticBlocks .......................................................................................................... 44
1.5.6 GetValueOfSimaticDBVariable .................................................................................... 45
1.5.7 IsSimaticS7300Station .................................................................................................... 45

SIMATIC Project Generator
Parameter Manual, 03/2016
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5.8</td>
<td>GetValueOfTemporaryVariable</td>
<td>46</td>
</tr>
<tr>
<td>1.5.9</td>
<td>GetDeviceProperty</td>
<td>48</td>
</tr>
<tr>
<td>1.6</td>
<td>Commands for the source and object manipulation</td>
<td>49</td>
</tr>
<tr>
<td>1.6.1</td>
<td>Commands for the source manipulation</td>
<td>49</td>
</tr>
<tr>
<td>1.6.1.1</td>
<td>SetLabel</td>
<td>49</td>
</tr>
<tr>
<td>1.6.1.2</td>
<td>SetSimaticValue</td>
<td>50</td>
</tr>
<tr>
<td>1.6.1.3</td>
<td>SetSymbolInSymbolTable</td>
<td>51</td>
</tr>
<tr>
<td>1.6.2</td>
<td>Commands for the object manipulation</td>
<td>52</td>
</tr>
<tr>
<td>1.6.2.1</td>
<td>InsertNetworksIntoOB</td>
<td>52</td>
</tr>
<tr>
<td>1.6.2.2</td>
<td>InsertVariablesIntoOB</td>
<td>54</td>
</tr>
<tr>
<td>1.6.2.3</td>
<td>SetDeviceProperty</td>
<td>55</td>
</tr>
<tr>
<td>1.6.2.4</td>
<td>ConfigureProfinetIRT</td>
<td>56</td>
</tr>
<tr>
<td>1.6.2.5</td>
<td>AddPNDevice</td>
<td>57</td>
</tr>
<tr>
<td>1.6.2.6</td>
<td>AddPNDeviceModule</td>
<td>59</td>
</tr>
<tr>
<td>1.6.2.7</td>
<td>CreateET200Module</td>
<td>61</td>
</tr>
<tr>
<td>1.6.2.8</td>
<td>CreatePNTopology</td>
<td>63</td>
</tr>
<tr>
<td>1.6.2.9</td>
<td>CreateProfinetSubsystem</td>
<td>64</td>
</tr>
<tr>
<td>A</td>
<td>Contact</td>
<td>65</td>
</tr>
<tr>
<td>A.1</td>
<td>Contacts</td>
<td>65</td>
</tr>
<tr>
<td>A.2</td>
<td>Internet addresses</td>
<td>66</td>
</tr>
</tbody>
</table>
1.1 Templates and examples

Templates for XML files

Detailed and executable templates for XML files are supplied with the project generator and can be used as a copy template or as an orientation aid for your own expansions.

Example

The following extract from the Communication_LCom.XML standard module provides an overview of the interaction of the individual commands in an XML description file.

Table 1-1 Extract from the XML description file of the LCom standard module

```xml
<CommandList Name="FBLComMachineCom"
DisplayText="Use Ethernet Communication LCom"
ModulInfoFile="SIMATIC\EquipmentModules\Communication_LCom\LCom_Ethernet_Communication_Library_for_SIMATIC_V1_1.pdf">
  <Command ID="1" Name="ChangeForm">
    <Control Action="add"
      Type="Button"
      Name="BT.Exit"
      Text="Exit"
      Location="12, 531"
      Size="130, 30"
      Enabled="true"
      Visible="true"
      ToolTip="Abort this program">
      <Events>
        <Click code="MyApp.NextCommand(0)"/>
      </Events>
    </Control>
  </Command>
</CommandList>
```
Command overview
1.1 Templates and examples

Table 1-2 Continuation of extract from the XML description file of the LCom standard module

<Control Action="add"
  Type="Button"
  Name="BT_Help"
  Text="Help"
  Location="12, 491"
  Size="130, 30"
  Enabled="true"
  Visible="true"
  ToolTip="Show help information">
  <Events>
    <Click code="MyApp.NextCommand(9)"/>
  </Events>
</Control>
<Control Action="add"
  Type="Label"
  Name="LBL_Head_Info"
  Text="Ethernet Communication LCom - Function Block Call"
  Location="173, 110"
  AutoSize="true"
  BackColor="__call_SetColor(Transparent)">
</Control>
<Control Action="add"
  Type="Label"
  Name="LBL_Info"
  Text="Fill in the required data and this tool will add a function block call in OB1 to the project."
  Location="160, 160"
  Size="600, 30"
  AutoSize="false"
  BackColor="__call_SetColor(White)">
</Control>
<Control Action="add"
  Type="Label"
  Name="LBL_ProjectName"
  Text="Project name"
  Location="175, 200"
  AutoSize="true"
  BackColor="__call_SetColor(White)">
</Control>...

If the example is completed further (as shown in the executable supplied file Communication_LCom.XML), the project generator screenshot for the example appears as follows:
Ethernet Communication LCom - Function Block Call

Fill in the required data and this tool will add a function block call in OB1 to the project.

Add FB call to the project:
- New instance DB name: DBFBLCOMMachineCom
- Parameter DB name: DBLCOMParameter
- Send DB name: DBLCOMSend
- Receive DB name: DBLCOMReceive

Communication parameters:
- SIMATIC is TCP client (active partner)
- Local port: 3456
- Remote port: 3456
- Connection ID: 105
- Sender cycle time (ms): 500

Data blocks in device:
- DBFBLCOMMachineCom: DB 105, FB 105
- DBLCOMParameter: DB 500, DB 500
- DBLCOMReceive: DB 502, DB 502
- DBLCOMSend: DB 501, DB 501

Help
Exit
## 1.2 Overview of parameters

The table below provides an overview of the most important parameters that can be set for the commands in the Project Generator. The parameters are listed together with their meaning, the data type, their possible values, and an example.

### Table 1-3 Overview of the commonly used parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Possible values</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
<td>STRING</td>
<td>Add, Remove</td>
<td>Action=&quot;add&quot;</td>
</tr>
<tr>
<td>Alignment</td>
<td>Specifies the alignment of the text within the control.</td>
<td>ENUM</td>
<td>Default, Left, Top, SnapToGrid</td>
<td>Alignment=&quot;HorizontalAlignment.Left&quot;</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automaticaly.</td>
<td>BOOL</td>
<td>True, False (default)</td>
<td>AutoSize=&quot;true&quot;</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color. All colors that are defined in the .Net object System.Drawing.Color are possible here (e.g. White, Green, Blue, Red, LightSlateGray, etc.). A transfer of the RGB color codes is not possible.</td>
<td>Is defined via the SetColor command.</td>
<td>BackColor=&quot;__call_setcolor(TPntransparent)&quot;</td>
<td></td>
</tr>
<tr>
<td>BlockType</td>
<td>The data block and instance data block types, as well as function and function block, are allowed.</td>
<td>ENUM</td>
<td>DATA_BLOCK/FUNCTION/FUNCTION_BLOCK</td>
<td>BlockType=&quot;DATA_BLOCK&quot;</td>
</tr>
<tr>
<td>Checked</td>
<td>Specifies whether the element is activated or not.</td>
<td>BOOL</td>
<td>True, False (default)</td>
<td>Checked=&quot;true&quot;</td>
</tr>
<tr>
<td>DropDownStyle</td>
<td>Specifies the display type of the control.</td>
<td>ENUM</td>
<td>Simple, DropDownList</td>
<td>DropDownListStyle = &quot;DropDown-List&quot;</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
<td>BOOL</td>
<td>True (default), False</td>
<td>Enabled=&quot;true&quot;</td>
</tr>
<tr>
<td>FullRowSelect</td>
<td>Specifies whether the whole row is selected.</td>
<td>BOOL</td>
<td>True, False (default)</td>
<td>FullRowSelect=&quot;true&quot;</td>
</tr>
<tr>
<td>HideSelection</td>
<td>Specifies whether the selection is hidden when the control loses focus.</td>
<td>BOOL</td>
<td>True, False (default)</td>
<td>HideSelection=&quot;false&quot;</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
<td>INTEGER, INTEGER</td>
<td>0.0</td>
<td>Location=&quot;450,200&quot;</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>Specifies whether multiple row selection is permitted</td>
<td>BOOL</td>
<td>True, False (default)</td>
<td>MultiSelect=&quot;false&quot;</td>
</tr>
<tr>
<td>Parameter</td>
<td>Meaning</td>
<td>Data type</td>
<td>Possible values</td>
<td>Code example</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
<td>STRING</td>
<td>---</td>
<td>Name=&quot;LV_Projects&quot;</td>
</tr>
<tr>
<td>Position</td>
<td>Only FIRST and LAST are permitted. If no position is given, the new networks are attached.</td>
<td>ENUM</td>
<td>FIRST/LAST</td>
<td>Position=&quot;first&quot;</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the control (width, height) if AutoSize is not selected.</td>
<td>INTEGER, INTEGER</td>
<td>0.0</td>
<td>Size=&quot;150,13&quot;</td>
</tr>
<tr>
<td>SizeMode</td>
<td>Specifies the display of the picture.</td>
<td>ENUM</td>
<td>Normal (Default) StretchImage AutoSize CenterImage Zoom</td>
<td>SizeMode = &quot;StretchImage&quot;</td>
</tr>
<tr>
<td>Source</td>
<td>Transfers the content of the control element if this is generated via a system function.</td>
<td>STRING for function call</td>
<td>---</td>
<td>Source=&quot;__call_GetAvailableDeviceVersions&quot;</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
<td>STRING</td>
<td>---</td>
<td>Text=&quot;Button1&quot;</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
<td>STRING</td>
<td>---</td>
<td>ToolTip=&quot;Select the version of the new device&quot;</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
<td>STRING</td>
<td>Button Label TextBox ListBox ListView RadioButton CheckBox ComboBox Picture</td>
<td>Type=&quot;TextBox&quot;</td>
</tr>
<tr>
<td>VersionsCheckType</td>
<td>EXTERNAL: (two-digit version check): is checked in the version in the block properties. INTERNAL (three-digit version check): the first version ID is searched for in the first comment field of the block and used for the version check.</td>
<td>ENUM</td>
<td>INTERNAL/EXTERNAL</td>
<td>VersionCheckType=&quot;INTERNAL&quot;</td>
</tr>
</tbody>
</table>
### 1.2 Overview of parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Possible values</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>Specifies the view of the control.</td>
<td>ENUM</td>
<td>LargeIcon (default) Details SmallIcon List Title</td>
<td>View=&quot;Details&quot;</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
<td>BOOL</td>
<td>True (default) False</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1-4** Overview of the commonly used parameters for items

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Possible values</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Accesses the source input via a reference (@source. …): Select which elements from the system function call are to be selected. All elements from the Source parameter are then accepted.</td>
<td>STRING</td>
<td>---</td>
<td>items=&quot;@source.name&quot;</td>
</tr>
<tr>
<td>Name</td>
<td>Elements can be added directly here if no elements have been specified via the Item tag.</td>
<td>STRING</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>Specifies the preselection of the current index of the combo box.</td>
<td>INTEGER</td>
<td>0</td>
<td>SelectedIndex =&quot;0&quot;</td>
</tr>
</tbody>
</table>

**Table 1-5** Overview of the commonly used parameters for events

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Possible values</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckedChanged</td>
<td>Is triggered when the state of the element has been changed.</td>
<td>---</td>
<td>----</td>
<td>&lt;Click code=&quot;MyApp.NextCommand(0)&quot; /&gt;</td>
</tr>
<tr>
<td>Click</td>
<td>This action is triggered by clicking the button.</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>SelectedIndexChanged</td>
<td>Is called when a different element has been selected within the control.</td>
<td>---</td>
<td>---</td>
<td>SelectedIndexChanged code=&quot;@<a href="mailto:BT_Next@.Enabled">BT_Next@.Enabled</a>=@<a href="mailto:LV_Projects@.SelectedItems.Count">LV_Projects@.SelectedItems.Count</a> &gt; 0&quot;</td>
</tr>
</tbody>
</table>
All events provided by .NET can also be used here.

Table 1-6  Overview of the commonly used parameters for limits

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Data type</th>
<th>Possible values</th>
<th>Code example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataType</td>
<td>The data type that is to be entered in the text field can be specified here. Because of the default setting, certain characters are hidden and the value checked for validity.</td>
<td>ENUM</td>
<td>OBJECT, IPAddress, BOOL, BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD</td>
<td><code>DataType=&quot;uint&quot;</code></td>
</tr>
<tr>
<td>Length</td>
<td>The length can also be limited here for the Object data type.</td>
<td>INTEGER</td>
<td>255</td>
<td><code>Length=&quot;&quot; Min=&quot;1&quot; Max=&quot;&quot;</code></td>
</tr>
<tr>
<td>Max</td>
<td>A value can be limited to a maximum value here.</td>
<td>INTEGER</td>
<td>Dependent on the data type</td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>A value can be limited to a minimum value here.</td>
<td>INTEGER</td>
<td>Dependent on the data type</td>
<td></td>
</tr>
</tbody>
</table>


The following section contains the descriptions of all controls that can be inserted in the user interface.
1.3  General commands

1.3.1  ChangeForm

The most frequently required properties of the controls (user interface elements) are described in this section. It is possible to access all properties provided by the Visual Basic .NET.

A code, which is generated from the relevant code element of the XML tag during runtime, is transferred to the events of the user interface elements. This means that the code that is required for the execution is only compiled at the respective position and then inserted.

If a reference is required to an active element of the user interface, the element can be accessed via the assigned name. For this purpose however, the @ character must be inserted before and after the name so that the correct reference can be found.

The syntax for the code must comply with Visual Basic .NET. The insert sequence for access to elements within a ChangeForm command is not relevant (i.e. access to a control can be from the first element (see the example below) directly to the last control.

Example

A checkbox is available on the user interface and the status Checked is to be set:

Table 1-7  Code example

```
Code="@<Name des Controls>@.Checked = True"
```
1.3.1.1 Button

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the element (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
</tbody>
</table>

Events

Click | This action is triggered by clicking the button.

Example

Table 1-8 Button code example

```xml
<Control
  Action="add"
  Type="Button"
  Name="BT_Button1"
  Text="Button1"
  Location="12, 531"
  Size="130, 30"
  Enabled="true"
  Visible="true"
  ToolTip="This is Button1">
  <Events>
    <Click code="MyApp.NextCommand(0)" />
  </Events>
</Control>
```
1.3 General commands

1.3.1.2 CheckBox

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Checked</td>
<td>Specifies whether the checkbox is activated or not.</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the control (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckedChanged</td>
<td>Is triggered when the state of the checkbox has been changed.</td>
</tr>
</tbody>
</table>

Example

Table 1-9 CheckBox code example

```
<Control
    Action="add"
    Type="CheckBox"
    Name="CB_CheckBoxName"
    Text="CheckBox1"
    BackColor="__call_SetColor(Transparent)"
    AutoSize="true"
    Location="200,277"
    Checked="true"
    Size="200,17"
    ToolTip="This is CheckBox1">
</Control>
```
1.3.1.3 **ComboBox**

![ComboBox Image](Image 1-1 ComboBox)

**Parameter**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>DropDownListStyle</td>
<td>Specifies the display type of the control.</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the control (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Source</td>
<td>Transfers the content of the control element if this is generated via a system function.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
</tbody>
</table>

**Items**

<table>
<thead>
<tr>
<th>Name</th>
<th>Elements can be added directly here if no elements have been specified via the Item tag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Specifies the alignment of the text within the control.</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>Specifies the preselection of the current index of the combo box.</td>
</tr>
<tr>
<td>Items</td>
<td>Accesses the source input via a reference (@source. ...): Select which elements from the system function call are to be selected. All elements from the Source parameter are then accepted.</td>
</tr>
</tbody>
</table>

**Events**

| SelectedIndexChanged | Is called when a different element has been selected within the control. |
1.3 General commands

Example

Table 1-10 ComboBox code example

```xml
<Control
    Action="add"
    Type="ComboBox"
    Name="CB_Machine_Type"
    Source="__call_GetAvailableMachineTypes('OMAC V3',
            'SIMATIC\EquipmentModules\InterfaceGenerator\InterfaceMachineTypes')"
    DropDownStyle = "DropDownList"
    Location="360, 360"
    Size="180,13"
    ToolTip="Select the machine type">
    <Items
        Name=""
        Alignment="HorizontalAlignment.Left"
        Items="@source.name">
        </Items>
    </Control>
```
1.3.1.4 Label

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the element (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
</tbody>
</table>

Example

Table 1-11 Label code example

```xml
<Control
    Action="add"
    Type="Label"
    Name="LBL_Label1"
    Text="Label1"
    BackColor="__call_SetColor(Transparent)"
    AutoSize="true"
    Location="173, 110" >
</Control>
```
1.3.1.5 ListBox

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the button (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Source</td>
<td>Transfers the content of the control element if this is generated via a system function.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
</tbody>
</table>

Events

- SelectedIndexChanged: Is called when a different element has been selected within the control.

Items

- items: Accesses the source input via a reference (@source. ...):
  Select which elements from the system function call are to be selected. All elements from the Source parameter are then accepted.
Example

Table 1-12  ListBox code example

```
<Control Action="add"
  Type="ListBox"
  Name="LB_Blocks_in_device"
  Source="__call_GetSimaticBlocks"
  Location="175, 350"
  Size="110, 100"
  ToolTip="Instance DBs in the selected device">
  <Items Name="Device"
    Size="100"
    Alignment="HorizontalAlignment.Left"
    items="@source.Name">
  </Items>
</Control>
```
1.3 General commands

1.3.1.6 ListView

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>View</td>
<td>Specifies the view of the control.</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>Specifies whether multiple row selection is permitted.</td>
</tr>
<tr>
<td>FullRowSelect</td>
<td>Specifies whether the whole row is selected.</td>
</tr>
<tr>
<td>HideSelection</td>
<td>Specifies whether the selection is hidden when the control loses focus.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the control (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>Source</td>
<td>Transfers the content of the control element if this is generated via a system function.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
</tbody>
</table>

**Events**

- **SelectedIndexChanged** Is called when a different element has been selected within the control.

**ListViewItem**

- **Name** Specifies the name of the main element.
- **Size** Specifies the size of the display within the control (can be visible or not depending on the selected view of the control).
- **Alignment** Alignment of the text within the control.

**ListViewSubItem**

- **Name** Specifies the name and the text that is entered as header in the table.
Command overview

1.3 General commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Specifies the size of the display within the column (can be visible or not depending on the selected view of the control).</td>
</tr>
<tr>
<td>Alignment</td>
<td>Specifies the alignment of the text within the column.</td>
</tr>
<tr>
<td>Item</td>
<td>Accesses the source input via a reference (@source, ...): Select which elements from the system function call are to be selected. All elements from the Source parameter are then accepted.</td>
</tr>
</tbody>
</table>

Example

**Table 1-13 ListView code example**

```xml
<Control
   Action="add"
   Type="ListView"
   View="Details"
   Name="LV_Projects"
   MultiSelect="false"
   FullRowSelect="true"
   HideSelection="false"
   Location="170,220"
   Size="600,300"
   Source="__call_GetProjects">
   <ListViewItem
      Name="Project"
      Size="100"
      Alignment="HorizontalAlignment.Left"
      Item="@source.name" />
   <ListViewSubItem
      Name="Projectpath"
      Size="300"
      Alignment="HorizontalAlignment.Left"
      Item="@source.logpath" />
   <ListViewSubItem
      Name="Last modified"
      Size="130"
      Alignment="HorizontalAlignment.Left"
      Item="@source.modified" />
   <ListViewSubItem
      Name="Creator"
      Size="50"
      Alignment="HorizontalAlignment.Left"
      Item="@source.creator" />
</ListViewItem>
<Events>
   <SelectedIndexChanged
code="@BT_Next@.Enabled=@LV_Projects@.SelectedItems.Count > 0"/>
</Events>
</Control>
```
1.3.1.7 Picture

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the picture (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>SizeMode</td>
<td>Specifies the display of the picture.</td>
</tr>
<tr>
<td>ImageLocation</td>
<td>Specifies the path to the picture. The path can also be specified relative.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
</tbody>
</table>

Example

Table 1-14 Picture code example

```
<Control
    Action="add"
    Type="Picture"
    Name="PB_Example2"
    Location="350, 0"
    Size="250,150"
    SizeMode = "StretchImage"
    ImageLocation="Pictures\Example2.png">
</Control>
```
1.3.1.8 RadioButton

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Checked</td>
<td>Specifies whether the radio button is activated or not.</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the button (width, height) if AutoSize is not selected.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Specifies the text of the tooltip for the element.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
</tbody>
</table>

Example

Table 1-15 RadioButton code example

```xml
<Control
    Action="add"
    Type="RadioButton"
    Name="RB_RadioButton1"
    BackColor="__call_SetColor(Transparent)"
    Text="RadioButton1"
    AutoSize="true"
    Enabled="true"
    Location="170,216"
    Checked="true"
    Size="200,17"
    ToolTip="This is RadioButton1">
</Control>
```
## Command overview

### 1.3 General commands

#### 1.3.1.9 TextBox

- **Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Specifies the action that is to be executed.</td>
</tr>
<tr>
<td>Type</td>
<td>Specifies the type of the control.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies a unique name for the control within the active user interface. The name can also be used for referencing within the newly compiled code.</td>
</tr>
<tr>
<td>Text</td>
<td>Specifies the text which is to be displayed on the element.</td>
</tr>
<tr>
<td>Location</td>
<td>Specifies the position on the user interface (x, y).</td>
</tr>
<tr>
<td>Size</td>
<td>Specifies the size of the element (width, height) if <strong>AutoSize</strong> is not selected.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Specifies whether an element can be operated.</td>
</tr>
<tr>
<td>Visible</td>
<td>Specifies whether an element is visible.</td>
</tr>
<tr>
<td>AutoSize</td>
<td>Specifies whether the size of the element is to be determined automatically.</td>
</tr>
<tr>
<td>BackColor</td>
<td>Specifies the background color.</td>
</tr>
</tbody>
</table>

- **Limits**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataType</td>
<td>The data type that is to be entered in the text field can be specified here. Because of the default setting, certain characters are hidden and the value checked for validity.</td>
</tr>
<tr>
<td>Length</td>
<td>The length can also be limited here for the <strong>Object</strong> data type.</td>
</tr>
<tr>
<td>Min</td>
<td>A value can be limited to a minimum value here.</td>
</tr>
<tr>
<td>Max</td>
<td>A value can be limited to a maximum value here.</td>
</tr>
</tbody>
</table>

- **Data types for the Limit tag data type**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Checks the syntax according to SIMATIC Manager object specifications.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>Checks for length, value and STEP 7 syntax in the IPv4 aaa.bbb.ccc.ddd format.</td>
</tr>
<tr>
<td>Bool</td>
<td>Checks for value and STEP 7 syntax.</td>
</tr>
<tr>
<td>Byte</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Word</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>DWord</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Int</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Dint</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Real</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Time</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Date</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
<tr>
<td>Tod</td>
<td>Checks for length, value, and STEP 7 syntax.</td>
</tr>
</tbody>
</table>
Example

Table 1-16  TextBox code example

<Control Action="add"
    Type="TextBox"
    Name="TB_LCom_FB_Number"
    Text="105"
    TextAlign="center"
    Location="615, 248"
    Size="60,13"
    Autosize="false"
    ToolTip="">
    <Limits DataType="udint" Length="" Min="1" Max="5000"/>
    <Events>
        <TextChanged Code="@TB_Connection_ID@.Text = @TB_LCom_FB_Number@.Text"/>
    </Events>
</Control>

1.3.2  ActivateLogging

The logging functionality of the project generator activities is activated with this command.

Command parameters

This command does not have any parameters.

Example

Table 1-17  Code example

<Command
    ID="100"
    Name="ActivateLogging"
    NCID="101">
1.3.3 **BrowseProject**

A file browser is opened with this command in order to select a STEP 7 project. When a valid project is selected, it is opened.

**Command parameters**

This command does not have any parameters.

**Example**

Table 1-18 Code example

```xml
<Click code="If MyApp.BrowseProject Then MyApp.NextCommand(32)"/>
```

1.3.4 **CheckPath**

This command can be called in the code of an event with the *MyApp* prefix.

The current value is read from the transferred control with this command. A check is performed as to whether the path is available.

**Command parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Transfer of the control with the start and target path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

Table 1-19 Code example

```xml
<Leave code="MyApp.CheckPath(@TB_Path@)"/>
```
1.3.5 CheckProjectPath

This command can be called in the code of an event with the MyApp prefix.
This command checks whether the project path already exists.

Command parameters

<table>
<thead>
<tr>
<th>Transfer parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
</tr>
<tr>
<td>Name</td>
</tr>
</tbody>
</table>

Example

Table 1-20  Code example

<Click code="If MyApp.CheckProjectPath(@TB_Path@.Text,@TB_Name@.Text) Then MyApp.NextCommand(26)"/>

1.3.6 DestroyForm

All control elements of the user interface are deleted with this command. Exceptions are the device overview and the display of the current configuration step.

This command is used when the configuration of a standard module stretches over several pages.

Command parameters

This command does not have any parameters.

Example

Table 1-21  Code example

<Command ID="1" Name="DestroyForm" NCID="20"/>
</Command>
1.3 General commands

1.3.7 FolderBrowser
This command can be called in the code of an event with the MyApp. prefix.
The current value from the transferred control is read out with this command and the browser
window with the read value started. After a path has been selected, the command writes the
path back to the control.
The return value of the function is a Boolean value which indicates whether the selection has
been confirmed with OK.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Transfer of the control with the start and target path</th>
</tr>
</thead>
</table>

Example

Table 1-22 Code example

<Click code="MyApp.FolderBrowser(@TB_Path@)"/>

1.3.8 NextCommand
This command is only called in the code of an event with the MyApp. prefix.
A jump is performed to the Element command line with the transferred ID with this
command. If the ID equal to 0 is transferred, the Exit dialog box is opened.

Command parameters

<table>
<thead>
<tr>
<th>Transfer parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCID</td>
</tr>
</tbody>
</table>

Example

Table 1-23 Code example

<Click code="MyApp.NextCommand(0)"/>
1.3.9 OpenFile

A file with the standard program set in Windows is started with this command. If several files are to be opened, the parameter line can be executed several times.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Specifies the path to the file</td>
</tr>
</tbody>
</table>

Example

Table 1-24 Code example

```xml
<Command
  ID="100"
  Name="OpenFile"
  NCID="20">
  <Parameter
    Path="SIMATIC\EquipmentModules\Communication_LCom\LCom_ethernet_communication_library_for_SIMATIC_V1_1.pdf"/>
</Command>
```
1.3 General commands

1.3.10 PingToIPAddress

A ping to an IP address is performed with this command and the result is output in a message box. If several pings are to be performed, the parameter line can be called several times.

Command parameters

Transfer parameters

Value | Target IP address

Example

Table 1-25 Code example

```xml
<Command
    ID="100"
    Name="PingToIPAddress"
    NCID="20">
    <Parameter Name="Parameter1" value="__ref_TB_IPAddress.Text"/>
</Parameter>
</Command>
```

1.3.11 ReadNextEquipmentModuleConfig

This command informs the ProjectGenerator that the configuration of an equipment module has been completed. This command is therefore the last command that is executed for the configuration of an equipment module. If a further command has been selected in the selection for the equipment module which follows the current command, the configuration is continued with the next equipment module. If this is not the case, the "Project Generation" page is opened after the call of this command.

Command parameters

This command does not have any parameters.

Example

Table 1-26 Code example

```xml
<Command
    ID="100"
    Name="ReadNextEquipmentModuleConfig"/>
```
1.3.12 SetColor

This command defines colors, e.g. the color of the background.

Command parameters

Transfer parameters

Color All colors that are defined in the .Net object System.Drawing.Color are possible here (e.g. White, Green, Blue, Red, LightSlateGray, etc.).
A transfer of the RGB color codes is not possible.

Example

Table 1-27 Code example

```
BackColor="__call_SetColor(Transparent)"
```

1.3.13 ShowAboutBox

The About box is activated with this command. The About box contains information on the precise product designation, the installed version, the company name, and the copyright.

Command parameters

This command does not have any parameters.

Example

Table 1-28 Code example

```
<Command
  ID="6"
  Name="ShowAboutBox"/>
```
1.3 General commands

1.3.14 StartupPath

This command returns the path to the main directory of the ProjectGenerator as a string. If the command is called via the VB code, this must be written as follows: MyApp.StartupPath

Example

Table 1-29 Code example

Use in the source code
REM Path of the ProjectGenerator
Dim ProjectGeneratorPath As String = MyApp.StartupPath

1.3.15 Step7DefaultPath

This command can be transferred directly to a control property via the __Call_ prefix. The current default path set for the projects in STEP 7 is returned with this command.

Command parameters

This command does not have any parameters.

Example

Table 1-30 Code example

Text="__call_Step7DefaultPath"
1.3.16 SetTemporaryVariable

With this command, values in the data management of the ProjectGenerator can be stored temporarily.

Any number of values can be stored temporarily with one call of the command. The parameter line must be filled in several times in this case. The values can be read with the GetValueOfTemporaryVariable function.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the temporary variables.</td>
</tr>
<tr>
<td>Type</td>
<td>Optional: You can choose between the options LOCAL and GLOBAL. Local temporary variables are automatically deleted when the next module is called; global variables are retained. If this parameter is not supplied, LOCAL is taken automatically.</td>
</tr>
<tr>
<td>Value</td>
<td>Value of the temporary variables.</td>
</tr>
</tbody>
</table>

Example

Two variables are saved. The first variable is called „Hello“ and has the value „World“. The second variable is called „ModuloLength“ and has the value „360“.

Table 1-31 Code example

Use in a command

```xml
<Command ID="100011" Name="setTemporaryVariable" NCID="">  
  <Parameter Name="Hello" Type="Global" Value="World"/>
  <Parameter Name="ModuloLength" Value="360"/>
</Command>
```

Use in the source code

```java
MyApp.myISL.setTemporaryVariable('ProjectName', 'Global', 'MyProject')
MyApp.myISL.setTemporaryVariable('ModuloLength', 'Local', 360.0)
```


1.4 Commands for inserting and deleting objects

1.4.1 CreateObject

An object that is defined via Type and the ExternalTypeName is created with this command. The name of the respective type is used to check whether this already exists. If it already exists, the command is aborted.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The object type of the device that is to be created is specified here.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the new object</td>
</tr>
<tr>
<td>Version</td>
<td>Version of the new object</td>
</tr>
<tr>
<td>ExternalTypeName</td>
<td>Type of device to be imported</td>
</tr>
<tr>
<td>ReferenceObject</td>
<td>Transfer of a reference object (list box) to directly update the display of</td>
</tr>
<tr>
<td></td>
<td>the user interface.</td>
</tr>
</tbody>
</table>

Example

Table 1-32 Code example

```xml
<Command
   ID="100"
   Name="CreateObject"
   NCID="20">
   <Parameter
      Type="Device"
      Name="__ref_TB_Name_New_Device.text"
      Version="__ref_CB_Name_New_Device_AvailableDeviceVersions.text"
      ExternalTypeName="__ref_CB_Name_New_Device_ExternalTypeName.text"
      ReferenceObject="LB_Devices"/>
</Command>
```
1.4.2 DeleteObject

A newly created object can be deleted with this command. Objects that already exist in a project cannot be deleted.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The <em>object type</em> of the device that is to be deleted is specified here.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Name of the object to be deleted</td>
<td></td>
</tr>
<tr>
<td>ExternalTypeName</td>
<td>Type of the object to be deleted</td>
<td></td>
</tr>
<tr>
<td>ReferenceObject</td>
<td>Transfer of a reference object (list box) to directly update the display of the user interface.</td>
<td></td>
</tr>
</tbody>
</table>

Example

Table 1- 33 Code example

```
<Command
    ID="100"
    Name="DeleteObject"
    NCID="20">
    <Parameter
        Type="Device"
        Name="__ref_TB_Name_New_Device.text"
        ExternalTypeName="__ref_CB_Name_New_Device_ExternalTypeName.text"
        ReferenceObject="LB_Devices"/>
</Command>
```
1.4 Commands for inserting and deleting objects

1.4.3 ImportSimaticLibrary

Libraries are imported to an active device with this command. Any number of libraries can be imported with one call of the command. The parameter line must be filled in several times in this case.

Command parameters

Parameter | Name |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the program folder in the library to be imported.</td>
</tr>
<tr>
<td>VersionsCheckType</td>
<td>In the EXTERNAL version check (two-digit version check), the version is checked using the version ID in the block properties. For INTERNAL (three-digit version check), the first version ID is searched for in the first comment field of the block and used for the version check.</td>
</tr>
<tr>
<td>Path</td>
<td>Path to the library to be imported. The path can also be specified relatively. The library must be available in STEP7 project format.</td>
</tr>
<tr>
<td>SymbolicFBName</td>
<td>Symbolic name of a block that represents a proxy for the entire library. In this block, the version check is also carried out.</td>
</tr>
</tbody>
</table>

Example

Table 1-34 Code example

```xml
<Command
   ID="10"
   Name="ImportSimaticLibrary"
   NCID="20">
   <Parameter
      Name="LComCP343"
      VersionCheckType="INTERNAL"
      Path="SIMATIC\EquipmentModules\Communication_LCom\Data\Libraries\LCom\LCom.s7p"
      SymbolicFBName="FBLComMachineCom"/>
</Command>
```
1.4.4 ImportSimaticSource

Sources can be imported into the active device with this command.

Any number of sources can be imported with one call. The parameter line must be filled in several times for this purpose.

A source may only contain one block.

Command parameters

Parameter
Name: Symbolic name of the block within the project.
The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.

VersionsCheckType: In the EXTERNAL version check (two-digit version check), the version is checked using the version ID in the block properties.
For INTERNAL (three-digit version check), the first version ID is searched for in the first comment field of the block and used for the version check.

Path: Path to the source to be imported. The path can also be specified relatively.

Example

Table 1-35 Code example

```xml
<Command ID="20"
  Name="ImportSimaticSource">
  <Parameter
    Name="TB_Name_of_added_Source"
    VersionCheckType="INTERNAL"
    Path="SIMATIC\EquipmentModules\Communication_LCom\Data\Sources\DFFBLComMachineCom.AWL" />
  <Parameter
    Name="TB_Name_of_added_Parameter_DB"
    VersionCheckType="INTERNAL"
    Path="SIMATIC\EquipmentModules\Communication_LCom\Data\Sources\DFFBLComParameter.AWL" />
</Command>
```
1.5 Information commands

1.5.1 GetAvailableSimaticDevices

All SIMATIC devices that are available in a directory for import are returned with this command. This command can be transferred directly to a control property via the __Call prefix. The command returns a collection (Arraylist) of SortedDictionary objects. These offer a "NAME" key for accessing the name of the SIMATIC station.

Command parameters

Transfer parameters
Version Specifies the main version of the device (S7300 or S7400)

Example

Table 1-36 Code example

Source="__call_GetAvailableSimaticDevices('S7300')"

Table 1-37 Use in the source code

REM Get all SIMATIC devices
Dim DeviceList As System.Collections.ArrayList = My-
App.myIsl.getavailablesimaticdevices('V3_2')
For Each tmpDevice As System.Collections.Generic.SortedDictionary(Of
String, String) in DeviceList
    MessageBox.Show(tmpDevice('NAME').ToString())
Next
1.5.2 GetAvailableSimaticDeviceVersions

All versions (S7300 or S7400) of the SIMATIC devices that are available for import are returned with this command. This command can be transferred directly to a control property via the __Call_ prefix. The command returns a collection (Arraylist) of SortedDictionary objects. These offer a "NAME" key for accessing the versions available.

Command parameters

This command does not have any parameters.

Example

Table 1- 38 Code example

Source="__call_GetAvailableSimaticDeviceVersions"

Table 1- 39 Use in the source code

REM Get all SIMATIC versions
Dim VersionList As System.Collections.ArrayList = MyApp.myIsl.getavailablesimaticdeviceversions()
    MessageBox.Show(devVersion('NAME').ToString())
Next
1.5.3 **GetDevicesInProject**

With this command, all SIMOTION and SIMATIC devices (not SINAMICS devices) are returned that are either already in the project or that should still be added with the ProjectGenerator. This command can be transferred directly to a control property via the `__Call` prefix. When called in the source code, a collection (Arraylist) of `SortedDictionary` objects is returned. These offer the following keys for accessing the properties: NAME, OBJECTID, TYPEID, TYPENAME, VERSION.

**Command parameters**

This command does not have any parameters.

**Example**

Table 1-40  Code example

```vbnet
Source = "__call_GetDevicesInProject"
```

Table 1-41  Use in the source code

```vbnet
Dim deviceList As System.Collections.ArrayList = MyApp.myIsl.GetDevicesInProject()
For Each tmpDevice As System.Collections.Generic.SortedDictionary(Of String, String) in deviceList
    MessageBox.Show(tmpDevice('NAME').ToString())
Next
```
1.5.4 GetSimaticDeviceName

The name of the active device is returned with this command. This command can be transferred directly to a control property via the __Call__ prefix. When used in the source code, the name is returned as a string.

Command parameters

This command does not have any parameters.

Example

Table 1-42 Code example

| Text | __call_GetSimaticDeviceName |

Table 1-43 Use in the source code

```vbnet
Dim deviceName As String
deviceName = MyApp.myIsl.GetSimaticDeviceName()
```
1.5.5 GetSimaticBlocks

All existing blocks in the project, e.g. for display in a *list box*, are returned with this command. This command can be transferred directly to a control property via the *Call_* prefix. When called in the source code, a collection (*ArrayList*) of *SortedDictionary* objects is returned. These offer the following keys for accessing the properties: NAME, ADDRESS, DATATYPE.

Command parameters

This command does not have any parameters.

Example

Table 1-44 Code example

```
Source="__call_GetSimaticBlocks"
```

Table 1-45 Use in the source code

```
Dim blocks As System.Collections.ArrayList = My-
App.myIs1.GetSimaticBlocks()
For Each tmpBlock As System.Collections.Generic.SortedDictionary(Of
String, String) in blocks
    MessageBox.Show(tmpBlock('NAME').ToString())
Next
```
1.5.6 **GetValueOfSimaticDBVariable**

This command reads out the currently valid value of a variable from a data block and returns it. If a value is not found either in the project or in the database, the *DefaultValue* is returned. This command can be transferred directly to a control property via the __Call__ prefix. When the command is used in Visual Basic .NET source codes, the value is returned as a string.

**Command parameters**

**Transfer parameters**
- **Name**: Name of the variable or constant
- **DBName**: Symbolic name of the block being searched for
- **DefaultValue**: Substitute value when no value is found

**Example**

Table 1-46 Code example

```plaintext
Checked="__call_GetValueOfSimaticDBVariable('config.boProductionModeActive','DBLPMLV3ModeStateManager',True)"
```

1.5.7 **IsSimaticS7300Station**

A type check of the station can be performed with this command. The response in the process code is type-specific. The result of the type check is returned as a value of the Boolean data type.

**Command parameters**

This command does not have any parameters.

**Example**

Table 1-47 Code example

```xml
<Click code="If MyApp.IsSimaticS7300Station Then
    MyApp.NextCommand(10)
Else
    MyApp.NextCommand(11)
End If
MyApp.NextCommand(60)
"/>
```
1.5 Information commands

1.5.8 GetValueOfTemporaryVariable

The value of a temporary variable is returned as a string with this command.

This command can be transferred directly to a control property via the __Call__ prefix. The returned value can also be converted to other data types.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the temporary variables.</td>
</tr>
<tr>
<td>Type</td>
<td>You can choose between the options LOCAL and GLOBAL.</td>
</tr>
<tr>
<td>Lokale</td>
<td>Temporary variables are deleted automatically when the next module is called; global variables are retained. If this parameter is not supplied, LOCAL is assumed automatically.</td>
</tr>
<tr>
<td>DefaultValue</td>
<td>Default value of the temporary variables if they are not found in the data management.</td>
</tr>
</tbody>
</table>

Note

Errors in conversion may occur when working with floating-comma numbers. See the code example below.
### Example

Table 1- 48  Code example

**Transfer to a control**

Text="__call__ GetValueOfTemporaryVariable(Hello,LOCAL,World_Not_Found)"

**Use in the source code**

REM define different global variables
MyApp.myISL.SetTemporaryVariable('DoubleValue','Global',3601.8)
MyApp.myISL.SetTemporaryVariable('DoubleValueAsStringWithDot','Global', '3601.8')
MyApp.myISL.SetTemporaryVariable('DoubleValueAsStringWithComma','Global', '3601,8')
MyApp.myISL.SetTemporaryVariable('StringValue','Global', 'SomeText')

Dim DoubleVariable As Double
Dim StringVariable As String

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValue' , 'Global', -1)
MsgBox (DoubleVariable)   REM No error and output correct (3601.8)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithComma',
'Global', -1)
MsgBox (DoubleVariable)   REM No error and output correct (3601.8)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithDot',
'Global', -1)
MsgBox (DoubleVariable)   REM No error but output not correct (36018)

DoubleVariable = MyApp.myISL.GetValueOfTemporaryVariable ('StringValue', 'Global', -1)
MsgBox (DoubleVariable)   REM error converting the String to a Double value

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValue', 'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error and output correct(3601.8)

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithDot',
'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error but output not correct(36018)

StringVariable = MyApp.myISL.GetValueOfTemporaryVariable ('DoubleValueAsStringWithComma',
'Global', -1)
DoubleVariable = Double.Parse(StringVariable)
MsgBox (DoubleVariable)   REM No error and output correct(3601.8)
1.5 Information commands

1.5.9 GetDeviceProperty

The value of every permissible HW Config property (attribute) of a device can be read out with this command. If it is the property of a module or a submodule, then the appropriate module (and submodule) number must be specified. As it may be that the required property is within a field, as with LocalInAddresses or LocalOutAddresses, it is also possible to optionally specify the name of the relevant object with ObjectType.

The command can only be executed within the VB code.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceName</td>
<td>Name of the device.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the HW Config property/attribute.</td>
</tr>
<tr>
<td>SlotNumber</td>
<td>Optional: Specification of the slot number for the addressing of subordinate slots.</td>
</tr>
<tr>
<td>SubSlotNumber</td>
<td>Optional: Specification of the subslot number for the addressing of subordinate slots.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>Optional: Specification of the object that is to be accessed, e.g. LocalInAddresses or LocalOutAddresses.</td>
</tr>
</tbody>
</table>

Example

Table 1-49  Code example

```xml
<Control
    Action="add"
    Type="Button"
    Name="BT_GetDeviceProperty"
    Text="GetDeviceProperty"
    Location="280, 360"
    Size="150, 30"
    Enabled="true"
    Visible="true"
    ToolTip="Test the function GetDeviceProperty">
    <Events>
        <Click code="
            DIM String_Value =
            MyApp.myIs1.GetDeviceProperty('MyET200', 'LogicalAddress', '2', '6', 'LocalInAddresses')
            "
        />
    </Events>
</Control>
```
1.6 Commands for the source and object manipulation

1.6.1 Commands for the source manipulation

1.6.1.1 SetLabel

A label is replaced in a complete source with this command. The label is supplemented by a leading "<" and following ">" character.

As precondition for this command, an ImportSimaticSource must be executed.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TargetName</td>
<td>Contains the name of the SIMATIC source</td>
</tr>
<tr>
<td>LabelName</td>
<td>Specifies the label that is replaced in the entire source.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value that is replaced.</td>
</tr>
<tr>
<td>Type</td>
<td>For access to SIMATIC sources, the type must be given as &quot;Simatic_Source_Label&quot;.</td>
</tr>
</tbody>
</table>

Example

Table 1- 50 Code example

```xml
<Command
  ID="80" Name="SetLabel"
  NCID="1000" >
  <Parameter
    TargetName="TB_Name_of_added_Source"
    LabelName="Inst_DB_Name"
    Value="TB_Name_of_added_Source"
    Type="Simatic_Source_Label"/>
</Command>
```
**SetSimaticValue**

With this command you can set variables of a block to a defined value that is also partially determined during runtime.

### Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SourceName</strong></td>
<td>Name of the data block source to which the value of a variable is to be added. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Name of the variable to be added. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Value of the variable to be added. The data type must correspond to the S7 data types. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
<tr>
<td><strong>Convert</strong></td>
<td>Converts the numerical value in the Value parameter to the given number system. Using the HEX tag, you can specify that the numerical value in the Value parameter should automatically be converted to a hexadecimal number.</td>
</tr>
</tbody>
</table>

### Example

**Table 1- 51 Code example**

```
<Command
   ID="50"
   Name="setSimaticValue"
   NCID="80" >
   <Parameter
       SourceName="TB_Name_of_added_Parameter_DB"
       Name="sParameter.sCfgConnection.boWithLComProtocol"
       Value="TRUE" />
   <Parameter
       SourceName="TB_Name_of_added_Parameter_DB"
       Name="sParameter.sCfgConnection.boIsTcpClient"
       Value="CB_Is_TCP_Client" />
   <Parameter
       SourceName="TB_Name_of_added_Parameter_DB"
       vName="sParameter.sCfgConnection.b16LocalPort"
       Value="TB_Local_port"
       Convert="HEX" />
</Command>
```
1.6.1.3  SetSymbolInSymbolTable

With this command, values can be added to the symbol table of the project and changed. Any number of lines may be changed or expanded with a single call. The parameter line can be filled in several times for this purpose.

A source may only contain one block.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SymbolicName</td>
<td>Symbolic name of the tag. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
<tr>
<td>BlockNumber</td>
<td>Block number of the tag. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
<tr>
<td>BlockType</td>
<td>Type of tag: DATA_BLOCK / FUNCTION / FUNCTION_BLOCK</td>
</tr>
<tr>
<td>InstanceFBNumber</td>
<td>For an instance data block, the FB-Number reference must be entered here. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.</td>
</tr>
</tbody>
</table>

Example

Table 1-52  Code example

```xml
<Command
   ID="40"
   Name="SetSymbolInSymbolTable"
   NCID="70">
   <Parameter
       SymbolicName="TB_Name_of_added_Source"
       BlockNumber="TB_Lcom_FB_Number"
       BlockType="DATA_BLOCK"
       InstanceFBNumber="105"/>
   <Parameter
       SymbolicName="TB_Name_of_added_Parameter_DB"
       BlockNumber="TB_Lcom_Param"
       BlockType="DATA_BLOCK"/>
</Command>
```
1.6 Commands for the object manipulation

1.6.2.1 InsertNetworksIntoOB

This command allows code networks to be added to any organization block. This command also checks whether the code is already available. If so, it is not added again. The code to be added can also contain labels that can be replaced by calling the SetLabel subparameter. Calling the parameter and subparameter lines can be repeated as often as required in a command call.

Command parameters

Parameter

Name
Name of organization block to which the code networks are to be added. The value can also be assigned a reference to a user interface element. The Text property of the element is evaluated here.

Path
Path to the code networks to be imported. The path can also be specified relatively.

Position
The position for adding the new networks can be given here. Only FIRST and LAST are allowed. If no position is specified, the new networks are attached.

DummyPath
If a new OB is created, an empty export of the block must be present that can be imported as a default block. The path to the exported source must be specified here.

Subparameter SetLabel

LabelName
Specifies the label that is replaced in the entire source.

Value
Specifies the value that is replaced.
Example

Table 1-53  Code example

```xml
<Command
   ID="60"
   Name="InsertNetworksIntoOB"
   NCID="41" >
   <Parameter
      Name="OB1"
      Path="SIMATIC\EquipmentModules\Communication_LCom\Data\Templates\OB1Call.txt"
      Position=""
      DummyPath="SIMATIC\EquipmentModules\Communication_LCom\Data\Templates\OB1.AWL">
      <SetLabel
         LabelName="Inst_DB_Name"
         Value="TB_Name_of_added_Sources"/>
      <SetLabel
         LabelName="Send_DB_Name"
         Value="TB_Name_of_added_Send_DB"/>
      <SetLabel
         LabelName="Receive_DB_Name"
         Value="TB_Name_of_added_Receive_DB"/>
      <SetLabel
         LabelName="Param_DB_Name"
         Value="TB_Name_of_added_Parameter_DB"/>
   </Parameter>
</Command>
```
1.6.2.2 InsertVariablesIntoOB

This command allows variables to be added to any organization block. This command also checks whether the variables are already available. If so, the variables are not added again.

Command parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the organization block to which the variable is to be added. The value can also be assigned a reference to a user interface element. The &quot;Text&quot; property of the element is evaluated here.</td>
</tr>
<tr>
<td>Variable</td>
<td>Name of the variable to be added. The name must not be longer than 24 characters. The value can also be assigned a reference to a user interface element. The &quot;Text&quot; property of the element is evaluated here.</td>
</tr>
<tr>
<td>DataType</td>
<td>Data type of the variable to be added. The data type must correspond to the S7 data types. The value can also be assigned a reference to a user interface element. The &quot;Text&quot; property of the element is evaluated here.</td>
</tr>
<tr>
<td>Comment</td>
<td>Comment of the variable to be added. The value can also be assigned a reference to a user interface element. The &quot;Text&quot; property of the element is evaluated here.</td>
</tr>
</tbody>
</table>

Example

Table 1- 54  Code example

```xml
<Command
   ID="70"
   Name="InsertVariableIntoOB"
   NCID="50">
   <Parameter
      Name="OB1"
      Variable="boInitDone"
      DataType="BOOL"
      Comment=""/>
   <Parameter
      Name="OB1"
      Variable="b16ErrorId"
      DataType="WORD"
      Comment=""/>
</Command>
```
1.6.2.3 SetDeviceProperty

The value of every permissible HW Config property (attribute) of a device can be set with this command. If it is the property of a module or a submodule, then the appropriate module (and submodule) number must be specified. It may be that the required property is within a field, as with LocalInAddresses or LocalOutAddresses. It is possible to optionally specify the name of the relevant object with ObjectType.

Command parameters

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceName</td>
<td>Specification of the sought device name.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the property/attribute.</td>
</tr>
<tr>
<td>Value</td>
<td>The value that is to be assigned.</td>
</tr>
<tr>
<td>SlotNumber</td>
<td>Optional: Specification of the slot number for the addressing of subordinate slots.</td>
</tr>
<tr>
<td>SubSlotNumber</td>
<td>Optional: Specification of the subslot number for the addressing of subordinate subslots.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>Optional: Specification of the object to be addressed, e.g. Local-InAddresses or Local-OutAddresses.</td>
</tr>
</tbody>
</table>

Example

Table 1-55 Code example

```
<Command ID="301" Name="SetDeviceProperty">
  <Parameter
    DeviceName="MyET200"
    SlotNumber="3"
    SubSlotNumber=""
    Name="LocalAddress"
    Value="1500"
    ObjectType="LocalOutAddresses"
  />
</Command>
```
ConfigureProfinetIRT

The settings required for the IRT communication can be made with this command. All IRT-capable devices are set up as sync slave and the controller as sync master in the respective PROFINET IO system.

The times specified in the parameters are set and all IRT-capable slots activated as slaves. The Ti/To mode is set to "IO Device" and the specified times set for all devices.

Command parameters

Parameter
DeviceName  Name of the controller to be configured as sync master.
CPUInterface  Specification of the controller interface on which the subsystem to be configured is located.
SyncDomain  Optional: Name of the sync domain. If this parameter is not specified, the standard sync domain is used.
CycleTime  Bus cycle time in ms.
Ti  Time to read-in the process values in µs.
To  Time to output the process values in µs.

Example

Table 1- 56  Code example

```xml
<Command ID="303" Name="ConfigureProfinetIRT">
  <Parameter
    DeviceName="newDevice"
    CPUInterface="PN1"
    SyncDomain=""
    CycleTime ="4000"
    Ti="125"
    To="250"
  />
</Command/>
```
1.6.2.5 AddPNDevice

Arbitrary PROFINET devices can be inserted in a PROFINET IO system by means of the MLFB or GSDML file with this command.

In special cases, a prefix must be attached to the MLFB. For this reason, it is recommended that an export is created of the relevant station from HW Config and that the MLFB specified there is used including the prefix.

If a GSDML file is used, it must already be installed in the hardware configuration. If more than one device or module is defined in the GSDML file, the appropriate ID from the GSDML file must be specified for the device.

Optionally, as many modules and submodules as permitted by the engineering system can be inserted in this device. The Module and SubModule elements can be specified a corresponding number of times.

Limitation of the possible ET200 stations: See Projectgenerator supplementary conditions.

Command parameters

Parameter

DeviceName Name to be assigned for the inserted device.
DeviceIdentifier Specification of the MLFB or GSDML file.
ID Optional: Specification of the ID for the specification of the device from the GSDML file.
IPAddress Specification of the IP address (also possible as hex value).
  For example: 192.168.0.6
  C0A80006
CPUInterface Specification of the controller interface on which the subsystem in which the device is to be inserted is located.
Version Optional: Specification of the version if the DeviceIdentifier parameter is an MLFB.
DeviceNumber Optional: Specification of the device number in the PROFINET IO system if a number other than the next free number is desired.

Module

ModuleName Name to be assigned for the inserted module.
ModuleIdentifier Specification of the MLFB or GSDML file.
ModuleID Optional: Specification of the ID for the specification of the module from the GSDML file.
SlotNumber Specification of the slot number for the addressing of subordinate slots.

SubModule

SubModuleName Name to be assigned for the inserted submodule.
SubModuleIdentifier Specification of the MLFB or GSDML file.
SubModuleID Optional: Specification of the ID for the specification of the submodule from the GSDML file.
Command overview
1.6 Commands for the source and object manipulation

| SlotNumber | Specification of the slot number for the addressing of subordinate slots. |
| SubSlotNumber | Specification of the subslot number for the addressing of subordinate subslots. |

Example

Table 1-57 Code example

```xml
<Command ID="307" Name="AddPNDevice">
  <Parameter
    DeviceName="__ref_TB_GSDName.Text"
    DeviceIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
    ID="DIM_20"
    IPAddress="192.168.1.6"
    CPUInterface="PNI"
    Version="V2.0"
    Address="2">
    <Module
      ModuleName="DI 16x24VDC ST"
      ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
      ModuleID="DI 16x24VDC ST"
      SlotNumber ="1"/>
    <Module
      ModuleName="DI 16x24VDC ST"
      ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
      ModuleID="DI 16x24VDC ST"
      SlotNumber ="2"/>
    <Module
      ModuleName="DI 16x24VDC ST"
      ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
      ModuleID="DI 16x24VDC ST"
      SlotNumber ="3"/>
  </Parameter>
</Command>
```
1.6.2.6 **AddPNDeviceModule**

Individual modules can be added to a PROFINET device by means of the GSDML file with this command.

As prerequisite, the GSDML file to be used must already be installed in HW Config.

Any number of modules and submodules can be inserted in a device with a single command.

A module can be created with the **Module** element. Several modules can be created through multiple use of the element.

A submodule can be created with the **SubModule** element. Several submodules can be created through multiple use of the element.

**Command parameters**

| Parameter       | Description                                                                 |
|-----------------|                                                                            |
| **DeviceName**  | Name of the device in which the modules or submodules are to be inserted.  |
| **ModuleName**  | Name to be assigned for the inserted module.                              |
| **ModuleIdentifier** | Specification of the MLFB or GSDML file.                                  |
| **ModuleID**    | Specification of the ID for the specification of the module from the GSDML file. |
| **SlotNumber**  | Specification of the slot number for the addressing of subordinate slots. |

**SubModule**

| Parameter         | Description                                                                 |
|-------------------|                                                                            |
| **SubModuleName** | Name to be assigned for the inserted submodule.                           |
| **SubModuleIdentifier** | Specification of the MLFB or GSDML file.                                  |
| **SubModuleID**   | Optional: Specification of the ID for the specification of the submodule from the GSDML file. |
| **SlotNumber**    | Specification of the slot number for the addressing of subordinate slots. |
| **SubSlotNumber** | Specification of the subslot number for the addressing of subordinate subslots. |
Example

Table 1- 58   Code example

<Command ID="310" Name="AddPNDeviceModule">
    <Parameter>
        DeviceName ="__ref_TB_GSDName.Text">
        <Module>
            ModuleName="DI 16x24VDC ST"
            ModuleIdentifier="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            ModuleID="DI 16x24VDC ST"
            SlotNumber ="1"/>
        <SubModule>
            SubModuleName ="My Port 1 (2xRJ45)"
            SubModuleIdentifier ="GSDML-V2.31-Siemens-ET200SP-20150326.xml"
            SubModuleID ="IDS_1P1 HF RJ45 V2.2"
            SlotNumber ="0"
            SubSlotNumber ="2"/>
        </Parameter>
    </Command>
1.6.2.7 **CreateET200Module**

Individual modules of an ET200 can be created and configured with this command.

**Command parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceName</td>
<td>Name of the ET200 station.</td>
</tr>
<tr>
<td>ModuleName</td>
<td>Name of the module.</td>
</tr>
<tr>
<td>MLFB</td>
<td>MLFB of the module.</td>
</tr>
<tr>
<td>SlotNumber</td>
<td>Slot number in which the module is to be inserted.</td>
</tr>
<tr>
<td>InAddress</td>
<td>Optional: Start address of the module.</td>
</tr>
<tr>
<td>InAddressSMType</td>
<td>Optional: Is only used for ET 200L and ET 200S modules and has the following coding:</td>
</tr>
<tr>
<td></td>
<td>16: Digital addresses</td>
</tr>
<tr>
<td></td>
<td>32: Analog addresses</td>
</tr>
<tr>
<td></td>
<td>256: Diagnostic addresses</td>
</tr>
<tr>
<td>InAddressLength</td>
<td>Optional:</td>
</tr>
<tr>
<td>InSubAddress</td>
<td>Optional: Subaddress of the module.</td>
</tr>
<tr>
<td>InPartProcessImage</td>
<td>Optional: Number of the assigned part process image.</td>
</tr>
<tr>
<td>InArea</td>
<td>Optional: Area identifier.</td>
</tr>
<tr>
<td></td>
<td>0: SIMATIC 400 module</td>
</tr>
<tr>
<td></td>
<td>1: SIMATIC 300 module</td>
</tr>
<tr>
<td></td>
<td>2: DP addressing</td>
</tr>
<tr>
<td></td>
<td>3: P area (S5 connection)</td>
</tr>
<tr>
<td></td>
<td>4: Q area (S5 connection)</td>
</tr>
<tr>
<td></td>
<td>5: IM3 area (S5 connection)</td>
</tr>
<tr>
<td></td>
<td>6: IM4 area (S5 connection)</td>
</tr>
<tr>
<td></td>
<td>7: Reserved</td>
</tr>
<tr>
<td>InBitAddress</td>
<td>Optional: The bit offset to the start address is stored here for ET 200 and ET 200S electronic modules.</td>
</tr>
<tr>
<td>OutAddress</td>
<td>Optional: Specification of the logical address.</td>
</tr>
<tr>
<td>OutAddressSMType</td>
<td>Optional: Is only used for ET 200L and ET 200S modules and has the following coding:</td>
</tr>
<tr>
<td></td>
<td>16: Digital addresses</td>
</tr>
<tr>
<td></td>
<td>32: Analog addresses</td>
</tr>
<tr>
<td></td>
<td>256: Diagnostic addresses</td>
</tr>
<tr>
<td>OutAddressLength</td>
<td>Optional: Length of the address in bytes (depending on the module)</td>
</tr>
<tr>
<td>OutSubAddress</td>
<td>Optional: Subaddress of the module.</td>
</tr>
<tr>
<td>OutPartProcessImage</td>
<td>Optional: Number of the assigned part process image.</td>
</tr>
</tbody>
</table>
OutArea

Optional: Area identifier.
0: SIMATIC 400 module
1: SIMATIC 300 module
2: DP addressing
3: P area (S5 connection)
4: Q area (S5 connection)
5: IM3 area (S5 connection)
6: IM4 area (S5 connection)
7: Reserved

OutBitAddress

Optional: The bit offset to the start address is stored here for ET 200 and ET 200S electronic modules.

Example

Table 1- 59  Code example

```xml
<Command ID="321" Name="CreateEt200Module">
  <Parameter ET200Name="MyET200"
    ModuleName="TestxCreateEt200Module"
    MLFB="6ES7 131-4BF50-0AA0" Version="V7.0"
    SlotNumber="4"
    InAddress="42"
    InAddressSMType=""
    InAddressLength=""
    InSubAddress=""
    InPartProcessImage=""
    InArea=""
    InBitAddress=""
    OutAddress=""
    OutAddressSMType=""
    OutAddressLength=""
    OutSubAddress=""
    OutPartProcessImage=""
    OutArea=""
    OutBitAddress=""
  />
</Command>
```
1.6.2.8 CreatePNTopology

The PROFINET topology can be configured with this command. The ports are connected according to the parameter specifications. Whereby, the Device as well as the PartnerDevice can be a station or a slave.

The Parameter element can be used several times for multiple interconnections (see example).

Command parameters

Parameter

- **DeviceName**: Name of the controller/device.
- **PNInterfaceName**: Name of the PROFINET interface.
- **PortNumber**: Number of the port.
- **PartnerDeviceName**: Name of the partner controller/device.
- **PartnerPNInterfaceName**: Name of the partner PROFINET interface.
- **PartnerPortNumber**: Number of the partner port.

Example

Table 1- 60  Code example

```xml
<Command ID="305" Name="CreatePNTopology">
  <Parameter
    DeviceName ="newDevice"
    PNInterfaceName ="PNxIO"
    PortNumber ="3"
    PartnerDeviceName ="NewCU"
    PartnerPNInterfaceName ="PN-IO"
    PartnerPortNumber ="1"
  />
  <Parameter
    DeviceName ="newDevice"
    PNInterfaceName ="PNxIO"
    PortNumber ="2"
    PartnerDeviceName ="GSDxDevice"
    PartnerPNInterfaceName ="Interface"
    PartnerPortNumber ="1"
  />
</Command>
```
1.6.2.9 **CreateProfinetSubsystem**

PROFINET IO systems can be created at the specified interface of the active controller with this command.

If a subnet with the specified name already exists, it is linked to the PROFINET IO system. Otherwise a new subnet is created and linked.

The name of the new subsystem is "PROFINET IO System" and is assigned the next free IO system number.

**Command parameters**

- **Parameter**
  - **CPUInterface**: Specification of the controller interface on which the subsystem in which the device is to be inserted is located.
  - **SubnetName**: Name of the subnet.

**Example**

```xml
<Command ID="319" Name="CreateProfinetSubsystem">
  <Parameter
    SubnetName ="Ethernet(101)"
    CPUInterface ="PN1"
  />
</Command>
```
A.1 Contacts

Siemens AG
Digital Factory
Factory Automation
Production Machines
DF FA PMA APC
Frauenauracher Strasse 80
D-91056 Erlangen, Germany
Fax: +49 9131 98 1297
tech.team.motioncontrol@siemens.com
A.2 Internet addresses

Additional information on various topics is provided on the following Internet pages.

See also

SIMOTION (www.siemens.com/simotion)
SINAMICS (www.siemens.com/sinamics)
Motion Control / Application Center (www.siemens.com/motioncontrol/apc)
Packaging (www.siemens.com/packaging)