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
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

⚠️ CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

⚠️ NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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 for the specific task, 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 adhered to. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of the 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

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in a separate list of references. This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises 10 documentation packages.

The following documentation packages are available for SIMOTION V4.2:

- SIMOTION Engineering System
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation/overview of documentation
- Additional links to download documents
- Using documentation online (find and search in manuals/information)

http://www.siemens.com/motioncontrol/docu

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following e-mail address:

docu.motioncontrol@siemens.com

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt this for the purpose of your own machine documentation:

http://www.siemens.com/mdm
Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:
http://www.siemens.com/sitrain

FAQs

You can find Frequently Asked Questions on the Service&Support pages under Product Support:
http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under Contact:
http://www.siemens.com/automation/service&support
# Table of contents

**Preface** .................................................................................................................................................. 3

1 **Introduction** .................................................................................................................................... 11
   1.1 Overview .................................................................................................................................... 11
   1.2 Schematic diagram of the function packages in the SIMOTION device ........................................ 12
   1.3 Form of delivery ......................................................................................................................... 13
   1.4 Possible applications .................................................................................................................. 14
       1.4.1 Standard information ........................................................................................................ 14
       1.4.2 User-defined information ............................................................................................... 15

2 **Commissioning** .............................................................................................................................. 17
   2.1 Hardware and software requirements ....................................................................................... 17
   2.2 Configuring the SIMOTION device interface ............................................................................ 17
   2.3 Log-in administration ................................................................................................................. 18
   2.4 Setting the language for AlarmS and user-defined diagnostics buffer messages .................. 22

3 **Operation (software)** ...................................................................................................................... 23
   3.1 IT DIAG overview and general functions ................................................................................. 23
       3.1.1 Overview .......................................................................................................................... 23
       3.2 Standard pages ..................................................................................................................... 24
           3.2.1 Home ............................................................................................................................... 24
           3.2.2 Device Info ..................................................................................................................... 26
           3.2.2.1 Device Info .................................................................................................................. 26
           3.2.2.2 IP Config ...................................................................................................................... 28
           3.2.3 Diagnostics ..................................................................................................................... 29
               3.2.3.1 Diagnostics ............................................................................................................. 29
               3.2.3.2 Task runtime .......................................................................................................... 30
               3.2.3.3 Service overview ................................................................................................. 31
               3.2.3.4 Watch ...................................................................................................................... 34
               3.2.3.5 Trace (device trace) ............................................................................................... 38
               3.2.3.6 Trace (system trace) ............................................................................................... 42
               3.2.3.7 Tasktrace ................................................................................................................. 46
           3.2.4 Messages&Logs ................................................................................................................. 49
               3.2.4.1 Diag buffer ............................................................................................................... 52
               3.2.4.2 Diag buffer drive ................................................................................................. 54
               3.2.4.3 Alarms ...................................................................................................................... 55
               3.2.4.4 Alarms drive .......................................................................................................... 58
               3.2.4.5 Alarm buffer ......................................................................................................... 59
               3.2.4.6 Syslog ...................................................................................................................... 60
               3.2.4.7 Userlog .................................................................................................................... 61
           3.2.5 Machine overview ............................................................................................................. 62
               3.2.5.1 Module information ............................................................................................... 62

SIMOTION IT Ethernet-based HMI and Diagnostic Functions
Diagnostics Manual, 11/2010
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.5.2 Topology</td>
<td>64</td>
</tr>
<tr>
<td>3.2.5.3 Topology table</td>
<td>65</td>
</tr>
<tr>
<td>3.2.5.4 Overview</td>
<td>66</td>
</tr>
<tr>
<td>3.2.5.5 Configuration</td>
<td>67</td>
</tr>
<tr>
<td>3.2.6 Manage config</td>
<td>68</td>
</tr>
<tr>
<td>3.2.6.1 Device update</td>
<td>68</td>
</tr>
<tr>
<td>3.2.6.2 Upgrading firmware prior to V4.2</td>
<td>70</td>
</tr>
<tr>
<td>3.2.6.3 Editing function</td>
<td>71</td>
</tr>
<tr>
<td>3.2.6.4 IT DIAG tab</td>
<td>74</td>
</tr>
<tr>
<td>3.2.6.5 IT DIAG Base</td>
<td>74</td>
</tr>
<tr>
<td>3.2.6.6 IT DIAG Server options</td>
<td>76</td>
</tr>
<tr>
<td>3.2.6.7 IT DIAG Mimetypes</td>
<td>77</td>
</tr>
<tr>
<td>3.2.6.8 IT DIAG configuration data</td>
<td>78</td>
</tr>
<tr>
<td>3.2.6.9 IT DIAG System</td>
<td>79</td>
</tr>
<tr>
<td>3.2.6.10 IT DIAG WebCfg transmission</td>
<td>81</td>
</tr>
<tr>
<td>3.2.6.11 IT DIAG text databases</td>
<td>82</td>
</tr>
<tr>
<td>3.2.7 Settings</td>
<td>84</td>
</tr>
<tr>
<td>3.2.8 Files</td>
<td>87</td>
</tr>
<tr>
<td>3.2.8.1 Files</td>
<td>87</td>
</tr>
<tr>
<td>3.2.8.2 Proc</td>
<td>89</td>
</tr>
<tr>
<td>3.2.9 Watch tables</td>
<td>90</td>
</tr>
<tr>
<td>3.3 Simplified standard pages</td>
<td>91</td>
</tr>
<tr>
<td>3.3.1 BASIC pages</td>
<td>91</td>
</tr>
<tr>
<td>3.3.2 Device Info</td>
<td>92</td>
</tr>
<tr>
<td>3.3.3 Diagnostics</td>
<td>94</td>
</tr>
<tr>
<td>3.3.4 Diag buffer</td>
<td>95</td>
</tr>
<tr>
<td>3.3.5 Diag buffer drive</td>
<td>96</td>
</tr>
<tr>
<td>3.3.6 Alarms</td>
<td>97</td>
</tr>
<tr>
<td>3.3.7 IP Config</td>
<td>98</td>
</tr>
<tr>
<td>3.3.8 Diagnostic files</td>
<td>99</td>
</tr>
<tr>
<td>3.3.9 Watch tables</td>
<td>100</td>
</tr>
<tr>
<td>3.3.10 User's Area</td>
<td>102</td>
</tr>
<tr>
<td>3.4 IT DIAG configuration</td>
<td>103</td>
</tr>
<tr>
<td>3.4.1 Introduction</td>
<td>103</td>
</tr>
<tr>
<td>3.4.2 Overview</td>
<td>104</td>
</tr>
<tr>
<td>3.4.3 Configuration of the file system</td>
<td>106</td>
</tr>
<tr>
<td>3.4.3.1 Virtual file system</td>
<td>106</td>
</tr>
<tr>
<td>3.4.3.2 Virtual vs. physical file system</td>
<td>106</td>
</tr>
<tr>
<td>3.4.3.3 Set external file system as preferential <code>&lt;PREFER_EXTERNAL&gt;</code></td>
<td>107</td>
</tr>
<tr>
<td>3.4.3.4 Links to the physical file system <code>&lt;LOCALLINK&gt;</code></td>
<td>108</td>
</tr>
<tr>
<td>3.4.3.5 Browsing of directories</td>
<td>109</td>
</tr>
<tr>
<td>3.4.3.6 Security concept</td>
<td>110</td>
</tr>
<tr>
<td>3.4.3.7 REALM</td>
<td>112</td>
</tr>
<tr>
<td>3.4.3.8 READ</td>
<td>113</td>
</tr>
<tr>
<td>3.4.3.9 WRITE</td>
<td>114</td>
</tr>
<tr>
<td>3.4.3.10 MODIFY</td>
<td>114</td>
</tr>
<tr>
<td>3.4.3.11 Creating directories and files</td>
<td>115</td>
</tr>
<tr>
<td>3.4.3.12 Browsing the file system</td>
<td>115</td>
</tr>
<tr>
<td>3.4.3.13 File access via FTP</td>
<td>116</td>
</tr>
<tr>
<td>3.5 User-defined pages</td>
<td>118</td>
</tr>
<tr>
<td>3.5.1 Introduction</td>
<td>118</td>
</tr>
<tr>
<td>3.5.2 Converting standard HTML pages to binary files</td>
<td>119</td>
</tr>
</tbody>
</table>
Table of contents

3.5.3 User-defined Home page ........................................................................................................... 121
3.5.4 Embedded, user-defined pages ................................................................................................ 122
3.5.5 Menu editor .............................................................................................................................. 124
3.5.6 JavaScript and web services ...................................................................................................... 127
3.5.6.1 Variable access with JavaScript and web services ................................................................ 127
3.5.6.2 Communication with the OPC XML DA server (opcxml.js) .................................................... 128
3.5.6.3 Representation of OPC XML DA data in the browser (appl.js) ................................................. 145
3.5.7 MiniWeb Server Language (MWSL) ......................................................................................... 158
3.5.7.1 Mode of operation of the MWSL ............................................................................................ 158
3.5.7.2 Structure of a MWSL file ....................................................................................................... 159
3.5.7.3 Error messages ....................................................................................................................... 160
3.5.7.4 Variable types ......................................................................................................................... 161
3.5.7.5 Script variables ....................................................................................................................... 162
3.5.7.6 Global variables ....................................................................................................................... 164
3.5.7.7 Configuration constants ......................................................................................................... 166
3.5.7.8 Variables and URL parameters ............................................................................................... 167
3.5.7.9 COOKIES ............................................................................................................................... 169
3.5.7.10 Variables and access to COOKIES ....................................................................................... 170
3.5.7.11 Variables and HTTP header information ............................................................................. 170
3.5.7.12 Operators .............................................................................................................................. 172
3.5.7.13 For.......................................................................................................................................... 173
3.5.7.14 If 174
3.5.7.15 Overview of MWSL functions ............................................................................................ 176
3.5.7.16 Mode of operation of the template mechanism .................................................................... 177
3.5.7.17 Structure of the template file ................................................................................................. 178
3.5.7.18 Structure of a data source ..................................................................................................... 179
3.5.7.19 Template transformation .................................................................................................... 181
3.5.7.20 MWSL in XML attributes ................................................................................................... 186
3.5.7.21 Examples .............................................................................................................................. 187
3.5.8 User-defined directory page .................................................................................................... 198
3.5.9 Server Side Includes (SSI) ........................................................................................................ 203
3.5.9.1 Integration of process values ................................................................................................. 203
3.6 OPC XML DA server .................................................................................................................. 204
3.6.1 Overview ................................................................................................................................. 204
3.6.2 Comparison of OPC XML DA / SIMATIC NET OPC DA ........................................................ 206
3.6.3 Schematic diagram at the design stage ..................................................................................... 207
3.6.4 Schematic diagram at runtime ................................................................................................ 208
3.6.5 Installation ............................................................................................................................... 209
3.6.5.1 Hardware and software requirements at the design stage .................................................... 209
3.6.5.2 Configuring SIMOTION device interface at runtime ............................................................ 210
3.6.6 Making unit variables available ............................................................................................... 210
3.6.7 Example of a client application ............................................................................................... 212
3.6.8 SIMOTION IT OPC XML DA server interface ........................................................................ 215
3.6.8.1 Overview ............................................................................................................................... 215
3.6.8.2 Methods which can be called synchronously ....................................................................... 215
3.6.8.3 Access to variables ............................................................................................................... 217
3.7 Trace interface via SOAP (TVS) ................................................................................................. 218
3.7.1 Trace Interface via SOAP ......................................................................................................... 219
3.7.2 Procedure/terms ....................................................................................................................... 220
3.7.3 Error handling .......................................................................................................................... 221
3.7.4 Basics of subscriptions ............................................................................................................ 222

SIMOTION IT Ethernet-based HMI and Diagnostic Functions
Diagnostics Manual, 11/2010
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7.5</td>
<td>Interface</td>
<td>224</td>
</tr>
<tr>
<td>3.7.5.1</td>
<td>Global definitions</td>
<td>224</td>
</tr>
<tr>
<td>3.7.5.2</td>
<td>Methods</td>
<td>227</td>
</tr>
<tr>
<td>3.7.5.3</td>
<td>Subscriptions</td>
<td>233</td>
</tr>
<tr>
<td>3.8</td>
<td>Variables providers</td>
<td>235</td>
</tr>
<tr>
<td>3.8.1</td>
<td>Overview</td>
<td>235</td>
</tr>
<tr>
<td>3.8.2</td>
<td>SIMOTION</td>
<td>235</td>
</tr>
<tr>
<td>3.8.2.1</td>
<td>Accessing system variables/technology object system variables</td>
<td>236</td>
</tr>
<tr>
<td>3.8.2.2</td>
<td>Accessing technology object configuration data (V4.1 and higher)</td>
<td>237</td>
</tr>
<tr>
<td>3.8.2.3</td>
<td>Accessing drive parameters (V4.1 and higher)</td>
<td>237</td>
</tr>
<tr>
<td>3.8.2.4</td>
<td>Accessing technological alarms (V4.1 and higher)</td>
<td>238</td>
</tr>
<tr>
<td>3.8.2.5</td>
<td>Changing the operating mode (V4.1 and higher)</td>
<td>239</td>
</tr>
<tr>
<td>3.8.2.6</td>
<td>RamToRom (V4.1 and higher)</td>
<td>239</td>
</tr>
<tr>
<td>3.8.2.7</td>
<td>ActiveToRam (V4.1 and higher)</td>
<td>240</td>
</tr>
<tr>
<td>3.8.2.8</td>
<td>Accessing the global variables (V4.2 and higher)</td>
<td>241</td>
</tr>
<tr>
<td>3.8.2.9</td>
<td>Accessing the IO variables (V4.2 and higher)</td>
<td>242</td>
</tr>
<tr>
<td>3.8.2.10</td>
<td>Accessing the AlarmS messages (V4.2 and higher)</td>
<td>243</td>
</tr>
<tr>
<td>3.8.3</td>
<td>SIMOTION diagnostics</td>
<td>244</td>
</tr>
<tr>
<td>3.8.3.1</td>
<td>Introduction</td>
<td>244</td>
</tr>
<tr>
<td>3.8.3.2</td>
<td>DeviceInfo group</td>
<td>244</td>
</tr>
<tr>
<td>3.8.3.3</td>
<td>ComplInfo group</td>
<td>245</td>
</tr>
<tr>
<td>3.8.3.4</td>
<td>CPUload group</td>
<td>247</td>
</tr>
<tr>
<td>3.8.3.5</td>
<td>MemoryLoad group</td>
<td>248</td>
</tr>
<tr>
<td>3.8.3.6</td>
<td>TaskRT group</td>
<td>248</td>
</tr>
<tr>
<td>3.8.3.7</td>
<td>DiagBuffer group</td>
<td>250</td>
</tr>
<tr>
<td>3.8.3.8</td>
<td>DiagBufferDrv group</td>
<td>253</td>
</tr>
<tr>
<td>3.8.3.9</td>
<td>Alarms group</td>
<td>254</td>
</tr>
<tr>
<td>3.8.3.10</td>
<td>AlarmsDrv group</td>
<td>255</td>
</tr>
<tr>
<td>3.8.3.11</td>
<td>ActiveTraces group</td>
<td>255</td>
</tr>
<tr>
<td>3.8.3.12</td>
<td>Watch group</td>
<td>256</td>
</tr>
<tr>
<td>3.8.3.13</td>
<td>Comparison with the device diagnostics of SIMOTION SCOUT</td>
<td>257</td>
</tr>
<tr>
<td>3.8.4</td>
<td>UserConfig</td>
<td>259</td>
</tr>
<tr>
<td>3.8.4.1</td>
<td>User-defined variables</td>
<td>259</td>
</tr>
<tr>
<td>3.9</td>
<td>Secure Socket Layer</td>
<td>260</td>
</tr>
<tr>
<td>3.9.1</td>
<td>Key files</td>
<td>261</td>
</tr>
<tr>
<td>3.9.2</td>
<td>Transferring the key files to the SIMOTION control</td>
<td>261</td>
</tr>
<tr>
<td>3.9.3</td>
<td>Creating key files with script (V4.1 and higher)</td>
<td>262</td>
</tr>
<tr>
<td>3.9.3.1</td>
<td>Importing the certificate into the browser</td>
<td>264</td>
</tr>
</tbody>
</table>

### List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>WebCfg.xml</td>
</tr>
<tr>
<td>5.1.1</td>
<td>&lt; ALTERNATE_PORTNUMBER &gt;</td>
</tr>
<tr>
<td>5.1.2</td>
<td>&lt; ALTERNATE_SSL_PORTNUMBER &gt;</td>
</tr>
<tr>
<td>5.1.3</td>
<td>&lt; BASE &gt;</td>
</tr>
<tr>
<td>5.1.4</td>
<td>&lt; BROWSEABLE &gt;</td>
</tr>
<tr>
<td>5.1.5</td>
<td>&lt; CONFIGURATION_DATA &gt;</td>
</tr>
<tr>
<td>5.1.6</td>
<td>&lt; DEFAULTDOCUMENT &gt;</td>
</tr>
<tr>
<td>5.1.7</td>
<td>&lt; MIME_TYPES &gt;</td>
</tr>
</tbody>
</table>
### Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.8</td>
<td>&lt;PORTNUMBER&gt;</td>
</tr>
<tr>
<td>5.1.9</td>
<td>&lt;SERVEROPTIONS&gt;</td>
</tr>
<tr>
<td>5.1.10</td>
<td>&lt;SSLPORNUMBER&gt;</td>
</tr>
<tr>
<td>5.1.11</td>
<td>&lt;TIMEZONE&gt;</td>
</tr>
<tr>
<td>5.1.12</td>
<td>&lt;USERDATABASE&gt;</td>
</tr>
<tr>
<td>5.1.13</td>
<td>BROWSEABLE attribute</td>
</tr>
<tr>
<td>5.1.14</td>
<td>LOCALLINK attribute</td>
</tr>
<tr>
<td>5.1.15</td>
<td>MODIFY attribute</td>
</tr>
<tr>
<td>5.1.16</td>
<td>PREFER_EXTERNAL attribute</td>
</tr>
<tr>
<td>5.1.17</td>
<td>READ attribute</td>
</tr>
<tr>
<td>5.1.18</td>
<td>REALM attribute</td>
</tr>
<tr>
<td>5.1.19</td>
<td>WRITE attribute</td>
</tr>
<tr>
<td>5.2</td>
<td>WebCfgFrame.xml</td>
</tr>
<tr>
<td>5.2.1</td>
<td>&lt;BASE&gt;</td>
</tr>
<tr>
<td>5.2.2</td>
<td>&lt;CONVERSION&gt;</td>
</tr>
<tr>
<td>5.2.3</td>
<td>&lt;DEFAPP&gt;</td>
</tr>
<tr>
<td>5.2.4</td>
<td>&lt;HTTP_RESULT_CODES&gt;</td>
</tr>
<tr>
<td>5.2.5</td>
<td>&lt;SOAPAPP&gt;</td>
</tr>
<tr>
<td>5.2.6</td>
<td>&lt;SSL&gt;</td>
</tr>
<tr>
<td>5.3</td>
<td>MWSL functions</td>
</tr>
<tr>
<td>5.3.1</td>
<td>AddHTTPHeader</td>
</tr>
<tr>
<td>5.3.2</td>
<td>CacheVar</td>
</tr>
<tr>
<td>5.3.3</td>
<td>ExistVariable</td>
</tr>
<tr>
<td>5.3.4</td>
<td>GetVar</td>
</tr>
<tr>
<td>5.3.5</td>
<td>InsertFile</td>
</tr>
<tr>
<td>5.3.6</td>
<td>ProcessXMLData</td>
</tr>
<tr>
<td>5.3.7</td>
<td>SetVar</td>
</tr>
<tr>
<td>5.3.8</td>
<td>ShareRealm</td>
</tr>
<tr>
<td>5.3.9</td>
<td>write</td>
</tr>
<tr>
<td>5.3.10</td>
<td>WriteVar</td>
</tr>
<tr>
<td>5.3.11</td>
<td>WriteXMLData</td>
</tr>
<tr>
<td>5.3.12</td>
<td>NodeIndex</td>
</tr>
<tr>
<td>5.3.13</td>
<td>NodeLevel</td>
</tr>
<tr>
<td>5.4</td>
<td>IT DIAG files</td>
</tr>
<tr>
<td>5.4.1</td>
<td>DIAGURLS.TXT</td>
</tr>
<tr>
<td>5.5</td>
<td>LCID country codes</td>
</tr>
<tr>
<td>5.5.1</td>
<td>LCID table</td>
</tr>
</tbody>
</table>

**Index** | 307
Introduction

1.1 Overview

Function packages

"SIMOTION IT Ethernet-based HMI and diagnostic functions" contains the following function packages (IT = Information Technology):

- SIMOTION IT DIAG
  This function package enables a direct diagnosis of the SIMOTION devices. Access is by means of a standard browser (e.g. Firefox) via the IP address of the SIMOTION device. You can use the standard diagnostic pages or your own HTML pages for access.

- SIMOTION IT OPC XML DA server
  This function package comprises a web service which facilitates the connection of applications to a control via the Internet and access to data and operating modes on SIMOTION devices. Commands are transmitted via the SOAP (Simple Object Access Protocol) communication protocol.

- Trace via SOAP (TVS)
  This function package comprises a web service which enables the tracing (recording) of variables from the SIMOTION variable provider management area.

Function packages in other documents

- SIMOTION VM
  This function package provides a runtime system for executing Java applications on the SIMOTION device. The Jamaica virtual machine (JamaicaVM) used for this purpose is an implementation of the "Java Virtual Machine Specification". The SIMOTION VM is described in a separate Programming Manual "SIMOTION – IT Virtual Machine".
1.2 Schematic diagram of the function packages in the SIMOTION device

Representation of the function packages

The following figure is a schematic diagram of the function packages in the SIMOTION device.

The data of the SIMOTION device can be accessed via the "variable providers".

Figure 1-1 Function packages
Introduction

1.3 Form of delivery

Form of delivery

"SIMOTION IT Ethernet-based HMI and Diagnostic Functions" are included in the control firmware.

Note
The functionality must be activated in the SIMOTION SCOUT project in the hardware configuration of the control. You can activate the "OPC XML/diagnostic pages" function in the hardware configuration on the "Ethernet extended" tab in the object properties of the control.

These settings are preset in V4.1.2 and higher.

Documentation, tools, examples, and configuration files

You can find the documentation, tools, examples, configuration files, and other supplementary features on the "Documentation, Utilities & Applications" DVD.

Runtime licenses before Version 4.2

The older versions require an OPC XML-DA single-user license for access to the Watch page, for example.

When any of these pages is opened, the following is displayed:

![Warning - Missing license](image)

Figure 1-2  Warning - Missing license

If you click the OK button, the requested page is opened. You can thus continue even without a license. However, an entry is made in the diagnostic buffer and the error LED on the control starts to flash.
1.4 Possible applications

1.4.1 Standard information

Application of diagnostic pages

The web pages provided by IT DIAG supply information on a SIMOTION device. The information is accessed via the Web browser and the Ethernet.

The SIMOTION device is connected to the local Ethernet for this purpose. Diagnostic pages can then be accessed from any computer in the network using the IP address of the device. HTTPS connections are also supported, which enables use of IT DIAG via open networks (Internet). See Secure Socket Layer (Page 260).

The use of IT DIAG standard pages does not require a special installation. The device is already appropriately set up.
1.4.2 User-defined information

Displaying information in user-defined pages

In addition to displaying the standard pages, IT DIAG allows you to create your own web pages.

With the aid of a JavaScript library, device data can be queried and displayed in a web page.

A further option is the use of the MiniWeb Server Language (MWSL). A language based on ECMA script, which is executed on the server side.

The "variable providers" can be used to read and write the following information on a web page:

- System variables of the SIMOTION device
- System variables and configuration data of the technology objects
- Global unit variables
- Drive parameters
- IO variables
- Global device variable

User-defined pages provide numerous options for displaying device information.
Introduction

1.4 Possible applications

SIMOTION IT Ethernet-based HMI and Diagnostic Functions

**Figure 1-4** Example of a user-defined SIMOTION IT DIAG web page

**MWSL**

The MWSL is executed on the server side. You should use the MWSL if the created pages are displayed on devices which do not support JavaScript. Variable functions can also be executed faster and more directly (closer to the system) than when using JavaScript.

One of the disadvantages of using the MWSL is the absence of dynamic response, since the pages are always generated statically. Another disadvantage is the load imposed by server evaluation, which can take quite a long time if controls are working at full capacity, thereby holding up other web processes and requests.

**JavaScript**

SIMOTION IT supports you in creating dynamic and flexible web pages thanks to an extensive JavaScript library. Unlike MWSL, the library is executed in the browser. The use of JavaScript relieves the load on the control and provides considerably more options than the MWSL. For display purposes, however, a modern browser with corresponding JavaScript support is required; this is something that cannot be guaranteed in all automation environments.
Commissioning

2.1 Hardware and software requirements

Hardware requirements
- SIMOTION device
- PC or Notebook with Ethernet connection.

Software requirements
- Browser: Firefox Version 3 or higher.

2.2 Configuring the SIMOTION device interface

Configuration of the Ethernet interface
SIMOTION IT DIAG can be accessed via any Ethernet interface used with SIMOTION, including PROFINET IO IRT.

In order to establish a connection between the standard diagnostics pages and a SIMOTION device via a browser, the following steps must be carried out to configure the Ethernet interface:

Table 2-1 Configuring the interface

<table>
<thead>
<tr>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The functionality must be activated in the SIMOTION SCOUT project in the hardware configuration of the CPU. You can activate the &quot;OPC XML/diagnostic pages&quot; function in the hardware configuration on the &quot;Ethernet Extended&quot; tab in the object properties of the CPU. This is the standard setting in V4.1.2 and higher.</td>
</tr>
<tr>
<td>2</td>
<td>It may be necessary to change the USER NAME and PASSWORD in the configuration file WebCfg.xml. The default settings for the USER NAME and PASSWORD are &quot;simotion&quot; and &quot;simotion&quot; respectively. These settings should be changed immediately.</td>
</tr>
<tr>
<td>3</td>
<td>To display the standard diagnostics pages in the browser, you must enter the IP address of the SIMOTION device, e.g. <a href="http://169.254.11.22">http://169.254.11.22</a>. The preset IP addresses are documented in the manuals for the respective controls. This factory setting can be changed in the HW Config and then loaded to the SIMOTION device.</td>
</tr>
</tbody>
</table>
2.3 Log-in administration

Structure of the log-in administration

The log-in system is structured as follows:

- There are Users.
- Each user has a password. The password can be in plain text or MD5 Hash format.
- There are secure areas (SecureGroups and/or Realms).
- Each secure area has a group of users who are authorized for access.
- A user can have access to different secure areas.

The User Database is part of file WebCfg.xml and can only be changed by reloading WebCfg.xml.

---

**Note**

Faulty changes in file WebCfg.xml can cause the device to crash.

---

**Note**

On delivery, the default logon settings for the user name and password on the device are "simotion" and "simotion" respectively.

Please change this access data immediately.
Sample configuration
<SERVERPAGES>
  <BASE LOCALINK="/">
    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                         <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                           <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                             <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                               <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                 <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                   <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                     <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                       <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                         <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                           <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                             <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                               <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                 <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                  <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                   <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                     <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                      <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                       <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                         <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                           <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                             <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                              <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                               <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                    <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                     <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                        <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                         <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                          <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                           <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                            <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                             <FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                                                                                             ...<FILES LOCALINK="/" PREFER_EXTERNAL="TRUE" BROWSEABLE="TRUE">
                                                                             </BASE LOCALINK>
                                                                       </SERVERPAGES>

If the USER is "simotion", the password is specified in plain text ("simotion"). USER "simotion" can only access the "Administrator" secure area.

The "Administrator" group has been permanently programmed for the WebCfg.xml update. Which users belong to this group, is specified in the USERDATABASE.

User1 has been inserted. This user belongs to the new group "Servicegroup" and has access to the "Settings.mcs" page.
Secure areas

Additional secure areas can be adapted to the requirements in question in WebCfgFrame.xml.

- Access to the settings:
  <settings.mcs SECUREGROUP="Administrator"/>

- Access to the "Files" directory:
  <DEFAPP SECUREGROUP="Administrator"/>

- Writing of variables in the HTML diagnostics pages:
  <VarApp SECUREGROUP="Administrator"/>

- Update of the project and firmware:
  <FWUpdtApp SECUREGROUP="Administrator"/>

- Access protection to the OPC XML DA service:
  <URL BASE="OpcXml" SECUREGROUP="Administrator"/>

MD5 hash

MD5 (message Digest Algorithm 5) is a cryptographic hash algorithm which saves a character string requiring protection (in this case the password) in the configuration but not as plain text.

Saving the password in plain text would have the disadvantage that a hacker could read it and use it to gain unauthorized access to the system. Instead, the password is saved as what is known as a hash. The hash is a virtually bijective type fingerprint of the password.

To authenticate a user, the client (in this case the web browser) sends the password to the server, which then generates the hash and the MD5. This hash can be compared with the one saved in the configuration and the system can respond accordingly. This procedure is considered one of the most secure of its type. You can find more information on the Internet, e.g. at http://de.wikipedia.org/wiki/Message-Digest_Algorithm_5.

On the Internet, there are several sites which explain how an MD5 hash can be created for your password. Alternatively, all common programming frameworks which work with Internet technology (e.g. Microsoft .NET or Java) offer corresponding implementations which provide you with a straightforward means of creating a program which will convert the passwords for you.

Integrating the MD5 in WebCfg.xml

Instead of the PASSWORD="..." attribute, the MD5="..." attribute is used. The attribute value is the MD5 hash, which is 32 characters in length and case-insensitive.

Example: <USER NAME="simotion" MD5="5fc8f76e94ad3ab985ad8b4f192dc9ef"/>
A hacker can no longer read the password "simotion" encrypted in the example. Users who know the password can authenticate themselves in the browser.

**Note**
A mixture of MD5 and plain text passwords can be used; if both procedures are configured for a user, the MD5 password is always used.

**Note**
The MD5 passwords cannot be edited with the user editor in IT DIAG.

**See also**
IT DIAG System (Page 79)
2.4 Setting the language for AlarmS and user-defined diagnostics buffer messages

Any of the SIMOTION SCOUT languages can be used when setting the language for AlarmS and user-defined diagnostics buffer messages.

Language localization

IT DIAG uses four rules for language localization; it is always the first rule to apply that is used:

1. Configuration constant ForceUserMsgLanguageID
   An entry for the required language is available in the configuration constants. The ForceUserMsgLanguageID variable is set to the corresponding country code (decimal value) for this purpose. The selected language must exist. If it does not, the THX display is used. More information is available regarding the configuration constants (Page 166) and the country codes (Page 301).

2. SIMOTION SCOUT export
   Performing a SCOUT export of user-defined AlarmS and diagnostics buffer messages and then uploading (Page 82) this data sets the IT DIAG language to the same one which is set in SCOUT.

3. Language of system diagnostics buffer texts
   An attempt is made to find the language that matches the installed system diagnostics buffer texts.

4. Other language settings
   If no matching language is found among the system diagnostics buffer texts, the system's default language is selected instead.
   The language which has been selected is documented in the syslog file.
3.1 IT DIAG overview and general functions

3.1.1 Overview

The SIMOTION device administers prefabricated standard diagnostics pages. These pages can be displayed using a generally available browser via Ethernet. You can also create your own HTML pages and integrate servicing and diagnostics information.

Purpose and benefits

The purpose and benefits of HTML diagnostics pages are as follows:

- Preconfigured diagnostics pages are available to the user for the direct diagnosis of the SIMOTION device.
- Service and diagnostics information of the device can be accessed without manufacturer-specific programs to assist in production monitoring or diagnostics.
- User-defined HTML pages can be integrated.
3.2 Standard pages

3.2.1 Home

SIMOTION device data

The following current data of the SIMOTION device is displayed on the home page:

- **Order Number**: Order number (MLFB) of the device
- **Revision Number**: Hardware version
- **Licence Serial Number**: The license key is tied to this serial number.
- **User Version**: SIMOTION Kernel user version
- **Operating state**: Operating mode of the SIMOTION device (RUN, STOP, STOPU)
- **System time**: Current time-of-day of the SIMOTION device

![Image of SIMOTION device data on home page](image-url)
For more information regarding the current device data, refer to the "Device Info (Page 26)" page.

**General links**

Each IT DIAG page includes three general links:

- "Watch" enables you to access the Watch function (Page 34), which is displayed in a separate window.
- "Overview" shows the Service Overview (Page 31) in a separate window.
- "CopyLink" enables you to copy the URL of the current page to the clipboard.

**CopyLink**

CopyLink displays the current URL of the current page.
### 3.2.2 Device Info

#### 3.2.2.1 Device Info

**Hardware and firmware information**

The following current hardware and firmware information of the SIMOTION device is displayed on the "Device Info" page:

<table>
<thead>
<tr>
<th>Manufacturer Name</th>
<th>Siemens AG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Number</td>
<td>Delivery number (MLFB) of the device</td>
</tr>
<tr>
<td>Revision Number</td>
<td>Hardware version</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Serial number of the SIMOTION device</td>
</tr>
<tr>
<td>User Version Firmware</td>
<td>SIMOTION Kernel user version</td>
</tr>
<tr>
<td>Build Number</td>
<td>Internal version number</td>
</tr>
<tr>
<td>Additional Hardware</td>
<td>Installed components of the SIMOTION device including:</td>
</tr>
<tr>
<td></td>
<td>MLFB, serial number, revision number, firmware name, user version number, internal version number</td>
</tr>
<tr>
<td>Technological Packages</td>
<td>Loaded technology packages including:</td>
</tr>
<tr>
<td></td>
<td>Package name, user version number, internal version number</td>
</tr>
</tbody>
</table>
Figure 3-3  Device Info
3.2.2.2 IP Config

Data of the SIMOTION device Ethernet interface

The following current interface data of the SIMOTION device is displayed on the IP-Config page:

- **IP Address**: Address of the TCP/IP interface
- **Subnet Mask**: Subnet Mask of the interface
- **MAC Address**: Address of the network card
- **Gateway**: Default gateway of the interface

The corresponding information is always displayed in the first column. It is not necessarily directly related to the IP address of the column and may even have been configured for the other interfaces.

**Ethernet-port status**:
Overview of Ethernet ports. The port speed and communication type are output for active ports.

![IP Config Diagram](image)

Figure 3-4 IP Config
3.2.3 Diagnostics

3.2.3.1 Diagnostics

Overview of the general state of the SIMOTION device

The following states of the SIMOTION device are displayed on the Diagnostics page:

- **Systemtime**: Current time-of-day of the SIMOTION device
- **Timezone**: Current difference between the Systemtime and GMT in minutes
- **CPU Load by cyclic Tasks**: Processor time of servo and IPO levels as a percentage of the total processor time
- **Memory Load**: Size and allocation of the memory, RAM disk, memory card, and non-volatile memory in bytes
- **State**: Current operating mode of the SIMOTION device

Select the tabs on the page to access more detailed information.

![Diagnostics Page](image-url)
3.2.3.2 Task runtime

Information on task runtimes and states

On the Task runtime page (opened via Diagnostics > Task runtime), you can view the following information:

- **Taskname**: Name of the task
- **Status**: Current status of the task
- **Actual**: Current runtime of the task in ms
- **Min**: Minimum runtime of the task in ms
- **Max**: Maximum runtime of the task in ms
- **Average**: Average runtime of the task in ms

![Task runtime screenshot](image-url)
### Service overview

SIMOTION SCOUT provides an overview screen that displays the state of the axes available in the project. The web server provides a corresponding page.

![Service overview screenshot](image)

**Figure 3-7 Service overview**

The columns in the table represent each of the axes. Clicking the **Axis** button reveals a selection of all the available axes, allowing you to choose the ones you require.

You can use the **Save** button to save the current setting in the device. A name for this must be entered in the input field to the left of the **Save** button.

You can use the **Load** button to load a setting and delete it using the **Delete** button.
The **Extended...** button opens a window in which the required system variables can be selected.

![Figure 3-8 Selection of variables](image1)

The **Axis...** button opens up a selection of axes.

![Figure 3-9 Selecting the axes](image2)
More Options

The More Options button expands the upper area of the screen to include options for saving or loading the configuration of a Service overview on a PC.

Click the Send selected file button to load a configuration received previously by E-mail, for example, from the PC to the device.

Using Select all and De-select all, all of the configurations displayed can be selected or deselected.

Click the Get selected button to save the selected configurations onto the PC as XML files.
3.2.3.4 Watch

Watch table

This page combines a variable browser and a watch table. The variables are entered in the watch table with the aid of the browser.

![Watch Table](image)

For monitoring variables, the web server provides a watch table and a symbol browser. The symbol browser provides the option of browsing the entire variable management area of a SIMOTION control. The watch table and the symbol browser are displayed in a tree topology on the left-hand side. The selected variables are displayed on their right and can be edited for the watch function.

Only users who have logged on can access this page.

See Log-in administration (Page 18).

In order to monitor unit variables, the "Permit OPC-XML" option must have been activated in the compiler settings for the associated unit. See Making unit variables available (Page 210).

The format column allows you to change the display format in the case of whole-number variables.

- DEC for decimal display (default).
- HEX for hexadecimal display.
- BIN for binary display.

All control values are interpreted according to this setting.
Accessing the drive parameters

The drive parameters are accessed via a tree topology. The parameters are selected using the same method as when accessing variables via the variable provider. See Variables providers (Page 235).

Parameters are displayed as a number without a preceding 'p' or 'r'. For example, parameter r0002 becomes 0002.

There are three options for accessing drive parameters:

1. Axis technology object

   Selecting a technology object

2. Drive object addressing

   Selecting a drive object (the name is generated from the diagnostics address)
Selecting a drive

![Image of a Simotion drive screen](image)

Figure 3-12 Selecting a DO parameter

3. Logical address

![Image of a Simotion drive screen](image)

Figure 3-13 Selecting a logical address

Selecting a drive parameter and a logical address

Editing the watch table

The table can be saved using the **Save** button. To do this, you need to enter a name in the input field next to the button.

You can use the **Load** button to load a saved table and delete it using the **Delete** button.

The Delete button (represented by a red cross) in the title row of the table can be used to delete all the table rows at once. Individual table rows can be deleted using the corresponding button at the end of the relevant row.

Clicking the **More Options** button expands the upper area of the screen to include options for saving the watch table settings on a PC and subsequently reloading them to the control.
You can find a more detailed description of the **More Options** functionality in the Service Overview section.

**See also**

Service overview (Page 31)
3.2.3.5 Trace (device trace)

Setting up a Device Trace

The SIMOTION control provides the user with the option of setting up a variable trace via a web service.

Version 4.2 and higher provides not only the device trace described here but also a distributed trace (Page 42) (System Trace).

Figure 3-15 Device Trace
Procedure for creating and executing a Device Trace:

- Select the **Device Trace** radio button
- Select the required signal from the provider list (to, unit, or var)
- Click the **Set** button to set the selected symbol as the required signal
- Set the recording and trigger conditions
- **Download** – Load the settings to the control
- **Start** – Starts the trace
- **Stop** – Stops the trace (only required for a manual trace)
- **Read** – Load the trace results to the PC in the form of a WTRC file. The WTRC file is then deleted on the device.
- View the WTRC file using the WebTraceViewer
- **Cancel** – Deletes the settings from the control
- **Reset** – Deletes the settings from the web page

The **Read** button is used to generate a file with the extension WTRC, which contains the up-to-date trace data. The file can be saved or viewed with the WebTraceViewer program.

Clicking the **More Options** button expands the upper area of the screen to include options for saving the device trace settings on a PC and subsequently reloading them to the control.

Only users who have logged on can access this page. See Log-in administration (Page 18).

**Note**

Only a limited amount of memory (256 KB), arranged as a ring buffer, is available for the Trace.

**Trace modes**

The device trace can be run in two modes:

1. **Triggered**
   
   The trace starts when a trigger event occurs and stops when a parameterizable time expires or when the trace buffer is full.

2. **Endless**
   
   The trace starts immediately and runs until it is stopped by the operator. The trace data is collected from a client on a continuous basis.

You can find a description of the recording settings and trigger conditions in the System Trace section.

**Saving and loading a trace configuration**

You can save a configuration under a name on the device by clicking the **Save** button and load it again using the **Load** button. You can find a more detailed description of the **More Options** functionality in the Service Overview (Page 31) section.
WebTraceViewer

The WebTraceViewer PC program enables the trace data to be displayed. The GetWebTraceViewer link can be used to save the WebTraceViewer on the PC. This link is not available with C modules. Alternatively, the WebTraceViewer can be obtained from the E-Support department or copied from the add-on DVD.

This program is able to graphically display the data saved in a WTRC file.

![WebTraceViewer](image)

Figure 3-16 WebTraceViewer

**Button functions**

1. Open file: Enables you to open WTRC files.
2. Save file: Enables you to save WTRC files.
3. Copy: Copies the content of the current WTRC window to the clipboard in bitmap format. This enables the graphic to be copied to a word processing program, for example.
4. Scroll mode: Enables you to shift the visible area of the graphic using the mouse.
5. Zoom mode: Enables you to expand and compress the graphic using the mouse.
6. Selection mode: If this button is activated, only a rectangular area of the graphic can be selected. Buttons 4 and 5 can then no longer be used.
CSV export

The **File Export** menu command allows you to save the trace data in CSV format so you can import it into a spreadsheet, for example.

Defective WTRC files

If the WebTraceViewer imports a defective file, it provides information about the error.

Figure 3-17  WebTraceViewer with faulty WTRC file
3.2.3.6 Trace (system trace)

Setting up and executing a System Trace

The system trace is available as of SIMOTION Version 4.2. The system trace can be used to record a trace involving multiple devices.

![System trace interface](Figure 3-18 System trace 1)
Procedure for creating and executing a System Trace:

- Select the **System Trace** radio button
- Select the required signal from the provider device list (to, unit, or var)
- Click the **Set** button to set the selected symbol as the required signal
- Set the recording and trigger conditions
- **Download** – Load the settings to the control
- **Start** – Start the system trace
- **Stop** – Stop the system trace (only necessary for manual trace)
- **Read** – Loads the trace results to the PC in the form of a WTRC file. The WTRC file is then deleted on the device.
- View the WTRC file using the WebTraceViewer PC program
- **Cancel** – Deletes the settings from the control
- **Reset** – Deletes the settings from the web page
Prerequisites

The devices must be connected and synchronized via PROFINET IO IRT for time synchronization of the distributed trace to function correctly. The trace can be used on as many devices as you wish, but the maximum number of signals is limited to 128, of which no more than 8 can be used for each individual device. Just one trigger is possible for each device.

Once the signals have been selected, the desired recording and trigger conditions must then be assigned.

Trace modes

The system trace can only be run in ‘triggered’ mode. The trace starts when a trigger event occurs and stops when a parameterizable time expires or when the trace buffer is full.

Recording settings

Figure 3-20  Example: Recording settings

![Recording settings screenshot]

Figure 3-21  Example: Basic cycle clock setting

- Condition: Measured value acquisition
- Cyc. Clock: Basic clock cycle
- Recording in ring buffer with max. size of 256 KB
Trigger conditions

<table>
<thead>
<tr>
<th>Designation</th>
<th>Description</th>
<th>Operand 1</th>
<th>Operand 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Edge</td>
<td>Rising edge triggered if variable increases</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Negative Edge</td>
<td>Falling edge triggered if variable decreases</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Within a tolerance band</td>
<td>Within a value range triggered if variable is within the specified interval</td>
<td>Lower limit of the interval</td>
<td>Upper limit of the interval</td>
</tr>
<tr>
<td>Outside a tolerance band</td>
<td>Outside a value range triggered if variable is outside the specified interval</td>
<td>Lower limit of the interval</td>
<td>Upper limit of the interval</td>
</tr>
<tr>
<td>Bit pattern</td>
<td>Bit pattern triggered if variable rounded with Operand1 is not equal to zero</td>
<td>Bit pattern</td>
<td>-</td>
</tr>
</tbody>
</table>

Overview of trigger conditions

**Trigger**
- Rising edge, falling edge
- Pretrigger = Time in ms, when the trigger is activated, this "run-in" is included in the recording

**Initialization**
The trace variables and trigger conditions are transferred to the devices concerned in order to initialize the trace. If the initialization has been completed without errors on at least one device, the trace can start.

**Viewing the trace**
The trace data can be displayed on the PC using the WebTraceViewer PC program.

**See also**
Trace (device trace) (Page 38)
3.2.3.7 Tasktrace

This page enables you to set up and control the SIMOTION Tasktrace (including trigger conditions).

Figure 3-23 Tasktrace upper area
The task trace provides a diagnostics option during runtime which can be used to obtain reliable information about the processes in the individual tasks (e.g. task change).

The trace recording is continuously written to a ring buffer.

Once underway, a trace recording can be stopped manually or held conditionally by a trigger event. The recording can then be loaded to the PC and displayed with the Task Profiler by pressing the Get Trace File button.

**Start Trace**

The Start Trace button starts the task trace with the settings that have been made previously and have been transferred to the device using Submit.

**Stop Trace**

You can stop the trace manually using the Stop Trace button.

The state of the trace is displayed in the field Tasktrace - Current State.
3.2 Standard pages

Start Writeout

The **Start Writeout** button writes the content of the trace buffer to the file 
"/USER/SIMOTION/SYSLOG/TASKTRACE/TTRACE.JEN" on the device.

The state of the write process is displayed in the fields **Writeout - Current State:** and **Writeout - Result:**.

Get Trace File

Click the **Get Trace File** button to load the TTrace.jen file to the PC and use the TaskProfiler program to display it. The setup of the TaskProfiler can be found on the installation DVD in the add-on directory.

Java Runtime Version 1.6 or higher is required to use the program.

Trigger Events

You can select and combine the **Trigger Events** as you wish using various checkboxes. The **submit trigger events** button transfers the selection to the device.

Trigger Mask

The **Trigger Mask** input field enables the expert to input **Trigger Events** as coded numbers. The **submit trigger mask** button transfers the input to the device and overwrites all previous inputs.

Level Settings / Level Mask

You can use these settings to determine which events are entered in the task trace.

Additional Trigger Settings

These settings enable you to back up a trace automatically.

- **Enable automatic writeout after stop**: The trace data is automatically backed up after the occurrence of a trigger event.
- **Enable automatic restart after writeout**: The trace is restarted after backing up the trace data.

Click **Trigger Delay** to set the time during which the trace remains active after a trigger condition occurs.
Current Tasktrace Settings
You can back up, load or delete a setting here.

Saving the trace settings
The current trace settings can be saved in XML file
"/user/simotion/hmi/files/persist/ttrace.xml" on the storage medium of the control. This file is
evaluated during power-up. As a result, it is also possible to activate the trace of system
function calls from the web interface. In addition, the web server allows you to delete this file.

3.2.3.8 Diagnostic files

Backing up diagnostic pages of the web server
You can use this page to back up general diagnostic data and individual IT DIAG HTML
pages.

The standard HTML pages of the web server contain valuable information for analyzing
problems that can occur during operation of the SIMOTION control.
3.2 Standard pages

Create general diagnostic files
This function will create several diagnostic information files and save them to memory card of the SIMOTION device under folder SYSLOG/DIAG.

Create general diagfiles

HTML - diagnostic files
This function will save some of the present HTML-files containing diagnostic information to memory card of the SIMOTION device under folder SYSLOG/DIAG.

To customize the list of files, please save file "DAUERLOG.TXT" in the same folder.

Create html diagfiles

Get diagnostic files
After pressing button "Zip all diagfiles", all diagnostic information files (general and HTML) which are present on the memory card of the SIMOTION device will be zipped into a file called "DIAGARCHIVE.ZIP".

Download the file by pressing button "Get diagarchive".

Zip all diagfiles
Get diagarchive

Delete diagnostic files
This function will delete all diagnostic information files (general and HTML) which are present on the memory card of the SIMOTION device under folder SYSLOG/DIAG.

Delete all diagfiles

Figure 3-25  Diagnostic files

Create general diagnostic files
This function saves diagnostic data for Support.
### 3.2 Standard pages

#### SIMOTION device

<table>
<thead>
<tr>
<th>SIMOTION device</th>
<th>Storage medium</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>D, C</td>
<td>CF Card/MMC</td>
<td>D:\USER\SIMOTION\HMI\SYSLOG\DIAG</td>
</tr>
</tbody>
</table>
| P               | Hard disk      | F:\Simotion\user\Card\USER\SIMOTION\\HMI\\SYSLOG\\       
|                 |                | DIAG                                      |
| P320            | CF card        | D:\Card\\USER\\SIMOTION\\HMI\\SYSLOG\\DIAG            |

The use of this function corresponds, for example, to actuating the service selector switch on the SIMOTION D control. HTML files used for diagnostics purposes are not saved.

**HTML - diagnostic files**

A selection of relevant diagnostic pages are backed up on the data carrier as HTML pages. You can use the DIAGURLS.TXT (Page 300) file to control which HTML pages are backed up.

**Zip all diagfiles**

This function zips the files that were previously created by clicking the "Create general diagnostic files" and "HTML diagnostic files" buttons. The zip file will be empty if no files were created prior to this.

**Get diagnostic files**

Enables you to download the ZIP file generated with the **Zip all diagfiles** button.

**Delete all diagfiles**

Deletes all diagnostic files present in directory ...\\USER\\SIMOTION\\HMI\\SYSLOG\\DIAG. The directory itself is retained.
3.2.4 Messages&Logs

3.2.4.1 Diag buffer

Diagnostics buffer information
On the "Diag Buffer" page (opened via Messages&Logs > Diag buffer) you can view the latest content of the diagnostics buffer of the control.

<table>
<thead>
<tr>
<th>Time</th>
<th>Time of the event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date of the event</td>
</tr>
</tbody>
</table>
| Event| Displays the event as text.

If the DGBUFTXT.EDB language file is missing, it will be displayed in hexadecimal notation.

Note
The text is displayed in English by default. To display the event text in a different language, you must transfer the DGBUFTXT-XX.EDB and DGEXTXT.EDB files in their relevant language versions to the .../USER/SIMOTION/HMICFG directory on the SIMOTION control memory card. See DiagBuffer group (Page 250).
Figure 3-26  Diag buffer
3.2.4.2 Diag buffer drive

Representation of the drive diagnostics buffer

Just as there is a SIMOTION diagnostics buffer, there is also a diagnostics buffer for the integrated drives.

- **Time**: Time of the event
- **Date**: Date of the event
- **Event**: Displays the event as text.

Figure 3-27 Display of the diagnostics buffer for the integrated drives

The diagnostics buffer for a CX32/CX32-2 controller extension can also be viewed in this way.
3.2.4.3 Alarms

Information about alarms

The alarm and alarmS/SQ messages of the device are displayed on the Alarms page.

<table>
<thead>
<tr>
<th>Table 3-1 Technological Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>TO</td>
</tr>
<tr>
<td>Nr</td>
</tr>
<tr>
<td>Text</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3-2 Process Alarms (AlarmS/SQ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmNo</td>
</tr>
<tr>
<td>State</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Text</td>
</tr>
<tr>
<td>More Info</td>
</tr>
</tbody>
</table>
The Quit All button allows you to close all alarms requiring acknowledgment.

Language setting for alarm texts

Alarm texts are displayed in English by default. To display the alarm texts in a different language, you must transfer the TOALARM.ADB file in the relevant language to the SIMOTION control memory card.

Only one language can be saved in SIMOTION at a time.

Procedure

1. Open the AddOn\4_Accessories\4_Alarm_Messages\Simotion_IT\V4.2 directory on the SIMOTION SCOUT Add-Ons DVD. For the language you can choose between ger (German) and eng (English), ita (Italian), fra (French). You will find the TOALARM.ADB file in the corresponding directory.

2. Insert the SIMOTION memory card in a reader/writer.

3. Copy the TOALARM.ADB file to the \USER\SIMOTION\HMICFG directory. You must create the directory if it does not already exist.

4. Insert the memory card in the SIMOTION device again.
Procedure for P350

1. Shut down the SIMOTION P.
2. Open the \AddOn\4_Accessories\4_Alarm_Messages\Simotion_IT\V4.2\ directory on the SIMOTION SCOUT Add-Ons DVD. For the language you can choose between ger (German) and eng (English), ita (Italian), fra (French). You will find the TOALARM.ADB file in the corresponding directory.
3. Copy the TOALARM.ADB file to the F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG directory (for the default installation).
4. Start the SIMOTION P.
3.2.4.4 Alarms drive

Drive faults and warnings

Similar to the technological alarms of the control, a page containing fault and warning messages of the drive is also available. Because alarm texts for drive alarms are currently not available, the display is at this stage only in numerical format.

The following are displayed:

- Time
- Fault time
- Type
- Error type
- Source
- DO name
- No.
- Fault code
- Value
- Fault value

If DOs (Drive Objects) are present in the device by name, they are also output by name.

The representation is in HEX (no alarm texts are output).

Figure 3-29 DriveAlarms
The drive alarms for the controller extension CX32/CX32-2 can also be displayed.

3.2.4.5 Alarm buffer

Contents of the alarm buffer

On the Alarm buffer page, you can view the following information:

<table>
<thead>
<tr>
<th>Index</th>
<th>Time</th>
<th>TO</th>
<th>Alarm</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>02.01.92</td>
<td>Achs1_1</td>
<td>40002</td>
<td>Missing enable(1) (parameter 1: ?) and/or incorrect mode (parameter 2: ?)</td>
</tr>
<tr>
<td>01</td>
<td>01.03.25 06:44</td>
<td>Achs1</td>
<td>30002</td>
<td>Command aborted (reason: ?, command type: ?)</td>
</tr>
<tr>
<td>02</td>
<td>01.02.19</td>
<td>Achs1</td>
<td>40005</td>
<td>Missing enable(1) (parameter 1: ?) and/or incorrect mode (parameter 2: ?)</td>
</tr>
<tr>
<td>03</td>
<td>01.02.19</td>
<td>Achs1</td>
<td>30002</td>
<td>Command aborted (reason: 5, command type: 701)</td>
</tr>
</tbody>
</table>

In contrast to the Alarms page, which shows the alarms that are currently pending, the Alarm buffer page shows a history of all the alarms.
3.2.4.6  Syslog

The Syslog page displays the syslog file for the relevant device.

![Syslog page from Siemens SIMOTION D435]

This file is maintained by the system. Events that are important for diagnostic purposes are documented, such as RAM2ROM.
3.2.4.7 Userlog

Userlog

Figure 3-32 Userlog

The Userlog shows free texts entered by users in SIMOTION SCOUT (Device Diagnostics > Userlog). The texts are saved in a file on the memory medium of the control and displayed on the web page (in read-only format).
3.2.5 Machine overview

3.2.5.1 Module information

Overview of configured modules

![Module information](image)

Overview of all modules configured on the machinery. Starting with the segment level, you can drill down to access information at the level of specific elements.

**Note**

For a correct representation of the information contained in the pages of the **Machine Overview** it is necessary to load an HW Config in IT DIAG. This must match the loaded SCOUT project, otherwise incorrect information is displayed. See Configuration (Page 67)
The hierarchy is always as follows: Segment > Device > Slot > Subslot (if present). Elements without subelements are not clickable.

Clicking on the segment displays all of the devices in the segment (real shot).

Clicking on **Details** displays further information at the bottom (info2 shot).

Clicking on **back** takes you one step back.
3.2.5.2 Topology

Overview of the configured topology

Figure 3-35 Topology of the device

The configured topology of a device is depicted on this page. Inaccessible nodes are highlighted in red.

The topology display shows how the nodes must be wired.
3.2.5.3 Topology table

Tabular overview of the configured topology

<table>
<thead>
<tr>
<th>Port Status</th>
<th>Name</th>
<th>Module Type</th>
<th>Port</th>
<th>Partner Port</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MyD435</td>
<td></td>
<td>port 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>port 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>port 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Rx)</td>
<td>IM153-4PN</td>
<td></td>
<td>port 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0120xU24/0D</td>
<td></td>
<td>port 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MyD435</td>
<td></td>
<td>port 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-36 Tabular topology table

This page offers a quick overview of the wiring in text form.
The information displayed corresponds to that of the topology (Page 64) page.
3.2.5.4 Overview

Overview of all modules configured on the network

This overview displays all modules configured on the network without topology information. This overview is primarily intended for very large projects. Inaccessible or failed nodes are shown in red.
3.2.5.5 Configuration

Loading HW Config information in IT DIAG

An HW Config export file must be loaded in IT DIAG. The texts and designations of the installed modules are only present once this has been done. The HW Config export file and the loaded SCOUT project must match, otherwise incorrect information is displayed.

Exporting in HW Config

- Open HW Config
- Menu StationExport
- Save the file.
- The control must be in STOP mode.
- Load the resulting file using the form on the IT DIAG page.
- The SIMOTION control subsequently performs a restart.

The file can then be found on the card in the directory /USER/SIMOTION/HMICFG/HWCONFIG.CFG.

Alternatively, you can also directly copy the file to the card using a card reader.
3.2.6 Manage config

3.2.6.1 Device update

Device update of the device

This page enables a device update to be loaded, and selected data to be saved to the PC from the device.

If several update archives have been written to the control one after the other, you have the option of restoring a previous configuration.

Figure 3-39 Manage Config
• **Get selected data** transfers the currently active device data to the PC. The backed-up data is in a form that allows it to be reimported to the device. 
  FW (firmware), TP (technology packages), **Project** (current project), **Scout Archive** (including the Scout backup), **IT DIAG** (IT DIAG configuration), **UDS** (including the Unit Data Sets)

• **Send new update data** transfers a file generated with the Devices Update Tool to the device. This process can take several minutes and restarts the device.

• **Restore last update** reactivates the last version of the device data of the preceding software update.

For more information regarding this topic, refer to the "Updating SIMOTION Devices" operating instructions.

---

**Note**

The SIMOTION P control does not support firmware download.

---

<table>
<thead>
<tr>
<th><strong>DANGER</strong></th>
</tr>
</thead>
</table>
| To send or download a project or firmware, the control must be switched to STOP mode.  
Type and contents of the file are not checked during transmission.  
If an invalid configuration is used, the USER directory must be deleted from the memory card. |

---

**Note**

If low-capacity cards (32 MB/64 MB) are used, problems may be encountered during the update due to insufficient memory space.

The amount of memory space required is determined by the size of the existing configuration plus that of the update.

Depending on the file involved, the SIMOTION control automatically executes the following actions when the "Send update data" button is clicked:

- **WebCfg.xml**
  
  Restart of the web server.  
  **Note:** All OPC XML DA subscriptions are lost.

- **MyProject.ZIP**
  
  Saving of the new project together with the Ethernet configuration on the (virtual) memory card and activation of the new project with a SIMOTION control restart.

- **XXXXXXXFW.ZIP**
  
  Saving of the firmware on the memory card and activation of the new firmware with a SIMOTION control restart.
Only users who have logged on can access this page. See Log-in administration (Page 18).

Use of older configuration data

Older configuration data that was created with the SIMOTION SCOUT function **Load to File System** can continue to be imported using IT DIAG.

The ZIP file generated by SCOUT as part of this process can be transferred to the device using **Send update data**.

### 3.2.6.2 Upgrading firmware prior to V4.2

A firmware update involving versions lower than Version 4.2 can result in the following: An old WebCfg.xml is retained on the device and causes empty diagnostic pages to be displayed.

**Option for avoiding this problem:**

- Explicit deletion of WebCfg.xml.

After the next reset, a new WebCfg.xml is generated by the device. The old WebCfg.xml should be backed up first so that settings can be transferred from the old configuration to the new WebCfg.xml.

**See also**

Device update (Page 68)
3.2.6.3 Editing function

Editing functions of the IT DIAG pages

The WebCfg.xml configuration file can be edited on some standard pages via the browser. The editing functions are always structured in the same way and are explained in this section.

The add row button inserts one line.

To edit the line, you first need to click the EDIT button in the corresponding line; the input fields can then be completed.
### 3.2 Standard pages

After editing, the changes must be saved in IT DIAG by clicking the **SAVE** button. By clicking the **SAVE** button, changes are saved only in the browser and are not yet transferred to the device. The **submit settings** button must be used to definitively transfer the changes to the device.

The **DELETE** button deletes the inputs in the relevant line of the page but not on the device. All the information on the page can be edited, saved or deleted at once using **edit all, save all**, and **delete all**.
To transfer the modified data onto the device, you need to click the submit settings button. This process is confirmed by a corresponding message and the web server of the device with the modified WebCfg.xml is restarted.
3.2 Standard pages

3.2.6.4 IT DIAG tab

Web pages for making changes to the configuration

The IT DIAG tab summarizes the web pages which are used for IT DIAG configuration. All settings lead to changes in the WebCfg.xml file. As an alternative to editing using the web pages, changes can be made directly in this XML file.

3.2.6.5 IT DIAG Base

Setting the LOCALLINKS

![Image of the IT DIAG Base tab](image)

The Base tab is used to edit the LOCALLINKS in the `<BASE>` tag of WebCfg.xml. The physical file system can be accessed via LOCALLINKS (Page 108).

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILENAME</td>
<td>Character string</td>
<td>Index.mbs</td>
</tr>
<tr>
<td>REALM</td>
<td>Character string</td>
<td>A group name: Administrator</td>
</tr>
<tr>
<td>LOCALLINK</td>
<td>Character string</td>
<td>LOCALLINK=&quot;$WWWROOT/NewDir/index.mbs&quot;</td>
</tr>
<tr>
<td>PREFER_EXTERNAL</td>
<td>TRUE/FALSE</td>
<td></td>
</tr>
<tr>
<td>BROWSEABLE</td>
<td>TRUE/FALSE</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>Character string</td>
<td>One or more group names: Administrator, Servicegroup</td>
</tr>
</tbody>
</table>
### Attributes

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE</td>
<td>Character string</td>
<td>One or more group names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Administrator, Servicegroup</td>
</tr>
<tr>
<td>MODIFY</td>
<td>Character string</td>
<td>One or more group names:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Anyone</td>
</tr>
</tbody>
</table>

Attribute overview Base tab

### See also

LOCALLINK attribute (Page 275)
3.2.6.6 IT DIAG Server options

Basic settings

Figure 3-44 IT DIAG Server options

This tab enables you to set basic parameters for the web server.

This page is used to make various settings for the `<SERVEROPTIONS>` tag in WebCfg.xml.

- **DEFAULTDOCUMENT** (Page 270) enables you to change the home page. The default setting is INDEX.MCS.
- **PORTNUMBER** (Page 272) defines the TCP/IP port for outputting the web server pages. The default setting is port 80 (http).
- **SSLPORTNUMBER** (Page 273) defines the TCP/IP port for outputting the web server pages in encrypted format. The default setting is port 443 (https).

**Non-changeable information**

- **BROWSEABLE** (Page 269) shows the directory display setting.
- **LANGUAGE** shows the language setting.
3.2.6.7 IT DIAG Mimetypes

MIME types

A MIME type can be linked to a file extension on this tab.

The MIME type is used to signal to the browser, by means of the HTTP header, what type of data is being transferred.

The setting in the Filter column can be used to specify whether relevant files are processed with the server filters MWSL and SSI and the tags they contain are evaluated accordingly before they are sent to the browser.

See also

<MIME_TYPES> (Page 271)
3.2.6.8  IT DIAG configuration data

Configuration of user-defined constants

<table>
<thead>
<tr>
<th>USERCONFIG</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User Constant</td>
<td>Current Value</td>
<td>edit all</td>
<td>save all</td>
<td>delete all</td>
</tr>
<tr>
<td>User Area</td>
<td>Embedded Simple</td>
<td>EDIT</td>
<td>SAVE</td>
<td>DELETE</td>
</tr>
<tr>
<td>User Dir</td>
<td></td>
<td>EDII</td>
<td>SAVE</td>
<td>DELETE</td>
</tr>
<tr>
<td>Include SuperDir</td>
<td>NO</td>
<td>EDIT</td>
<td>SAVE</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

Send the settings made above to SIMOTION device. Afterwards the Webserver will be restarted to create an IT Diag configuration, considering the new file.

Figure 3-46  IT DIAG Configuration data

This page enables you to create and edit configuration constants.

See also

Configuration constants (Page 166)
3.2.6.9 IT DIAG System

User database

The System page enables you to manage users. Passwords, group rights, and access rights can be assigned to users here.

Send the settings made above to SIMOTION device. Afterwards the Webserver will be restarted to create an IT Diag configuration, considering the new file.
3.2 Standard pages

Figure 3-47   User database

The **add user** button creates a new user. **EDIT** enables you to edit and **SAVE** enables you to save the user details.

The **add GROUP** button can be used to assign groups to a user. The **delete USER ’<user name>’** button deletes the corresponding user.

See also

- Log-in administration (Page 18)
- `<USERDATABASE>` (Page 274)
3.2.6.10 IT DIAG WebCfg transmission

Transmitting configurations to the device

![WebCfg transmission page](image)

The user data can be sent to or received by the device via this page. The page also enables you to transmit a WebCfg.xml which has been edited at a local level to the device.
3.2.6.11 IT DIAG text databases

Transfer of user-defined messages from SIMOTION SCOUT to the device

On this page, IT DIAG provides an option to transfer user-defined AlarmS and DiagBuffer messages, which have previously been exported in SIMOTION SCOUT, to the device.

For AlarmS, select the IAlarm_S_Navigate.xml file, and for DiagBuffer, select the IUserMsg_Navigate.xml file of a SIMOTION SCOUT language export. It is possible to select different languages for AlarmS and DiagBuffer messages.

Once the files have been transferred to the device, the messages exist in two files
- dgusralarm.edb
- dgusrtxt.edb

in the /user/simotion/HMICFG directory. These files can be transferred to other controls.
Language export from SIMOTION SCOUT

In SIMOTION SCOUT, the **Project messages** menu command enables you to export user-defined messages.

![SIMOTION SCOUT language export language selection](image)

**Figure 3-50** SIMOTION SCOUT language export language selection

![SIMOTION SCOUT language export, specification of the target directory](image)

**Figure 3-51** SIMOTION SCOUT language export, specification of the target directory

During the export, all user-defined texts in all available languages are exported in XML files. During the upload to the device, only the language preselected in SIMOTION SCOUT is saved.

Every change made in SIMOTION SCOUT requires the texts to be exported and uploaded again.
3.2 Standard pages

3.2.7 Settings

This page allows you to change various settings.

Settings for the SIMOTION device can be changed in the Operation state and Time Settings areas.

In the User Pages area, you can change how user-defined pages and the IT DIAG menu editor appear.

Note

The Settings page is password-protected.

On delivery, the default settings are User = simotion and password = simotion.

You should change this password in order to avoid security problems.

See Log-in administration (Page 18).

Figure 3-52 Settings
Changing the state of the SIMOTION device

Operation state

In the field for the operating mode of the SIMOTION device, the request to change the operating state can be triggered by pressing the RUN, STOPU, or STOP button as appropriate.

The switch on the control has a higher priority than this input, i.e. if this switch is set to STOP, then RUN is not possible.

**Note**: For the purpose of transferring a project or firmware, the current operating mode must be set to STOP.

<table>
<thead>
<tr>
<th>DANGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the operating mode is not switched under controlled conditions, this may endanger the safety of personnel and the machine.</td>
</tr>
<tr>
<td>Note the safety regulations before using these functions to change the operating mode of a SIMOTION device.</td>
</tr>
</tbody>
</table>

Time Settings

The system time and the time zone for the SIMOTION device are set in minutes, including sign, in the field for the time settings.

- **Systemtime**: Local time-of-day of the SIMOTION device
- **Timezone**: Difference between the Systemtime on site (i.e. local time) and GMT

The system time and the time zone are relevant for the OPC XML DA access.

The OPC XML DA client expects all times sent by the SIMOTION device to be in GMT. However, a SIMOTION device is set to local time (GMT + X); therefore, a time zone must be set for the SIMOTION device.

The **Change Timezone** button opens a list of time zones, from which one time zone can be selected.

For browsers which do not support the list display, the difference must be entered in minutes, with sign, in the range -720 to +780.

The time zone can also be set under **Hardware configuration > Object properties of the CPU > "Ethernet Extended" > OPC XML / diagnostic pages** and then applied by running a download.
3.2 Standard pages

User Pages

The Enable user menu editor checkbox enables you to activate the menu editor link on the user-defined pages. This option will only take effect once Embedded has been selected from the User Pages drop-down box.

The User Pages drop-down box affects how the user-defined pages are displayed. See Embedded, user-defined pages (Page 122).
3.2.8 Files

3.2.8.1 Files

You can create, select, and delete subdirectories on the memory card in the SIMOTION device via the Files page. Furthermore, you can save, display, and delete files.

![Files page screenshot](image)

**File and directory management**

The user-specific directories and files are stored in a separate directory. With a default installation, the directory settings are:

<table>
<thead>
<tr>
<th>SIMOTION device</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, D</td>
<td>\USER\SIMOTION\HMI\FILES</td>
</tr>
<tr>
<td>P350</td>
<td>F:\SIMOTION\USER\CARD\USER\SIMOTION\HMI\FILES</td>
</tr>
<tr>
<td>P320</td>
<td>D:\Card\USER\SIMOTION\HMI\FILES</td>
</tr>
</tbody>
</table>
To create subdirectories, enter the desired name in the input field and then confirm by clicking the Create Directory button.

You can delete files and directories using the Bin icon. You must make sure that a directory is empty before deleting it. If the directory contains files, these will have to be deleted first.

---

**Note**

You can check the amount of memory available on the card on the diagnostic page in the "Memory Card" line (Diagnostics (Page 29)).

---

**Copying files to the SIMOTION control**

The **Send selected file** button enables you to transfer a file from the local file system to the SIMOTION control. You can use the button displaying the folder symbol to select a file from your local file system and click the **Send selected file** button to transfer it to the SIMOTION control.

---

**Note**

If you upload a file with the same name as one already saved in the SIMOTION control, the existing file will be overwritten.
3.2.8.2  Proc

Accessing the device variables using the Proc file system

![Image of Proc file system]

The Proc file system shows the device variables as a drive in the browser. This enables device variables to be read out via FTP, for example.

Variables are accessed via a path specification and the addition of the extension "bin" to the name of the variable.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO configuration data</td>
<td>/cfg/&lt;toname&gt;/&lt;varname&gt;.bin</td>
</tr>
<tr>
<td>TO system variables</td>
<td>/to/&lt;toname&gt;/&lt;varname&gt;.bin</td>
</tr>
<tr>
<td>Device system variables</td>
<td>/var/&lt;varname&gt;.bin</td>
</tr>
<tr>
<td>Program variables</td>
<td>/unit/&lt;unitname&gt;/&lt;varname&gt;.bin</td>
</tr>
</tbody>
</table>

Arrays are also accessed via a path.
Variable: unit/UnitName.StructName.StructCompSimple
Path: /unit/UnitName/StructName/StructCompSimple.bin

Access to arrays and structures
Variable: unit/UnitName.Array[5].StructName.StructCompSimple
Path: /unit/UnitName/Array/5/StructName/StructCompSimple.bin

The files in the ProcFS comprise the contents of variables in binary format, in the display (endianness) of the control used.
3.3 Simplified standard pages

3.3.1 BASIC pages

Showing IT DIAG pages on devices with small displays

Special pages are provided in Version 4.1.3 and higher in order to achieve the best possible display of IT DIAG pages on devices such as cell phones or PDAs.

The following minimum configuration is recommended for the display of the basic IT DIAG pages:

- Mobile operating system with installed web browser, which supports the HTML 4 standard
- Minimum screen resolution of 320 x 240 pixels and color display
- Touch screen or stylus-operated device
- JavaScript (ECMA-262) is required if the full scope of functions is to be enjoyed.

You can access these pages via the address http://<IPAddr>/BASIC.

Figure 3-55 Start screen for simplified HTML pages
3.3 Simplified standard pages

3.3.2 Device Info

Hardware and firmware information

The following up-to-date hardware and firmware information for the SIMOTION device is displayed on the Device Info page:

<table>
<thead>
<tr>
<th>Manufacturer Name</th>
<th>Siemens AG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Number</td>
<td>Order number (MLFB) of the device</td>
</tr>
<tr>
<td>Revision Number</td>
<td>Hardware version</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Serial number of the SIMOTION device</td>
</tr>
<tr>
<td>User Version Firmware</td>
<td>SIMOTION Kernel user version</td>
</tr>
<tr>
<td>Build Number</td>
<td>Internal version number</td>
</tr>
<tr>
<td>Additional Hardware</td>
<td>Installed components of the SIMOTION device including:</td>
</tr>
<tr>
<td></td>
<td>MLFB, serial number, revision number, firmware name, user version number, internal version number</td>
</tr>
<tr>
<td>Technological Packages</td>
<td>Loaded technology packages including:</td>
</tr>
<tr>
<td></td>
<td>Package name, user version number, internal version number</td>
</tr>
</tbody>
</table>
3.3 Simplified standard pages

Device Info

Manufacturer Name: SIEMENS AG
Order Number: 6AU1 485-6AA00-6AA1
Revision Number: F
Serial Number: ST-V01234567
User Version Firmware: V 4.2.0.0
Build Number: V 60.0.0.0; unc060d435kernel.9 builder
Licence Serial Number: ST0E822220000017918
Operating State: STOP

Additional Hardware

<table>
<thead>
<tr>
<th>MLFB</th>
<th>Serial-Nr.</th>
<th>Revision-Nr.</th>
<th>FW-Name</th>
<th>User-Ver.</th>
<th>Build-Nr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAU1000-0400-0400</td>
<td>ST686222200000017918</td>
<td></td>
<td>INFIXICS integrated</td>
<td>V 60.0.0</td>
<td>V 0.0.0</td>
</tr>
<tr>
<td>GFO5312-0FA00-0400</td>
<td>ST-V01658119</td>
<td></td>
<td>prokernel PA-V2.2</td>
<td>V 12.1.8.95</td>
<td>V 0.0.0.0 112730000001000056</td>
</tr>
<tr>
<td>Root loader</td>
<td>44444444444444</td>
<td></td>
<td></td>
<td>V 520.0.0</td>
<td>V 0.0.0</td>
</tr>
<tr>
<td>RDIS</td>
<td>V60 00:00:00</td>
<td></td>
<td></td>
<td>V 60.0.0</td>
<td>V 0.0.0</td>
</tr>
</tbody>
</table>

Technological Packages

<table>
<thead>
<tr>
<th>TP-Name</th>
<th>User-Ver.</th>
<th>Build-Nr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpccm</td>
<td>V 4.2.0.0</td>
<td>V 60.0.0.0; unc060_d88fpccmwin9.builder</td>
</tr>
</tbody>
</table>

Figure 3-56 Device info on simplified HTML pages
3.3.3 Diagnostics

Overview of the general state of the SIMOTION control

The Diagnostics page displays the following states of the SIMOTION control:

- **Systemtime**: Current time of day of the SIMOTION control
- **Timezone**: Current difference between the Systemtime and GMT in minutes
- **CPU Load by cyclic Tasks**: Processor time of servo and IPO levels as a percentage of the total processor time
- **Memory Load**: Size and allocation of the memory, RAM disk, memory card, and non-volatile memory in bytes
- **State**: Current operating mode of the SIMOTION control

![SIMOTION D435](Image)

**Diagnostics**

**Systemtime:**
Thu Jun 24 11:25:19 2010

**Timezone:**
GMT +60 min

**CPU load by cyclic tasks:**
10%

**Memory Load:**

<table>
<thead>
<tr>
<th></th>
<th>Used Bytes</th>
<th>Size Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM - Disk</td>
<td>4062208</td>
<td>25065984</td>
</tr>
<tr>
<td>RAM</td>
<td>11767803</td>
<td>36700160</td>
</tr>
<tr>
<td>Memory Card</td>
<td>44818432</td>
<td>512180224</td>
</tr>
<tr>
<td>Retentive Data</td>
<td>1736</td>
<td>373240</td>
</tr>
</tbody>
</table>

**State:**

- DC 5V
- RUN
- STOPU
- STOP

Figure 3-57 Diagnostics shown on simplified HTML pages
### 3.3.4 Diag buffer

#### Diag buffer information

The Diag buffer page shows the events in the diagnostics buffer.

- **Time**: Time of the event
- **Date**: Date of the event
- **Event**: Displays the event as text.
- **HexValue**: Hex value of the diagnostics buffer message

#### Diag buffer information

If the DGBUFTXT.EDB language file is missing, it will be displayed in hexadecimal notation.

**Figure 3-58 Diagnostics buffer shown in simplified format**

<table>
<thead>
<tr>
<th>Nr</th>
<th>Time</th>
<th>Date</th>
<th>Event</th>
<th>HexValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11:28:22.083</td>
<td>24.06.10</td>
<td>Sinamics Integrated: Start of diagnostic buffer, station address = 3</td>
<td>16#F30E305 16#0000 16#0000 16#00 16#00</td>
</tr>
<tr>
<td>2</td>
<td>15:30:55.744</td>
<td>10.02.92</td>
<td>Ramp-up completed, cyclic operation</td>
<td>16#F30240C 16#0000 16#0000 16#00 16#00</td>
</tr>
<tr>
<td>3</td>
<td>15:30:52.897</td>
<td>10.02.92</td>
<td>Cyclic data exchange P2D F1 started</td>
<td>16#F302416 16#0000 16#0000 16#00 16#00</td>
</tr>
<tr>
<td>4</td>
<td>15:30:52.981</td>
<td>10.02.92</td>
<td>Cyclic data exchange P2D F1 completed</td>
<td>16#F302418 16#0000 16#0000 16#00 16#00</td>
</tr>
<tr>
<td>5</td>
<td>15:30:51.794</td>
<td>10.02.92</td>
<td>Fault D03: Fault code 1042, fault value 0x00004e0</td>
<td>16#F30241D 16#0412 16#C000 16#00 16#00</td>
</tr>
</tbody>
</table>
### 3.3.5 Diag buffer drive

#### Diag buffer drive information

The **Diag buffer drive** page shows the events in the drive diagnostics buffer for the integrated drives.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:28:22</td>
<td>003</td>
</tr>
<tr>
<td>15:30:55:744</td>
<td>10:02:92</td>
</tr>
<tr>
<td>15:30:52:897</td>
<td>10:02:92</td>
</tr>
<tr>
<td>15:30:52:881</td>
<td>10:02:92</td>
</tr>
<tr>
<td>15:30:51:794</td>
<td>10:02:92</td>
</tr>
</tbody>
</table>

If the DGEXTXT.EDB language file is missing, it will be displayed in hexadecimal notation.

#### HexValue

- Hex value of the drive diagnostics buffer message

---

SIMOTION D435

---

**Extended Diagnostic buffer**

<table>
<thead>
<tr>
<th>Nr</th>
<th>Time</th>
<th>Date</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11:28:22</td>
<td>003</td>
<td>24.06.10</td>
</tr>
<tr>
<td>2</td>
<td>15:30:55:744</td>
<td>10:02:92</td>
<td>Ramp-up completed, cyclic operation</td>
</tr>
<tr>
<td>3</td>
<td>15:30:52:897</td>
<td>10:02:92</td>
<td>Cyclic data exchange PZD F1 started</td>
</tr>
<tr>
<td>4</td>
<td>15:30:52:881</td>
<td>10:02:92</td>
<td>Cyclic data exchange PZD F1 completed</td>
</tr>
<tr>
<td>5</td>
<td>15:30:51:794</td>
<td>10:02:92</td>
<td>Fault DC3: fault code 1042, fault value 0x0000400</td>
</tr>
</tbody>
</table>

---

Figure 3-59 Diag buffer drive
### 3.3.6 Alarms

Information about alarms

<table>
<thead>
<tr>
<th>Level</th>
<th>Category of the alarm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Time of the alarm</td>
</tr>
<tr>
<td>TO</td>
<td>Technology object that triggered the alarm</td>
</tr>
<tr>
<td>Nr</td>
<td>Alarm number</td>
</tr>
<tr>
<td>Text</td>
<td>Displays the alarm message as text</td>
</tr>
</tbody>
</table>

#### SIMOTION D435

**Alarms**

**Alarm Count:** 1

<table>
<thead>
<tr>
<th>Level</th>
<th>Time</th>
<th>TO</th>
<th>Alarm</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>24.06.10</td>
<td>Achse_1</td>
<td>20805</td>
<td>Device type 2, log address 256 faulted (Bit 0, encoder number 0, reason 0x80h)</td>
</tr>
</tbody>
</table>

Figure 3-60 Alarms shown in simplified format
3.3.7 IP Config

Data of the SIMOTION control Ethernet interface

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Address of the TCP/IP interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet Mask</td>
<td>Subnet Mask of the interface</td>
</tr>
<tr>
<td>MAC Address</td>
<td>Address of the network card</td>
</tr>
<tr>
<td>Gateway</td>
<td>Default gateway of the interface</td>
</tr>
</tbody>
</table>

The corresponding information is always displayed in the first column. It is not necessarily directly related to the IP address of the column and may even have been configured for the other interfaces.

Figure 3-61 IP Config
3.3.8 Diagnostic files

Backing up diagnostic pages of the web server

You can use this page to back up general diagnostic data and individual IT DIAG HTML pages.

![SIMOTION D435](image)

**Diagnostic files**

Create general diagnostic files

This function will create several diagnostic information files and save them to memory card of the SIMOTION device under folder SYSLOG/DIAG.

Please note: Application is running in background. Please wait a few seconds after pressing the button!

- Create general diag files

**HTML - diagnostic files**

This function will save some of the present HTML-files containing diagnostic information to memory card of the SIMOTION device under folder SYSLOG/DIAG.

To customize the list of files, please edit file "DIAGURLS.TXT" in the same folder.

Please note: Application is running in background. Please wait a few seconds after pressing the button!

- Create html diag files

**Get diagnostic files**

After pressing button “Zip all diag files”, all diagnostic information files (general and HTML) which are present on the memory card of the SIMOTION device will be zipped into a file called “DIAGARCHIVE.ZIP”.

Please note: Applications are running in background. Please wait a few seconds after pressing a button!

- Zip all diagfiles

**Delete diagnostic files**

This function will delete all diagnostic information files (general and html) which are present on the memory card of the SIMOTION device under folder SYSLOG/DIAG.

Please note: Application is running in background. Please wait a few seconds after pressing the button!

- Delete all diagfiles

Figure 3-62 Diagnostic files

**See also**

Diagnostic files (Page 49)
3.3.9 Watch tables

Watch tables

Figure 3-63 Watch tables

This page shows all created Watch tables. These Watch tables are the same as on the standard IT DIAG page. They can be saved, deleted, and uploaded. Editing is not possible here.
Figure 3-64 Display of a Watchtable

See also

Watch (Page 34)
3.3.10 User's Area

User-defined pages are displayed in the User's Area.

Figure 3-65  User's Area

User's Area
3.4 IT DIAG configuration

3.4.1 Introduction

There are two files which can be used to configure IT DIAG:

- WebCfg.xml
- WebCfgFrame.xml

The WebCfg.xml configuration file is used to make user-relevant settings in the web server.

The WebCfgFrame.xml contains the manufacturer's IT DIAG settings.

WebCfg.xml

The file is subdivided into several different sections, e.g. server options and user database. WebCfg.xml can be reloaded during runtime. This action restarts the web server. The modified settings are available after the restart.

The Manage Config > IT DIAG standard pages can be used to safely modify entries in WebCfg.xml. IT DIAG Base (Page 74)

The configuration file is divided into various areas:

- Virtual file system: Mapping of a hierarchical file system by means of the configuration.
- Server options: Replace the home page of the standard diagnostic pages with your own home page (see User-defined home page (Page 121)), port settings.
- Configuration area: Module-specific configuration data
- User database: Controls access to the diagnostic pages (see Log-in administration (Page 18) section).
- File types: Specification of the MIME type (Page 271) in the HTTP header.

The WebCfg.xml file can be found either on the supplied DVD in the 3_Configuration directory (in the default state) or on the SIMOTION control memory card in the USER\SIMOTION\HMICFG\ directory.

WebCfgFrame.xml

WebCfgFrame.xml contains the IT DIAG basic settings for the device.

NOTICE

Please change only the settings described in this document. Changes to WebCfg.xml and WebCfgFrame.xml settings not described in this manual can lead to errors.
3.4.2 Overview

An overview of the possible configurations available is provided here. The appendix contains further explanations and examples of the individual options.

Due to their complexity, the virtual file system and the <DEFAPP> tag are explained in more detail on the following pages.

WebCfg

Table 3-3  WebCfg.xml overview

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value/Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ALTERNATE_PORTNUMBER&gt; (Page 267)</td>
<td>[Port number]:</td>
<td>Alternative port number</td>
</tr>
<tr>
<td></td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>&lt;ALTERNATE_SSL_PORTNUMBER&gt; (Page 268)</td>
<td>[Port number]:</td>
<td>Alternative SSL port number</td>
</tr>
<tr>
<td></td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>&lt;BASE&gt; (Page 268)</td>
<td></td>
<td>Link lists for all HTML pages</td>
</tr>
<tr>
<td>&lt;CONFIGURATION_DATA&gt; (Page 270)</td>
<td></td>
<td>This tag encloses the other configuration data</td>
</tr>
<tr>
<td>&lt;MIME_TYPES&gt; (Page 271)</td>
<td>[&lt;FILE EXTENSION&gt; Tags]: Text</td>
<td>Assignment of file types to file extensions</td>
</tr>
<tr>
<td>&lt;SERVEROPTIONS&gt; (Page 272)</td>
<td></td>
<td>Contains the server options</td>
</tr>
<tr>
<td>&lt;TIMEZONE&gt; (Page 273)</td>
<td>[+-Minutes]:</td>
<td>Time zone synchronization</td>
</tr>
<tr>
<td></td>
<td>Signed integer</td>
<td></td>
</tr>
<tr>
<td>&lt;USERDATABASE&gt; (Page 274)</td>
<td></td>
<td>Access protection can be provided for a wide range of areas of the web server, starting with HTML pages, directories, etc., right up to individual actions of applications.</td>
</tr>
</tbody>
</table>

Table 3-4  <SERVEROPTIONS>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BROWSEABLE&gt; (Page 269)</td>
<td>Allow or prohibit browsing in a directory or globally.</td>
</tr>
<tr>
<td>&lt;DEFAULTDOCUMENT&gt; (Page 270)</td>
<td>Document that is output if a document is not contained in the URL.</td>
</tr>
<tr>
<td>&lt;PORTNUMBER&gt; (Page 272)</td>
<td>Port number of the TCP/IP server. Default : 5001</td>
</tr>
<tr>
<td>&lt;SSLPORTNUMBER&gt; (Page 273)</td>
<td>Port number of the SSL port. Default : 5443</td>
</tr>
</tbody>
</table>
WebCfgFrame

Table 3-5  WebCfg.xml overview

<table>
<thead>
<tr>
<th>Tag</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;BASE&gt;</code> (Page 280)</td>
<td>Lists of links for standard pages (HTML and XML)</td>
</tr>
<tr>
<td><code>&lt;CONVERSION&gt;</code> (Page 281)</td>
<td>1. Sends data</td>
</tr>
<tr>
<td></td>
<td>2. Receives WebCfg.xml</td>
</tr>
<tr>
<td></td>
<td>3. Generates the directory browser</td>
</tr>
<tr>
<td><code>&lt;DEFAPP&gt;</code> (Page 282)</td>
<td>Default application</td>
</tr>
<tr>
<td><code>&lt;HTTP_RESULT_CODES&gt;</code> (Page 284)</td>
<td>Adaptation of HTTP error messages</td>
</tr>
<tr>
<td><code>&lt;SSL&gt;</code> (Page 285)</td>
<td>Contains an entry for encrypting the SSL connection</td>
</tr>
</tbody>
</table>
3.4.3 Configuration of the file system

3.4.3.1 Virtual file system

The web server was primarily developed for embedded devices. It cannot be categorically assumed that these target systems contain file systems, such as memory cards or hard disks.

However, the addressing model that is used on the Internet (URLs) assumes a hierarchical file system on the server.

Because XML files also exhibit a hierarchical structure, it makes sense to use an XML file to map the required file hierarchy to a file (or memory block).

That is the main reason why the web server has the WebCfg.xml XML configuration file. It also made sense to store other configuration parameters in this file in order to achieve a central location where the web server is configured together with its modules. This file can also be created and loaded automatically, e.g. if a CS system (such as SIMOTION SCOUT, STEP7, etc.) is to configure the web server in a target system together with other configuration parameters of the target system.

Each time a request is made by a client, a matching entry (tag) or link is first searched for in the "XML file system". All security settings, etc., are specified exclusively in the "XML file system".

Structure of XML file system within the XML configuration file:

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE LOCALLINK="link_to_external_file_system">
        <www LOCALLINK="/" BROWSEABLE="TRUE"
             REALM="FileAdministrator"/>
        <Test.mcs LINK="/Tests/Test.mcs="/>
        <Default.mcs>
            <![CDATA[
                <HTML>
                    <HEAD>
                        [...]
                    </HEAD>
                ]]>  
            </Default.mcs>
    </BASE>
    [...]
</SERVERPAGES>
```
3.4.3.2 Virtual vs. physical file system

As explained in the "Virtual file system" section previously, each file access searches in the XML file system first.

The following scenarios are possible:

1. A link to the physical file system is found. The file is loaded from the physical file system and sent back to the client.

2. The searched file is available in the XML file system and then is returned to the requesting client.

The two data access scenarios are graphically represented as follows:

![Diagram showing possible accesses via XML file system](image)

Figure 3-66 Possible accesses via XML file system

**Note**

A link can also be a link to a directory and not just to a file. If a link is found when a URL is being resolved, it is replaced by its target and then searching continues.

As far as the security settings of the external file system are concerned, it is those of the most recently found XML node that apply, i.e. the node whose LOCALLINK attribute triggered the search in the external file system.
3.4.3.3 Set external file system as preferential <PREFER_EXTERNAL>

The `PREFER_EXTERNAL` attribute can be set only in conjunction with `LOCALLINK`.

If `PREFER_EXTERNAL` is set and a file exists in both the XML and the external file system, preference is given to the file from the external file system.

If this attribute is not set, preference is given to the file from the XML system.

The same applies to directories (with write access).

3.4.3.4 Links to the physical file system <LOCALLINK>

Local links are the sole means of accessing the physical file system.

Each data node of the XML file system can have a `LOCALLINK` attribute, including the `<BASE>` node. The `<BASE>` node corresponds to the root entry of the file system. If a `LOCALLINK` attribute was found, an attempt will continue to be made nevertheless to parse the XML file system. The XML file system has precedence over the external file system. This rule applies to read and write accesses.

This priority can be changed using the `PREFER_EXTERNAL` attribute so that the external file system takes precedence over the XML file system.

**Note**

Files which already exist in the file system may be concealed as a result of this order of precedence.

If another `LOCALLINK` attribute is found during further XML parsing, the last attribute found always applies.

Please note that the search in the physical file system always starts at `$WWWRoot`. `$WWWROOT` refers to the base path WWWRoot; the external file system is accessed relative to this base path.

Links to directories must always be ended with a hierarchy separator (`"/"`).
Examples of accessing the file system:

<table>
<thead>
<tr>
<th>URL</th>
<th>Target in the physical file system</th>
<th>Entry in WebCfg.xml</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Default.mcs</td>
<td>$WWWROOT/Default.mbs</td>
<td>&lt;Default.mbs LOCALLINK=&quot;$WWWROOT/Default.mbs&quot; PREFER_EXTERNAL=&quot;TRUE&quot;/&gt;</td>
<td>Hides a file from the XML directory. See WebCfg.xml example above.</td>
</tr>
<tr>
<td>/NewFile.mcs</td>
<td>$WWWROOT/NewFile.mbs</td>
<td>&lt;NewFile.mbs LOCALLINK=&quot;$WWWROOT/NewFile.mbs&quot; PREFER_EXTERNAL=&quot;TRUE&quot;/&gt;</td>
<td>Makes a file from the physical file system accessible.</td>
</tr>
<tr>
<td>/index.mcs</td>
<td>$WWWROOT/NewDir/index.mbs</td>
<td>&lt;index.mbs LOCALLINK=&quot;$WWWROOT/NewDir/index.mbs&quot; PREFER_EXTERNAL=&quot;TRUE&quot;/&gt;</td>
<td>Hides a standard page.</td>
</tr>
<tr>
<td>/XMLDir/NewFile.mcs</td>
<td>Kindknoten von &lt;XMLDir&gt; im XML-Dateisystem</td>
<td></td>
<td>Adds a node in the XML file system.</td>
</tr>
</tbody>
</table>

The files in the physical file system must have the extension .mbs, otherwise the web server will issue an error message. These pages are output in the browser with the extension .mcs. This also applies to linked pages.

### 3.4.3.5 Browsing of directories

Browsing (Browse) of directories can be activated or deactivated.

This is controlled using the **BROWSEABLE** attribute. If the attribute is **TRUE**, a directory view is allowed.

Setting the **BROWSEABLE** value to **TRUE** enables the browsing of directories by default.

The following example assumes that WWWRoot points to a directory with the following structure:

```xml
/  
  /Datei1
  /Directory1/
  /Directory1/Datei2.mcs
  /Directory1/Datei3.mcs
  /Directory1/Directory2
  /Datei4
```
WebCfg.xml:

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...] 
    <BASE LOCALLINK="/">
        <www LOCALLINK="/" BROWSEABLE="TRUE" .../>
    </BASE>
    [...] 
</SERVERPAGES>
```


In the XML file system, the parser searches for www in the root directory and finds LOCALLINK="/".

In the physical file system, the parser searches for /Directory1. The forward slash "/" in this path is retained, as was specified in the LOCALLINK="/" tag. Directory1 refers to the path.

The Directory1 directory exists in the physical file system. Since Browseable = TRUE and no default HTML page has been specified, the browse view of the directory is returned.

See also

- `<BROWSEABLE>` (Page 269)
- `<DEFAULTDOCUMENT>` (Page 270)

3.4.3.6 Security concept

Permission information in the form of attributes can be stored at each XML node of the XML file system:

- **REALM** (secure area)
- **READ** (reading rights)
- **WRITE** (writing rights)
- **MODIFY** (modification rights)

**REALM** may only contain one group name, while **READ**, **WRITE**, and **MODIFY** may contain a list of group names separated by "," characters. No spaces or other Whitespace characters may be used.

A set of user groups is assigned to each user.

If a file is requested by a user, the XML file system is searched through for this file. The XML tree is run through corresponding to the file path. If several XML nodes are run through, the logged-in user must have rights for all of the "touched" nodes.
Example:

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>

[...]
</SERVERPAGES>

<BASE LOCALLINK="/">
    <MainDir REALM="USER" LOCALLINK="/Base/">
        <www LOCALLINK="/WebSeiten/">
            BROWSEABLE="TRUE"
            READ="Administrator"
            WRITE="FileAdministrator" />
    </MainDir>
    <Test.mcs LOCALLINK="/Tests/Test.mcs/"/>
    <XMLDir>
    </XMLDir>
    <Default.mcs>
        <![CDATA[
            <HTML>
            <HEAD>
            [...] 
        ]]> 
    </Default.mcs>
</BASE>

[...]
</SERVERPAGES>
```

<table>
<thead>
<tr>
<th>URL</th>
<th>Access</th>
<th>Groups</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>/&lt;File&gt;.mcs</td>
<td>Read</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>/&lt;File&gt;.mcs</td>
<td>Write</td>
<td>None</td>
<td>Access not permitted</td>
</tr>
<tr>
<td>/MainDir/&lt;File&gt;.mcs</td>
<td>Read</td>
<td>USER</td>
<td>Login mask if USER group is not present</td>
</tr>
</tbody>
</table>
3.4.3.7 REALM

Setting up a security area

Realm is used to designate a secure area in the WWW environment. If a directory is entered and the user is not a member of the specified realm (or the user has not yet logged in), a login prompt appears (authentication required).

If a file protected by REALM is accessed, the client must be authenticated. Web browsers usually display a user prompt requiring the user to enter his/her user name and password:

![Figure 3-67 Login for Internet Explorer](image)

The site identifies the address of the server (in this case, localhost). Realm is the secure area and corresponds to the REALM attribute.

The REALM attribute can be used to enable or force a user login.

Note

Only one REALM can be specified for a directory. In a directory hierarchy, different REALMS must be separate, not superimposed.

Because the file objects are accessed on a hierarchical basis, different hierarchy levels may well have different secure groups.

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        <Motion REALM="Operator">
            <Tests REALM="Tester">
                [...]
            </Tests>
        </Motion>
    </BASE>
    [...]
</SERVERPAGES>
```

An access to Tests and its content is not possible. This is true even if a user is in the "Operators" and "Testers" security groups.
Here, a user with the "Operators" and "Tester" secure groups has access to Motion and Tests as well their subordinate objects.

See also

<USERDATABASE> (Page 274)
REALM attribute (Page 278)

3.4.3.8 READ

Setting up read authorization with the READ attribute

If the READ attribute is specified for a directory, the user must be a member of one of the groups specified for the READ attribute. With READ, several groups can be specified. These must be separated with commas and no white space characters may be used.

Example

<MyDir READ="User,Administrator" />

Users that belong to the User or Administrator group (or both) may read the content of the directory.

If a user does not have read rights, i.e. he/she does not belong to any of the groups that are specified with READ, a FORBIDDEN message is generated. A login for the client is not initiated.

If no READ attribute is present for a directory, read access is always permitted.

See also

READ attribute (Page 277)
3.4.3.9 WRITE

Setting write authorizations with the WRITE attribute

If a directory has a WRITE attribute and the logged-in user is a member of one of the specified groups, the user may only create new files in this directory.

The user may:
- Not create any new directories
- Not overwrite any files
- Not delete any files
- Create new files

Note
To create files, the user also needs READ rights!

See also
WRITE attribute (Page 279)

3.4.3.10 MODIFY

Unlocking directories for modification

If a directory has a MODIFY attribute and the logged-in user is a member of one of the specified groups, the user may carry out all write operations in this directory:

The user may:
- Create new directories
- Overwrite files
- Delete files
- Create new files

The user must, of course, have READ rights for the directory as well (otherwise, he/she would not have access to the directory to start with).

See also
MODIFY attribute (Page 276)
3.4.3.11 Creating directories and files

If directories or files are created, they inherit the authorizations of the directory that contains them.

Rights cannot be changed via the directory browser. Rather, they can only be changed directly by modifying the WebCfg.xml file.

3.4.3.12 Browsing the file system

The web server allows you to visualize a (physical) directory in the client.

For this purpose, the BROWSEABLE attribute for the LOCALLINK tag or the global <BROWSEABLE>-tag must be set to TRUE.

If a client accesses this link, a directory view of the directory is created. Navigation from this directory to subdirectories is also possible (also to higher-level directories if browsing is allowed for them).

Provided you have sufficient permissions, you can send, receive and delete files as well as create and delete directories. The appearance of the directory in the client can be freely configured.

The <DEFAPP> tag = "Default application" is used to configure the directory browser.

If there is no authentication mechanism on the web server, write access is generally not permitted (see Security concept).

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    ...
    <BASE>
        <www LOCALLINK="/UserData" BROWSEABLE="TRUE"
             REALM="Bediener"/>
        <Test.mcs LINK="/Tests/Test.mcs"/>
        <Default.mcs>
            <![CDATA[
                <HTML>
                    <HEAD>
                        [...]
                    ]]>  
                </Default.mcs>
            [...]  
            </SERVERPAGES>

In this example, a directory view of the local directory "/UserData" (relative to WWWRoot!) would be returned to the client if it requests the URL /www and has been authenticated as a user of the "Operator" REALM.

Write access to the directory is not possible because a WRITE or MODIFY attribute has not been specified for the directory entry.
```
3.4.3.13 File access via FTP

Introduction

The "File access via FTP" function makes it possible to access specific files on the memory card. An FTP client can be, for example, Windows Explorer.

Condition

In the WebCfg.xml file, a user must be in the "FTPUser" group in order to log on to the FTP. The WebCfg.xml file is generated during the first ramp-up.

Excerpt from WebCfg.xml

In the excerpt from WebCfg.xml displayed below, the "FTPUser" group has been assigned to the "simotion" user.

```xml
<UserDataBase>
  <FILE NAME="UserDataBase.xml">
    <!CDATA[
      <?xml version="1.0" encoding="UTF-8"?>
      <UserDataBase>
        <USER NAME="anonymous" PASSWORD="anonymous">
          <DESCRIPTION>Anonymous</DESCRIPTION>
          <GROUP NAME="Anyone"/>
          <GROUP NAME="OPC_XML"/>
        </USER>
        <USER NAME="internal" PASSWORD="internal">
          <DESCRIPTION>Internal user</DESCRIPTION>
          <GROUP NAME="Anyone"/>
        </USER>
        <USER NAME="simotion" PASSWORD="simotion">
          <DESCRIPTION>Default User</DESCRIPTION>
          <GROUP NAME="Administrator"/>
          <GROUP NAME="FTPUser"/>
          <GROUP NAME="Anyone"/>
          <GROUP NAME="OPC_XML"/>
        </USER>
      </UserDataBase>
    ]]>>
    </FILE>
  </UserDataBase>
```
3.4 IT DIAG configuration

WARNING
Caution while accessing system files.

Note
FTP access is only supported in binary mode.
3.5 User-defined pages

3.5.1 Introduction

Individually designed web pages with access to device data

SIMOTION IT DIAG offers the option of creating individually designed web pages. The options for designing these web pages are described in this section.

For access to device data, two variants are available for selection:

- The JavaScript libraries opcxml.js and appl.js
- The MiniWeb server language (MWSL)

The screenshot below shows an example of a user-defined page that displays variables in tabular format.

![User-defined page](image)

Figure 3-68 User-defined page
3.5.2 Converting standard HTML pages to binary files

IT DIAG uses its own internal file type ".mbs" to output HTML pages. The conversion tools U7NW2XMX.exe (graphic user interface) or U7NW2XCX.exe (command line) can be used to create these files as of SIMOTION control V4.1.

For the purposes of the conversion, MCS files with the same basic structure as standard HTML pages must be created. Additional tags and functions specific to IT DIAG can be used to access the device data and to optimize page display.

Steps for creating and converting HTML pages
1. Create the HTML pages with a tool of your choice. The pages are assigned the file extension ".mcs".
2. Call the conversion tool with the source directory and target directory. The converted files will appear in the target directory with the same name as the associated source files and the file extension ".mbs".
3. Copy the files from the target directory to the \USER\SIMOTION\HMI\FILES directory on the memory card of the control. Alternatively, files can be transferred to the device using the Files page.

Converting HTML files to binary files using the conversion tool

Figure 3-69 U7NW2XCX.exe conversion tool

Optionally, you can select a source file or source path. When you specify a source path, all *.mcs files are converted starting from this path.

Press the Start button to convert the HTML pages and store them in the target path.
Converting HTML files to binary files using the command line conversion tool

The call syntax is:

`U7NW2XCX.exe [-f] <source> <target>`

**Call parameters:**

- **-f**  
  Optional: Processing of an individual file

- **Source**  
  Source path or source file.  
  When you specify the source path, all *.mcs files are converted starting from this path.

- **Destination**  
  Target directory

**Note**

You will find the conversion tools on the SIMOTION SCOUT AddOn DVD, in the 4_Accessories\Simotion_IT6_Tools directory.

**Example of a user-defined page**

An HTML page can be created using any text editor.

```html
<html>
<head/>
<body>
  <br>&nbsp State: <MWSL<!--WriteVar("DeviceInfo.BZU");--></MWSL>
</body>
</html>
```

Newfile.mcs

In this example, the source text is saved as a file with the name Newfile.mcs.

The MWSL is used in the example to output device data. Structure of a MWSL file (Page 159)

The device data is accessed in the MWSL-Ausdrücken by means of the variable provider. DeviceInfo group (Page 244)

Once the conversion is complete, you receive the file Newfile.mbs, which you copy to the device using the files page.
3.5 User-defined pages

Figure 3-70  Newfile.mbs in the browser

The page can only be displayed if the User's Area settings are correct. In this example, the EmbeddedSimple version has been selected. Embedded, user-defined pages (Page 122)

3.5.3 User-defined Home page

You can create your own home page and display it instead of the home page for the standard diagnostic pages of the control. To do this, you need to change the default page of the web server in the WebCfg.xml file.

Procedure

1. Create your own home page and save it, for example, under the name MYINDEX.MCS.
2. Transfer the home page to the memory card of the SIMOTION device using the Files page.
3. Open the WebCfg.xml file in an available editor. The file can be found either on the supplied DVD in the 3_Configuration directory (in the default state) or on the SIMOTION device memory card (possibly in a modified state) in directory \USER\SIMOTION\HMICFG.
4. Replace the file name index.mcs in the <SERVEROPTIONS> in element <DEFAULTDOCUMENT VALUE="index.mcs" /> with the name of your home page, including the path name "files" (all user-defined HTML pages are stored in the FILES directory).
5. Save the changed WebCfg.xml on the memory card via the "Settings" page.

An example of Embedding an HTML page is provided in the chapter titled 'Virtual file system (Page 106)."
3.5.4 Embedded, user-defined pages

Integrating user-defined pages

With Version 4.1.3 and higher, it is possible to embed user-defined pages in the framework of standard SIMOTION pages.

![Figure 3-71 User's Area with embedded page](image)

The menu in the User's Area links the files in the FILES folder in two different ways:

- **EmbeddedSimple**: The User's Area page loads all web pages contained in the FILES folder as tabs. The file name is shown without an extension.
- **Embedded**: The page loads a user-definable tab.

You can switch between EmbeddedSimple and Embedded on the Settings (Page 84) page.
Settings in WebCfg.xml

In WebCfg.xml, the appearance of the User's Area can be set with the configuration constants `<UserArea>` and `<UserDir>`.

The `<UserArea>` tag can be used to set how the tab is displayed (the default setting is shown in bold):

```
<UserArea>({ StandAlone | Embedded | EmbeddedSimple })</UserArea>
```

`<UserDir>` denotes the directory for the tab files relative to the FILE directory.

```
<UserDir></UserDir>
```

**StandAlone**

```
<UserArea>StandAlone</UserArea>
```

Access to the User's Area is permanently linked to the user.mcs file. In order to display the User's Area, this file must exist and be available for calling.

**Automatic**

```
<UserArea>EmbeddedSimple</UserArea>
```

Select this option to use all the files found in the directory labeled `<UserDir>` to create the tab.

The respective file name (without the extension) is used as the title, and the corresponding menu link refers to this file.

**Using the menu editor**

```
<UserArea>Embedded</UserArea>
```

Select this option to display the menu editor (Page 124), which is used to make custom settings for menus, in the User's Area.

WebCfg.xml example:

```
<SERVERPAGES version="59.00">
  [...]
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <UserArea>Embedded</UserArea>
      <UserDir/>
    </USERCONFIG>
  </CONFIGURATION_DATA>
  [...]
</SERVERPAGES>
```
3.5.5 Menu editor

Creating individual menus using the menu editor

With Version 4.1.3 and higher, you can use the Menu editor link to call the menu editor, which enables you to configure custom menus for the User's Area.

Prerequisites for using the menu editor

In order to use the menu editor, the `<UserArea>` must be set to Embedded in the configuration data in WebCfg.xml.

![Configuration data menu editor](image)

Figure 3-72 Configuration data menu editor

The Enable user menu editor option must then be selected on the Settings page.
### 3.5 User-defined pages

**Working with the menu editor**

The User's Area page now contains the **Menu editor** tab.

**Figure 3-73** Settings menu editor

**Figure 3-74** Starting the menu editor for the first time

The first time the menu editor is launched, a largely blank page appears.
New menu items can be created with the **Append new entry** button.

![Menu editor](image)

**Figure 3-75** Menu editor with several items

In the screenshot above, the newfile.mcs file has been added. The buttons can be used to add or delete and change the position of files.

The names of the files to be displayed for the corresponding menu commands are entered in the **MCS file name** column.

The **Menu text** column contains the name of the menu item.

- **Button ①** is used to create new menu items; these are inserted before the current item in each case.
- **Button ②** deletes the corresponding menu item.
- **Button ③** moves the menu item up.
- **Button ④** moves the menu item down.
3.5.6 JavaScript and web services

3.5.6.1 Variable access with JavaScript and web services

Access to a device with the JavaScript library

The DOM functionality of JavaScript enables simple web service clients to be implemented. This opens up a multitude of new options within a browser, e.g.:

- Reading and cyclic updating of variable contents using an OPCXML read command
- Writing of variables using an OPC XML-DA Write command
- Browsing of the entire SIMOTION variable management area
- Setting up and querying an OPC XML-DA Subscription

The functionality is provided by several JavaScript files:

- opcxml.js: Contains functions for the structure of the necessary XML documents and for communication with an OPC XML-DA server.
- appl.js: Based on opcxml.js and implements the following objects:
  - Variable browser. Representation of the SIMOTION variable management area in a tree topology within the browser
  - Property viewer: Representation of variable properties (value, data type, access rights, Enums) in the form of a table within a browser. For writable variables, the table contains an entry field for changing the variable content.
  - Watch table: Representation of a watch table in the browser
3.5.6.2 Communication with the OPC XML DA server (opcxml.js)

OPCReadRequest

The OPCReadRequest class can be used to read the values for a list of variables.

```javascript
function OPCReadRequest(parLocaleId, parResultCB)

Transfer parameters:
- parLocaleId: Language identifier ("DE", "EN")
- parResultCB: Callback function that must be provided by the caller
  This function is called by OPCReadRequest when a response has arrived from the OPC XML DA server. The OPCReadRequest object is disposed of automatically (by calling the "destructor" method) if the callback function supplies "true" as a return value.

function OPCReadRequestCB(parResponse)

Transfer parameters:
- parResponse: Array of ItemValues with the result of the read request.

function ItemValue()
{
    this.mItemPath;
    this.mItemName;
    this.mItemHandle;
    this.mItemValue;
    this.mItemResultId;
}
```

If mItemResultId is defined, an error occurred during reading. In this case, mItemResultId is assigned the OPC XML DA error ID.
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=ISO-8859-1">
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript">
        function read()
        {
            var tmpReadCB = function(parResponse)
            {
                tmpResultStr = "";
                for (var tmpIndex = 0; tmpIndex < parResponse.length;
                tmpIndex++)
                {
                    var tmpItemValue = parResponse[tmpIndex];
                    var tmpValue = (tmpItemValue.mItemValue) ?
                        tmpItemValue.mItemValue :
                        tmpItemValue.mItemResultId;
                    tmpResultStr += tmpItemValue.mItemPath
                        + "::
                        + tmpItemValue.mItemName
                        + " = "
                        + tmpValue
                        + "\n";
                }
                alert(tmpResultStr);
                return true;
            }
            var tmpReadRequest = new OPCReadRequest("DE",tmpReadCB);
            tmpReadRequest.addItem("SIMOTION","var/userdata.user1");
            tmpReadRequest.addItem("SIMOTION","var/userdata.user2");
            tmpReadRequest.addItem("SIMOTION","var/userdata.user10");
            tmpReadRequest.sendReadRequest();
        }
    </script>
    <title>Insert title here</title>
</head>
<body>
    <input type="button" onclick="read();" value="Read"/>
</body>
</html>
```
OPCGetPropertiesRequest

The OPCGetPropertiesRequest class can be used to read the properties of variables:
- Values
- Data types
- Access rights
- Enum components

function OPCGetPropertiesRequest(parLocaleId, parResultCB)

Transfer parameters:
- **parLocaleId**: Language identifier ("DE", "EN")
- **parResultCB**: Callback function that must be provided by the caller
  This function is called by OPCGetPropertiesRequest when a response has arrived from
  the OPC XML DA server. The OPCGetPropertiesRequest object is disposed of
  automatically (by calling the "destructor" method) if the callback function supplies "true"
  as a return value.

function OPCGetPropertiesRequestCB(parResponse)
parResponse: Array of PropertyResults that contain the properties of variables:

function PropertyResult()
{
    this.mItemPath;
    this.mItemName;
    this.mResultId;
    this.mValue;
    this.mType;
    this.mAccessRights;
    this.mEnums;
}

User interface:
- addItem(parItemPath, parItemName) adds a variable to the variables list
- removeItem(parItemHandle) deletes a variable from the variables list
- sendGetPropertiesRequest() sends the read request
- destructor() releases the entire request object
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; 
charset=ISO-8859-1">
<title>GetProperties</title>
<script type="text/javascript" src="/common.js"></script>
<script type="text/javascript" src="/opcxml.js"></script>
<script type="text/javascript">
function getProperties()
{
    var tmpGetPropertiesCB = function(parResponse)
    {
        tmpResultStr = "";
        for (var tmpIndex = 0; tmpIndex < parResponse.length; 
tmpIndex++)
        {
            var tmpPropertyResult = parResponse[tmpIndex];
            var tmpEnums = "";
            if (tmpPropertyResult.mEnums && 
(deepCopy(tmpPropertyResult.mEnums.length > 0))
            {
                for (var tmpIndex = 0;
                    tmpIndex < tmpPropertyResult.mEnums.length; 
tmpIndex++)
                {
                    tmpEnums += " " +
                    tmpPropertyResult.mEnums[tmpIndex] + "\n";
                }
            }
            if (!tmpPropertyResult.mResultId)
            {
                tmpResultStr += tmpPropertyResult.mItemPath + 
"::" +
                tmpPropertyResult.mItemName + 
": \n Type = " +
                tmpPropertyResult.mType + 
"\n value = " +
                tmpPropertyResult.mValue + 
"\n AccessRights = " +
                tmpPropertyResult.mAccessRights + 
"\n Enums = \n" + tmpEnums +"\n";
            }
            else
            {
                tmpResultStr += tmpPropertyResult.mItemPath + 
"::" + tmpPropertyResult.mItemName + 
": \n ResultId = " +
                tmpPropertyResult.mResultId + 
"\n\n";
            }
        }
    }
    getProperties();
</script>
</head>
<body>
</html>
```
```javascript
var tmpGetPropertiesRequest = new OPCGetPropertiesRequest("DE",tmpGetPropertiesCB);
tmpGetPropertiesRequest.addItem("SIMOTION", "var/userdata.user1");
tmpGetPropertiesRequest.addItem("SIMOTION", "var/userdata.user20");
tmpGetPropertiesRequest.addItem("SIMOTION", "dev/Service.BZU.value");
tmpGetPropertiesRequest.sendGetPropertiesRequest();
```

**OPCWriteRequest**

OPCWriteRequest **writes three values of one or more variables.**

```javascript
function OPCWriteRequest(parLocaleId,parResultCB)

Transfer parameters:
- **parLocaleId**: Language identifier ("DE", "EN")
- **parResultCB**: Callback function that must be provided by the caller
  The function is called on conclusion of the send request.

function OPCWriteRequestCB(parResultList)

parResultList is an array of ItemValues that contains the results of the write request.

function ItemValue()

```
User interface:

- `addItem(parItemPath, parItemName, parType)` adds a variable to the variables list and returns a variable handle, which can be used to reference the variable within the request. ParType designates the OPC XML DA data type to be used for writing. If parType is not transferred, the "xsi::string" data type is applied.
- `removeItem(parItemHandle)` removes a variable from the variables list
- `setItemValue(parItemHandle, parValue)` sets the value that is to be written for a variable
- `sendWriteRequest()` sends the write request
- `destructor()` releases all resources occupied by the write object
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <script type="text/javascript" src="/common.js"></script>
  <script type="text/javascript" src="/opcxml.js"></script>
  <script type="text/javascript">
    function writeValues()
    {
      var tmpWriteCB = function(parWriteResult)
      {
        var tmpString = "";
        for (var tmpIndex = 0; tmpIndex < parWriteResult.length; tmpIndex++)
        {
          var tmpItemValue = parWriteResult[tmpIndex];
          var tmpValue = (tmpItemValue.mItemResultId) ? tmpItemValue.mItemResultId : tmpItemValue.mItemValue;
          tmpString += tmpItemValue.mItemPath + "::" + tmpItemValue.mItemName + " = " + tmpValue + "\n";
        }
        alert(tmpString);
      }
      var tmpWrite = new OPCWriteRequest("DE", tmpWriteCB);
      var tmpItemHandle = tmpWrite.addItem("SIMOTION", "var/userdata.user1");
      tmpWrite.setItemValue(tmpItemHandle, "123");
      tmpItemHandle = tmpWrite.addItem("SIMOTION", "var/userdata.user2");
      tmpWrite.setItemValue(tmpItemHandle, "234");
      tmpItemHandle = tmpWrite.addItem("SIMOTION", "var/userdata.user10");
      tmpWrite.setItemValue(tmpItemHandle, "345");
      tmpWrite.sendWriteRequest();
    }
    </script>
  <title>Write</title>
</head>
<body>
  <input type="button" value="Write" onclick="writeValues()"/>
</body>
</html>
```
OPCBrowseRequest

The OPCBrowseRequest class can be used to browse the variable management area of a control.

function OPCBrowseRequest(parLocaleId, parResultCB)

Transfer parameters:

- **parLocaleId**: Language identifier ("DE", "EN")
- **parResultCB**: Callback function that must be provided by the caller. The function is called on conclusion of the send request.

function OPCBrowseRequestCB(parResult, parItemPath, parItemName)

- **parResult** is an array of type BrowseResult and contains the Browse information.

```plaintext
define BrowseResult()
{
  mItemPath;
  mItemName;
  mName;
  mIsItem;
  mHasChildren;
}
```

- **parItemPath** and **parItemName** are the path and name of the directory whose content is displayed in BrowseResult.

User interface:

- **sendBrowseRequest(parItemPath, parItemName)** sends the browse request.
  
  **parItemPath** and **parItemName** are the path and name of the directory to be browsed.
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript">
        function browse()
        {
            var tmpBrowseRequestCB = function(parBrowseResult, parItemPath, parItemName)
            {
                var tmpString = parItemPath + '::' + parItemName + '
                for (var tmpIndex = 0; tmpIndex < parBrowseResult.length; tmpIndex++)
                {
                    var tmpBrowseResult = parBrowseResult[tmpIndex];
                    tmpString += tmpBrowseResult.mItemName + '
                }
                alert(tmpString);
            }
            var tmpBrowseRequest =
                new OPCBrowseRequest("DE", tmpBrowseRequestCB);
            tmpBrowseRequest.sendBrowseRequest("SIMOTION","var/");
        }
    </script>
    <title>Browse</title>
</head>
<body>
    <input type="button" value="Browse" onclick="browse();"/>
</body>
</html>
```
The OPCSubscriptionRequest class can be used to set up, poll, and delete an OPC XML DA subscription.

```javascript
function OPCSubscriptionRequest(parLocaleId, parResultCB, parCancelCB)
```

Transfer parameters:

- **parLocaleId**: Language identifier ("DE", "EN")
- **parResultCB**: Callback function that must be provided by the user. This is called following setup and a refresh action.

```javascript
function OPCSubscriptionRequestCB(parResultList, parResult)
parResultList is an array of type OPCItemValue, which contains the variable values which have been determined.

function OPCItemValue()
{
    mItemPath
    mItemName
    mItemHandle
    mItemValue
    mItemResultId
}
```

- **parCancelCB**: Callback function that must be provided by the user. This is called after a subscription is released.

```javascript
function OPCSubscriptionCancelCB()
The function has no transfer parameters.
```

User interface:

- **addItem(parItemPath, parItemName)** adds the transferred variable to the internal list of subscription variables
- **removeItem(parItemHandle)** deletes the transferred variable from the list of subscription variables
- **cancel()** logs out an active subscription on the server
• **refresh()** reads the current variable values. Only variables whose values have changed since the preceding query are transferred. The first call of refresh after generating the OPCSubscription object or the first call after a cancel causes the subscription to log in with the current internal variable list on the server. Refresh must be called cyclically. The hold time mechanism of the OPC XML DA subscription is not supported, i.e. a wait time must be programmed between each 2 refresh cycles by means of a JavaScript timer. However, the wait time of the OPC XML DA subscription is active, i.e. a response to a refresh call is sent only after the wait time elapses, provided a variable value has not changed in the meantime.

• **destructor()**
Logs out the subscription on the server and releases the resources occupied by the subscription object. Destructor must be called before releasing the object.
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<script type="text/javascript" src="/common.js"></script>
<script type="text/javascript" src="/opcxml.js"></script>
<script type="text/javascript">
var gloSubscription;
var gloItemHandle_1;
var gloItemHandle_2;
function subscription()
{
  if (!gloSubscription)
  {
    var tmpSubscriptionCB = function(parValues)
    {
      for (var tmpIndex = 0;
      parIndex < parValues.length;
      tmpIndex++)
      {
        var tmpItemHandle =
          parValues[tmpIndex].mItemHandle;
        var tmpItemValue =
          parValues[tmpIndex].mItemValue;
        if (tmpItemHandle == gloItemHandle_1)
        {
          var tmpValueNode =
            document.getElementById("user1");
          tmpValueNode.firstChild.nodeValue =
            tmpItemValue;
        }
      
        else if (tmpItemHandle == gloItemHandle_2)
        {
          var tmpValueNode =
            document.getElementById("user2");
          tmpValueNode.firstChild.nodeValue =
            tmpItemValue;
        }
    }
    
    var tmpTimerCB = function()
    {
      gloSubscription.refresh();
    }
    setTimeout(tmpTimerCB,300);
    
    var tmpCancelCB = function()
    {
      if (gloSubscription
```


```javascript
{  
gloSubscription.destructor();  
gloSubscription = null;  
}

gloSubscription =  
new OPCSubscriptionRequest("DE",  
tmpSubscriptionCB,tmpCancelCB);  
gloItemHandle_1 =  
gloSubscription.addItem("SIMOTION",  
    "var/userdata.user1");  
gloItemHandle_2 =  
gloSubscription.addItem("SIMOTION",  
    "var/userdata.user2");  
gloSubscription.refresh();  
}
```

```html
function cancel()  
{  
    if (gloSubscription)  
        gloSubscription.cancel();  
}
</script>

<title>Subscription</title>

<body>
<div>  
<input type="button" value="Start"  
onclick="subscription();"/>
<input type="button" value="Cancel" onclick="cancel();"/>
</div>
<table>  
<tr>  
    <td>user1</td>  
    <td id="user1">user1</td>  
</tr>  
<tr>  
    <td>user2</td>  
    <td id="user2">user2</td>  
</tr>  
</table>
</body>
</html>
```
OPCSubscriptionAutoRefresh provides the same function as OPCSubscriptionRequest, with the exception that the timer-controlled call of the refresh function occurs automatically.

```plaintext
function OPCSubscriptionAutoRefresh(parLocaleId, parResultCB, parCancelCB,
parCycleTime)
```

Transfer parameters:
- `parLocaleId`
- `parResultCB`
- `parCancelCB`
  - See OPCSubscriptionRequest
- `parCycleTime`
  - Cycle time in which the refresh function is called, in ms

User interface:
- `startRefresh()` starts the query cycle
- `cancel()` See OPCSubscriptionRequest
- `addItem(parItemPath, parItemName, parItemHandle)` See OPCSubscriptionRequest
  - After the variable is added, the refresh cycle is restarted automatically.
- `removeItem(parItemHandle)`
- `destructor()`
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=ISO-8859-1">
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript">
        var gloSubscription;
        var gloItemHandle_1;
        var gloItemHandle_2;
        var gloItemHandle_3;
        function subscription()
        {
            if (!gloSubscription)
            {
                var tmpSubscriptionCB = function(parValues)
                {
                    for (var tmpIndex = 0; tmpIndex < parValues.length;
                    tmpIndex++)
                    {
                        var tmpItemHandle =
                        parValues[tmpIndex].mItemHandle;
                        var tmpItemValue =
                        parValues[tmpIndex].mItemValue;
                        if (tmpItemHandle == gloItemHandle_1)
                        {
                            var tmpValueNode =
                            document.getElementById("user1");
                            tmpValueNode.firstChild.nodeValue =
                            tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_2)
                        {
                            var tmpValueNode =
                            document.getElementById("user2");
                            tmpValueNode.firstChild.nodeValue =
                            tmpItemValue;
                        }
                        else if (tmpItemHandle == gloItemHandle_3)
                        {
                            var tmpValueNode =
                            document.getElementById("user3");
                            tmpValueNode.firstChild.nodeValue =
                            tmpItemValue;
                        }
                    }
                }
                var tmpCancelCB = function()
            }
```

SIMOTION IT Ethernet-based HMI and Diagnostic Functions

Diagnostics Manual, 11/2010
```javascript
if (gloSubscription) {
  gloSubscription.destructor();
  gloSubscription = null;
}

if (gloSubscription)
{
  gloSubscription = new OPCSubscriptionAutoRefresh("DE",
    tmpSubscriptionCB, tmpCancelCB, 300);

gloItemHandle_1 = gloSubscription.addItem("SIMOTION",
  "var/userdata.user1");
gloItemHandle_2 = gloSubscription.addItem("SIMOTION",
  "var/userdata.user2");
gloSubscription.startRefresh();
}

function addVar()
{
  if (gloSubscription)
  {
    gloItemHandle_3 = gloSubscription.addItem("SIMOTION",
      "var/userdata.user3");
  }
}

function removeVar()
{
  if (gloSubscription)
  {
    gloSubscription.removeItem(gloItemHandle_3);
  }
}

function cancel()
{
  if (gloSubscription)
  {
    gloSubscription.cancel();
  }
}
```

```html
<title>Auto refresh</title>
</head>
<body>
</div>
<input type="button" value="Start"
onclick="subscription();"/>
<input type="button" value="Add variable"
onclick="addVar();"/>
<input type="button" value="Remove variable"
onclick="removeVar();"/>
<input type="button" value="Cancel" onclick="cancel();"/>
```
3.5 User-defined pages

</div>
<table>
<tr>
<td>user1</td>
<td id="user1">user1</td>
</tr>
<tr>
<td>user2</td>
<td id="user2">user2</td>
</tr>
<tr>
<td>user3</td>
<td id="user3">user3</td>
</tr>
</table>
</body>
</html>
3.5.6.3 Representation of OPC XML DA data in the browser (appl.js)

The JavaScript library appl.js contains a collection of classes for representing the data determined with OPC XML DA requests.

ApplDataTable

ApplDataTable implements a dynamic table in which process variables can be displayed. The variable values are updated cyclically with an OPC XML DA subscription.
function 
ApplDataTable(parDocument, parClassName, parColumnClasses, parColumnIds , parHeader)

Transfer parameters:
- parDocument: JavaScript document used to generate elements
- parClassName: Entered as "class" attribute in the "Table" tag of the HTML table
- parColumnClasses: Array whose "length" attribute specifies the number of table columns. The values of the array are used as a "class" attribute for the table columns (<td class="...">
- parColumnIds: Array whose values are used together with the parRowId parameter (described below) for the "id" attributes of the table columns. The value of the "id" attribute is produced by chaining together the ColumnId and RowId (in that order).
- parHeader: Array containing the column headings of the table

The transfer parameters are used to set the "class" and "id" attributes such that the table display can be specified using style sheets.

User interface:
- addRow(parRowId, parRowClass) appends a new row to the table
  - parRowId: Used as a value for the id attribute of row (<tr id="...">
  - parRowClass: Used as a value for the "class" attribute of row (<tr class="...">
- addElement(parElement, parDestructor, parColSpan, parColClass)
  Inserts the HTML element transferred using parElement in the table.
  - parElement: HTML element to be inserted in the table
  - parDestructor (optional)
    : Function that is called if the HTML element is deleted from the table (used in Internet Explorer to prevent memory leaks).
  - parColSpan (optional)
    : Specifies the number of columns the element is to span.
  - parColClass (optional)
    parColClass can be used to overwrite the ColClass assigned when the ApplDataTable object is created, if special formatting is to be applied for the current element.
• addVariable(parPath,parName,parColSpan,parColClass): Inserts a variable in the table. The value of the variables is updated cyclically.
  – parPath
    Variable path (e.g. "SIMOTION" or "SIMOTION diagnostics")
  – parName
    Variable name (e.g. "var/userdata.user1")
  – parColSpan (optional)
    See addElement
  – parColClass (optional)
    See addElement

• addText(parText,parColSpan,parColClass): Inserts text in the table.
  – parText
    : Text to be inserted.
  – parColSpan (optional)
    See addElement
  – parColClass (optional)
    See addElement

• addRemoveButton (parRemoveCB,parRemoveData,parColSpan,parColClass)
  Inserts a button in the table for removing the current row.
  – parRemoveCB(parData) (optional)
    : Callback function that is called if the row is deleted.
  – parData
    Data transferred when addRemoveButton was called
  – parRemoveData (optional)
    Data transferred when parRemoveCB is called as a parameter
  – parColSpan (optional)
    See addElement
  – parColClass (optional)
    See addElement

• addImage(parImage,parColSpan,parColClass): Inserts an image in the table.
  – parImage: URL of the image to be inserted
  – parColSpan (optional), see addElement
  – parColClass (optional), see addElement
3.5 User-defined pages

- getVariables() supplies an array of all variables of the table. Each entry of the array consists of an object with the elements mItemPath, mItemName, and mItemHandle.

- addRefreshCB(parRefreshCB) registers a Callback function on the table object, which is called on completion of a refresh cycle; an array of OPCItemValue-Objekten is transferred to the Callback function.

```cpp
OPCItemValue
{
    mItemPath
    mItemName
    mItemHandle
    mItemValue
    mItemResultId
} destructor()
```

This function must be called if the table is no longer required. This function releases all the resources occupied by the table. In particular, the subscription used by the table is logged off on the OPC XML DA server. The function must also be called when the page is exited (onunload).
Example:

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <script type="text/javascript" src="/common.js"></script>
    <script type="text/javascript" src="/opcxml.js"></script>
    <script type="text/javascript" src="/appl.js"></script>
    <script type="text/javascript">
      var gloApplDataTable = null;
      function init()
      {
        if (gloApplDataTable == null)
        {
          var tmpColumnIds  = new Array("Name_0","Value_0",
                                      "Name_1","Value_1");
          var tmpColumnClasses = new Array("Name","Value",
                                           "Name","Value")
          var tmpHeader        = new Array("Name","Value",
                                           "Name","Value");
          gloApplDataTable = new ApplDataTable(document,
                                                 "ReadTableClass",
                                                 tmpColumnClasses,
                                                 tmpColumnIds,
                                                 tmpHeader);
          gloApplDataTable.addRow("Row_0","RowClass_0");
          gloApplDataTable.addText("user1");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user1");
          gloApplDataTable.addRow("Row_1","RowClass_1");
          gloApplDataTable.addText("user2");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user2");
          gloApplDataTable.addRow("Row_2","RowClass_0");
          gloApplDataTable.addText("user3");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user3");
          gloApplDataTable.addRow("Row_3","RowClass_1");
          gloApplDataTable.addText("user4");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user4");
          gloApplDataTable.addRow("Row_4","RowClass_0");
          gloApplDataTable.addText("user5");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user5");
          gloApplDataTable.addRow("Row_5","RowClass_1");
          gloApplDataTable.addText("user6");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user6");
          gloApplDataTable.addRow("Row_6","RowClass_0");
          gloApplDataTable.addText("user7");
          gloApplDataTable.addVariable("SIMOTION",
                                        "var/userdata.user7");
        }
      }
    </script>
  </head>
</html>
```
In the final section of the source code, the <style> tag demonstrates how the colors of table rows and individual table cells can be changed using CSS formatting.

This means that the tr.RowClass_0 declaration, for example, can ensure that a yellow background is produced when gloApplDataTable.addRow("Row_0","RowClass_0"). is called.
ApplBrowser enables the variable management area of the control to be queried.

After creating an object of this type, the first browse operation is started automatically. If the browse information of a subtree has been received, the callback function \texttt{parNewTreeFct} is first called. Afterwards, the callback function \texttt{parNewNodeFct} or \texttt{parNewLeafFct} is called, depending on the type of information (node or leaf). All callback functions receive an object of type \texttt{ApplBrowseElement} as the first parameter.

User interface \texttt{ApplBrowseElement}:

- \texttt{getElement()} supplies an HTML anchor object (<a ...>) for displaying information
- \texttt{setCB(parCB)}: Registers a Callback function at the \texttt{ApplBrowseElement}-Objekt; this function is called if a user clicks the anchor object. This function also receives an \texttt{ApplBrowseElement}-Objekt as a transfer parameter.
- \texttt{destructor()} must be called if the object is no longer required. This also applies to exiting the page (\texttt{onunload}).

\texttt{ApplBrowser(parDocument, parItemPath, parItemName, parNewTreeFct, parNewNodeFct, parNewLeafFct)}

Transfer parameters:

- \texttt{parDocument}: JavaScript document used to generate elements
- \texttt{parItemPath, parItemName} specifies the starting point for browsing the variable management area
- \texttt{parNewTreeFct(parBackElement, parItemPath, parItemName)}: Callback function that is called when the structure of a new subtree begins.
  - \texttt{parBackElement} object of type \texttt{ApplBrowseElement}  
    The content describes the start node. If a user clicks the HTML anchor of this object, the elements of the preceding subtree (if present) are determined.
  - \texttt{parItemPath, parItemName}: Path and name of the current subtree
- \texttt{parNewNodeFct(parBrowseElement)}: Callback function that is called for a node element
  - \texttt{parBrowseElement} object of type \texttt{ApplBrowseElement}  
    The content describes a node. If a user clicks the HTML anchor of this object, a browse operation is started for the subtree connected to this node.
- \texttt{parNewLeafFct(parBrowseElement)} Callback function that is called for a leaf element
  - \texttt{parBrowseElement} object of type \texttt{ApplBrowseElement}  
    The content describes a leaf.
3.5 User-defined pages

User interface:

- destructor releases all resources occupied by the browser
Example:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" 
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Browser demo</title>
<script type="text/javascript" src="/common.js"></script>
<script type="text/javascript" src="/opcxml.js"></script>
<script type="text/javascript" src="/appl.js"></script>
<script type="text/javascript">
var gloBrowseTable = null;
var gloBrowser = null;
function addItem(parBrowseElement)
{
    var tmpBodyElement = gloBrowseTable.firstChild;
    var tmpTableRowElement = document.createElement("tr");
    var tmpTableDataElement = document.createElement("td");
    var tmpLinkElement = parBrowseElement.getElement();
    tmpTableDataElement.appendChild(parBrowseElement.getElement());
    tmpTableRowElement.appendChild(tmpTableDataElement);
    tmpBodyElement.appendChild(tmpTableRowElement);
}
function browse()
{
    var tmpNewTreeFct = function(parBrowseElement,parItemPath,parItemName)
    {
        var tmpBrowseTableHook = document.getElementById("BrowseTable");
        if (tmpBrowseTableHook.firstChild)
            tmpBrowseTableHook.removeChild(tmpBrowseTableHook.firstChild);
        gloBrowseTable = document.createElement("table");
        var tmpBodyElement = document.createElement("tbody");
        gloBrowseTable.appendChild(tmpBodyElement);
        tmpBrowseTableHook.appendChild(gloBrowseTable);
        if (parBrowseElement)
            {parBrowseElement.getElement().firstChild.nodeValue = "< "+
                addItem(parBrowseElement);
            alert("Path: " + parItemPath + "\nName: " + parItemName);
        }
    }
    var tmpNewNodeFct = function(parBrowseElement)
    {
        parBrowseElement.getElement().firstChild.nodeValue = "+ " +
            addItem(parBrowseElement);
    }
    var tmpNewTreeFct = function(parBrowseElement)
    {
        parBrowseElement.getElement().firstChild.nodeValue = "< " +
            addItem(parBrowseElement);
    }
    alert("Path: " + parItemPath + "\nName: " + parItemName);
    }
); var tmpNewNodeFct = function(parBrowseElement)
    {
        parBrowseElement.getElement().firstChild.nodeValue = "+ " +
            addItem(parBrowseElement);
    }
```
```
var tmpNewLeafFct = function(parBrowseElement)
{
    var tmpCB = function(parBrowseElement)
    {
        var tmpText = "Path: " +
                parBrowseElement.mItemPath + "\n" +
        "Name: " +
                parBrowseElement.mItemName + "\n";
        alert(tmpText);
    }
    parBrowseElement.setCB(tmpCB);
    addItem(parBrowseElement);
}

gloBrowser = new ApplBrowser(document, ",", ",",
        tmpNewTreeFct,
        tmpNewNodeFct,
        tmpNewLeafFct);

document.write(gloBrowser);

gloBrowser.destructor();
</script>

<body onload="browse();" onunload="leave();">

</body>

</html>
ApplBrowseTree

ApplBrowseTree builds upon the ApplBrowser and displays the browse result in tree format.

ApplBrowseTree(parDocument,
    parItemPath,
    parItemName,
    parTablePrefix,
    parLeafCB,
    parNodeCB,
    parBackCB,
    parFilterCB,
    parLeafImg,
    parNodeImg,
    parBackImg)

Transfer parameters:
- parDocument JavaScript document used to generate elements
- parItemPath, parItemName : Specifies the starting point for browsing the variable management.
- parTablePrefix Prefix for referencing elements in CSS
- parLeafCB Callback- function that is called if a leaf is clicked
- parNodeCB : Callback function that is called if a node is clicked. Each of the callback functions receives an ApplBrowseElement as a first parameter.
- parBackCB Callback- function that is called if the first element of the tree is clicked
- parLeafImg, parNodeImg, parBackImg : Symbols that are displayed by a back, node, or leaf element.

User interface:
- getElement() : Supplies a table element that contains the browse results.
- destructor() Releases the table and all resources connected to it
Example:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Browse table demo</title>
  <style type="text/css">
    table.BrowseTree {table-layout:fixed;}
    td.BrowseTreeImg0 {width:20px;}
    td.BrowseTreeImg1 {width:20px;}
    td.BrowseTreeBrowse {overflow:visible;text-align:left;}
    a.refLeaf {cursor:pointer;}
    a.refNode {cursor:pointer;}
  </style>
  <script type="text/javascript" src="/common.js"></script>
  <script type="text/javascript" src="/opcxml.js"></script>
  <script type="text/javascript" src="/appl.js"></script>
  <script type="text/javascript">
    var gloBrowseTree = null;
    function browse()
    {
      var tmpLeafCB = function(parBrowseElement)
      {
        var tmpText = "Path: " + parBrowseElement.mItemPath + "\n" + "Name: " + parBrowseElement.mItemName + "\n";
        alert(tmpText);
      }
      var tmpFilterCB = function(parBrowseElement)
      {
        var tmpItemName = parBrowseElement.mItemName;
        var tmpPos = tmpItemName.indexOf("unit/");
        if (tmpPos != 0)
        {
          tmpPos = tmpItemName.indexOf("to/");
          if (tmpPos != 0)
            tmpPos = tmpItemName.indexOf("var/");
        }
        return (tmpPos == 0);
      }
      gloBrowseTree = new ApplBrowseTree(document,
        "SIMOTION",
        ",",
        "BrowseTree",
        tmpLeafCB,
        undefined,
        undefined,
        tmpFilterCB,
        "/ledred.gif",
        "/ledgreen.gif",
        "/ledaqua.gif");
    }
```

var tmpBrowseTreeHook =
    document.getElementById("BrowseTree");
tmpBrowseTreeHook.appendChild(gloBrowseTree.getElement());
}

function leave()
{
    if (gloBrowseTree)
        gloBrowseTree.destructor();
}
</script>
</html>
3.5 User-defined pages

3.5.7 MiniWeb Server Language (MWSL)

3.5.7.1 Mode of operation of the MWSL

The web server language is a script language that is interpreted on the web server. It is rather similar to the JavaScript language but represents a small subset of the language scope.

The MWSL enables the client to be operated with a simple browser without scripting, because the web server generates the pages dynamically.

MWSL enables access and processing of variables. Among other things, it allows access to process variables that are present on the web server system. MWSL and the integrated template mechanism can then be used very effectively to process and evaluate these variables.

The template mechanism used to produce the dynamic pages is similar to a very simplified XSLT process. See XSL transformation ().

The client requests a URL on the web server.

An MWSL file is located there. From this file, the MWSL service generates a temporary HTML file on the web server. This file is then sent to the client and displayed there.
3.5 User-defined pages

3.5.7.2 Structure of a MWSL file

An MWSL file is basically an HTML file which also contains MWSL tags. To distinguish them from HTML files, MWSL files have the extension ".mcs". Converting standard HTML pages to binary files (Page 119)

Example:

```html
<HTML>
  <HEAD>
    ...
  </HEAD>
  <BODY>
    <table>
      <tr>
        [...] 
        <td>
          <MWSL>
            <!-- 
            //MWSL code to be executed 
            -->
          </MWSL>
        </td>
        [...] 
        <td>
          <MWSL>
            <!-- 
            //MWSL code to be executed 
            -->
          </MWSL>
        </td>
        [...] 
      </tr>
    </table>
  </BODY>
</HTML>
```

If the MWSL functionality is needed, you must add the following tags:

- The `<MWSL>`-tag introduces an MWSL script
- The `</MWSL>`-tag marks the end of the script

The HTML comment characters that follow the `<MWSL>`-tag are not mandatory but are, nevertheless, recommended because they protect the MWSL code from the HTML interpreter, thus preventing incorrect outputs.

For reasons of clarity, the examples below do not always include the HTML code and begin directly with the "<MWSL>" tag.
3.5 User-defined pages

3.5.7.3 Error messages

MWSL error messages

MWSL pages with faulty MWSL operations feature a comment containing information about the cause of the error.

The source text of a page can be loaded to the editor in the Internet Explorer by right-clicking with the mouse and selecting the menu command Show Source Text.

Example

This example shows the comment associated with the query relating to an unavailable variable.

exec.mcs
<html>
<head>
<title>SIMOTION</title>
<meta name="DC.Subject" content="SIMOTION">
<meta name="DC.Publisher" content="Siemens AG">
<meta name="DC.Format" content="text/html">
<meta name="DC.Language" content="en">
<meta name="DC.Rights" content="Copyrights Siemens AG 2003">
</head>
<body style="font-family: Arial">
<p>
<MWSL>WriteVar("var/userData.user8");</MWSL>
</p>
<p>
<MWSL>WriteVar("var/userData.user9");</MWSL>
</p>
</body>
</html>

In file exec.mcs the statement WriteVar("var/userData.user9"); is used to query a non-existent variable.
3.5.7.4 Variable types

MWSL distinguishes between script variables and global variables:

- Script variables are defined within the script
- Global variables are provided by variable sources

Note

Global variables are not part of the script engine. Rather, they represent information from the web server environment. Variables are accessed exclusively via access functions. The global variables are grouped into variable sources according to their origin.
3.5.7.5 **Script variables**

These variables are only valid on the current page.

The variables apply beyond MWSL tags, i.e. they can be created in one MWSL tag and be used starting in the next MWSL tag.

For these variables, no distinction is made between variable types, i.e. there is no Int, Char, etc.

A variable is simply created with

```
var <VariableName> = <Value>;
```

The variable type is determined internally by the variable assignment.

Example:

```mws
<!--
var string1 = "Hello";
var string2 = "World";
write(string1 + " " + string2);
-->
</MWSL>
```

Two variables are created in the example shown above: `string1` and `string2`.
The two strings are strung together (with spaces).
The `write` command outputs the result (see `write()`).
Output: Hello World

Example:

```mws
<!--
var num1 = 5;
var num2 = 7;
var Result;
Result = num1 + num2;
-->
</MWSL>
```

Two variables are created in the example shown above: `num1` and `num2`.
The two numbers are added, and the result is stored in the `Result` variable.
`Result` contains the value 12.
The data type is converted in the same way as ECMA Script 262. See List of abbreviations (Page 265).
Keyword var
The keyword var introduces a variable declaration. In ECMA script, variables do not have to be explicitly declared.

Syntax:

```
var VarName = InitialValue, VarName2 = InitialValue2, ...;
```

Multiple variables are declared and (optionally) initialized with initial values.
You can specify several declarations, separated by commas.

For further information, please refer to the ECMA script definitions.

Ranges of visibility and validity
The visibility and validity of variables is analogous to ECMA script. (However, MWSL currently does not recognize any functions.)

Example:
```
<SWL>
Var MyVar = 10;
{
    MyVar = 20;
    Write ("Inner:" + MyVar + "," );
}
Write ("Outer:" + MyVar + 
"");
</SWL>
```

Output: Inner: 20, Outer: 20
In this example, the variable MyVar of the outer level will be accessed in the operation block, because a variable named MyVar was not declared on the operation block level.
Therefore, the MyVar = 20 operation changes the value of the variable on the outer level.

```
<SWL>
Var MyVar = 10;
{
    Var MyVar = 20;
    Write ("Inner:" + MyVar + "," );
}
Write ("Outer:" + MyVar + 
"");
</SWL>
```

Output: Inner: 20, Outer: 10
Global variables

Global variables enable access to the variable management area of the web server. There are three types of global variables:

- **PROCESS** variables enable access to normal variables of the web server. This is known as standard access.
- **URL** variables provide access to variables contained in a URL.
- **HTTP** variables return the content of variables in the HTTP header.

For test purposes, these variables can also be created by the VarSimulator (variable simulator).

A variable can be accessed using the following command:

```c
GetVar("Color", "PROCESS");
```

The variable source **PROCESS** must be written in upper case. If the **Color** variable is not present, "null" is returned.

**PROCESS** is the standard variable source. Therefore, **PROCESS** can also be omitted.

Example:

```c
GetVar("Color");
```

If a variable provider is to be addressed directly, the name of the desired provider can be specified instead of the variable source **PROCESS**.

Format string for the GetVar and WriteVar functions

The format string always starts with a `%` sign, followed by the specification of the number of characters. This is then followed by the type information.
The following type information is available:

- %d for Integer values
- %f for Float values
- %s for Strings

Example:

- "%.6s" Outputs the first 6 characters of the specified variable (as a string).
- "%.3s" Outputs the first 3 characters of the specified variable (as a string).
- "%s" Outputs the complete variable (as a string).
- "%3.2f" Outputs the variable interpreted as Float. The 3 indicates that 3 total places are output. The 2 indicates that, of the 3 places, 2 places after the decimal point will be displayed.
- "%4d" Outputs the variable interpreted as Integer. Four places are output. This parameter can only be transferred if the variable source "PROCESS" has also been transferred.

If the format string is omitted, the complete variable content is returned.

See also

GetVar (Page 290)
WriteVar (Page 296)
3.5.7.7 Configuration constants

Accessing constants of the WebCfg.xml configuration file

It is possible to create constant variables in the WebCfg.xml file. Accessing these constants enables more flexible programming, meaning that user-defined pages can be controlled using relevant configuration parameters.

A constant is defined in the following section of WebCfg.xml:

```xml
/SERVERPAGES/CONFIGURATION_DATA/UserConfig

<SERVERPAGES>
  <CONFIGURATION_DATA>
    <USERCONFIG>
      <MyParam>MyParamValue</MyParam>
    </USERCONFIG>
  </CONFIGURATION_DATA>
</SERVERPAGES>
```

It is possible to access the `MyParam` constant in an HTML page by specifying the "constants/MyParam" path.

```html
<html>
<head>
</head>
<body style="font-family: Arial">
<table width="100%" height="100%" bgcolor="#00349A">
<tr>
  <td style="color: #FFFFFF">
    <b><MWSL>WriteVar("constants/MyParam");</MWSL></b>
  </td>
</tr>
</table>
</body>
</html>
```
3.5.7.8 Variables and URL parameters

MWSL offers the option of editing URL parameter values using the WriteVar, GetVar, SetVar, and ExistVariable functions.

Example of a URL with appended parameters. The line break has been added for better readability:

http://localhost/MWSL/StringOperationtest.mcs?
Parameter1=Hallo&Parameter2=du!&StartValue=2&EndValue=5

The URL displays StringOperationtest.mcs on the page and transfers the Parameter1, Parameter2, StartValue, and EndValue parameters.

To output the URL variable Parameter1, the following command would be specified, for example:

WriteVar("Parameter1", "URL");

Note that "URL" must be written in upper case.

If a URL Variable that is not present in the URL is requested, an empty string ("") is always returned. This is not a script error.
Parameters in URLs

In a URL, the parameter transfer begins after the "?" character. Individual parameters are separated by "&" characters. The value is assigned after the "=" character.

Certain characters require a coding in order to be transferred correctly. The following table provides an overview of the most commonly used escape codes.

Table 3-7   URL escape codes

<table>
<thead>
<tr>
<th>Character</th>
<th>Escape code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>%20</td>
</tr>
<tr>
<td>&lt;</td>
<td>%3C</td>
</tr>
<tr>
<td>&gt;</td>
<td>%3E</td>
</tr>
<tr>
<td>#</td>
<td>%23</td>
</tr>
<tr>
<td>%</td>
<td>%25</td>
</tr>
<tr>
<td>{</td>
<td>%7B</td>
</tr>
<tr>
<td>}</td>
<td>%7D</td>
</tr>
<tr>
<td>[</td>
<td>%5B</td>
</tr>
<tr>
<td>]</td>
<td>%5D</td>
</tr>
<tr>
<td>;</td>
<td>%3B</td>
</tr>
<tr>
<td>/</td>
<td>%2F</td>
</tr>
<tr>
<td>?</td>
<td>%3F</td>
</tr>
<tr>
<td>:</td>
<td>%3A</td>
</tr>
<tr>
<td>@</td>
<td>%40</td>
</tr>
<tr>
<td>=</td>
<td>%3D</td>
</tr>
<tr>
<td>&amp;</td>
<td>%26</td>
</tr>
<tr>
<td>$</td>
<td>%24</td>
</tr>
</tbody>
</table>

The coding consists of the "%" character followed by the ASCII hexadecimal value of the desired character.

Additional information: SELFHTML ()
3.5.7.9 COOKIES

The following example shows how a cookie can be created in an MWSL file.

For this purpose, for example, the following META tag is inserted in the HEAD:

```
http-equiv="SET-COOKIE" content="siemens_automation_language=de;"
```

Example:

```html
<HTML>
  <HEAD>
    <META http-equiv="SET-COOKIE"
          content="siemens_automation_language=de;">.
  </HEAD>
  <BODY>
    [...] 
  </BODY>
</HTML>
```

For more information on cookies: http://de.selfhtml.org/

Setting a cookie as an HTTP header

It is possible to set HTTP header for the http response from MWSL. For example, this allows a cookie to be set via the HTTP header and not as a META tag.

Example:

```mws
<!--
var strCookie;
strCookie = "Set-cookie: siemens_automation_language=";
strCookie = strCookie + GetVar("Language", "URL");
strCookie = strCookie + ", path=/\n"
AddHTTPHeader( strCookie );
-->
</MWSL>
```

In this example, a cookie for detection of the language setting is set.
### 3.5 User-defined pages

#### 3.5.7.10 Variables and access to COOKIES

The access occurs similarly as for the URL parameters. The single difference is that the variable type is now `COOKIE` instead of `URL`.

The variable specified in the example can be accessed with the following command, for example:

```plaintext
GetVar("siemens_automation_language", "COOKIE");
```

Note that `COOKIE` must be written in upper case.

If the `COOKIE siemens_automation_language` is set with `en`, for example, the above call would return this value.

For more information on cookies: [http://de.selfhtml.org/](http://de.selfhtml.org/)

If the variable is not present, there is no output.

#### 3.5.7.11 Variables and HTTP header information

The head tag of an HTML page can contain a wide range of general information. The MWSL functions `GetVar` and `WriteVar` can be used to read and write in the header.

Example:

```plaintext
HTTP/1.1 GET URL\R\N
HEADER1: WERT1\R\N
HEADER2: WERT2\R\N
[...]
\R\N
```

`GetVar("HEADER1", "HTTP")` returns `VALUE1`.

HTTP HEADERS can be generated via HTML meta tags.

Example:

```html
<HTML>
  <HEAD>
    [...]  
    <META http-equiv="Accept-Language" content="de"> 
    [...]  
  </HEAD>
  <BODY>
    [...]  
  </BODY>
</HTML>
```
In this example, the HTTP variable `Accept-Language` is defined in the META tag using the "http-equiv="Accept-Language"" command. It is initialized by the `content` attribute with the value `de`.

Additional information on META information:
http://de.selfhtml.org/html/kopfdaten/meta.htm#allgemeines

This variable is accessed similarly as for the URL parameters.
The difference is that the variable source is `HTTP` and not `URL`.
The variable specified in the example can be accessed with the following command:

```java
GetVar("Accept-Language", "HTTP");
```

The variable source `HTTP` must be written in upper case.

Format string for the `GetVar` and `WriteVar` functions

The starting character and the number of characters of the variable source to be returned is specified.

This parameter can only be transferred if the variable source `HTTP` has also been transferred.
If the format string is omitted, the complete variable content is returned.

See also

Global variables (Page 164)
3.5 User-defined pages

3.5.7.12 Operators

MWSL operators

All operators presented here behave as defined in ECMA 262. For more information, refer to the ECMA 262 Specification.

Boolean values are converted to numerical values 0 (for FALSE) and 1 (for TRUE), where applicable.

Table 3-8 Relational operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>This operator returns TRUE if the left variable is less than the right variable. Otherwise the return value is FALSE.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>This operator returns TRUE if the left variable is less than or equal to the right variable. Otherwise the return value is FALSE.</td>
</tr>
<tr>
<td>&gt;</td>
<td>This operator returns TRUE if the left variable is greater than the right variable. Otherwise the return value is FALSE.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>This operator returns TRUE if the left variable is greater than or equal to the right variable. Otherwise the return value is FALSE.</td>
</tr>
<tr>
<td>==</td>
<td>This operator returns TRUE if the left variable is equal to the right variable. Otherwise the return value is FALSE.</td>
</tr>
</tbody>
</table>

Table 3-9 Logic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Logical NOT&lt;br&gt;This operator returns FALSE if the subsequent parameter is TRUE. If the subsequent parameter is FALSE, the return value of the operator is TRUE.</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND&lt;br&gt;This operator returns TRUE if a TRUE value is present on the left side and right side. Otherwise the return value is FALSE.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3-10 Arithmetic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Plus operator&lt;br&gt;This operator adds the left and right variables together.</td>
</tr>
<tr>
<td>=</td>
<td>Assignment operator&lt;br&gt;This operator assigns the value of the right variable to the left variable.</td>
</tr>
</tbody>
</table>
### Operator Comments

<table>
<thead>
<tr>
<th>Operator</th>
<th>Comment</th>
</tr>
</thead>
</table>
| −        | Minus operator  
This operator subtracts the value of the right variable from the left variable. |
| /        | This operator divides the left variable by the right variable. The return value is an integer. |
| %        | Modulo  
This operator returns the remainder of a division. |
| *        | This operator multiplies the left variable times the right variable. |
| ++       | Increment  
This operator increments (+1) the prefixed variable. |
| --       | Decrement  
This operator decrements (-1) the prefixed variable. |

#### 3.5.7.13 For

The MWSL provides a loop mechanism, such as is already familiar from JavaScript. For a detailed description, refer to the ECMA 262 Specification.

**Syntax**

```plaintext
for( Start statement; End condition; Run statement)
{
    Loop body, Code to be executed
}
```

**Sequence:**

1. The start instruction is executed
2. The loop body is executed
3. The run operation is executed
4. As long as the end condition is true, the processing is repeated starting from the loop body (2.).

**Example:**

```plaintext
for(i=1; i<5; i++)
{
    write(i);
}
```

Output: 1234
### 3.5.7.14 If

The `if` condition is familiar from Ecma 262.

**Syntax**

```
if(<condition>)
    Statement1
else
    Statement2;
```

If the condition is true, instruction 1 is run.

If the condition is false, instruction 2 is run. If no `else` part is present, processing continues in accordance with the `if` operation.

**Example:**

```
<WSL>
<!--
[...]
if (ExistVariable("Parameter", "PROCESS"))
{
    WriteVar("Parameter");
}
[...]
-->;
</WSL>
```

If the `Parameter` process variable exists, its content is output.

If not, the instruction is skipped and the program execution is then resumed. In the example above, this would be the code that follows after the closing curly bracket.

If an `else` branch is present, it is run provided that the condition has not been fulfilled.
Example:

```xml
<MWSL>
  <!—-
  [...]  
  if (ExistVariable("Parameter")
  { 
    WriteVar("Parameter");
  }
  else
  { 
    write("Parameter process variable is not present");
  }
  [...]  
  -->
</MWSL>
```

If the Parameter variable is not present, the else part is executed. A message is then output, indicating that no variable with the specified name exists.

As the examples show, an operation can be replaced by an operation block. An operation block is a list of operations that is enclosed in curly brackets.

Example:

```xml
<MWSL>
  <!—-
  [...]  
  if (ExistVariable("Parameter") && GetVar("Parameter")>=3)
  { 
    WriteVar("Parameter");
  }
  else
  { 
    write("Parameter process variable is not permitted or is not available");
  }
  [...]  
  -->
</MWSL>
```

If the Parameter process variable exists and the content is greater than or equal to 3, it is output. Otherwise, a corresponding output is made.
3.5.7.15 Overview of MWSL functions

The MWSL provides a variety of functions, which are presented in the following overview table. For detailed descriptions of the functions, refer to the appendix.

Table 3-11 MWSL functions

<table>
<thead>
<tr>
<th>Function name</th>
<th>Parameters</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddHTTPHeader()</td>
<td>&lt;Http Header&gt;</td>
<td>Insert &lt;Http Header&gt; in a page.</td>
</tr>
<tr>
<td>CacheVar()</td>
<td>&lt;Variable&gt;</td>
<td>Insert &lt;Variable&gt; in the cache memory.</td>
</tr>
<tr>
<td>ExistVariable()</td>
<td>&lt;Variable Name&gt;, &lt;Variable Source&gt;</td>
<td>Query of the existence of a variable.</td>
</tr>
<tr>
<td>GetVar()</td>
<td>&lt;Variable Name&gt;, &lt;Variable Source&gt;, &lt;Format String&gt;</td>
<td>Return of the value of a variable of the corresponding variable source</td>
</tr>
<tr>
<td>InsertFile()</td>
<td>&lt;Text File&gt;</td>
<td>Import of a &lt;Text File&gt;. A path can be specified.</td>
</tr>
<tr>
<td>ProcessXMLData()</td>
<td>&lt;DATA&gt;, &lt;TEMPLATE&gt;</td>
<td>Generation of dynamic HTML files with special XML files.</td>
</tr>
<tr>
<td>SetVar()</td>
<td>&lt;Variable Name&gt;, &lt;Value&gt;</td>
<td>Sets values of variables.</td>
</tr>
<tr>
<td>ShareRealm()</td>
<td>&lt;Group Name&gt;</td>
<td>Indicates whether the current user is a member of the group transferred as a parameter. The return value can be TRUE or FALSE.</td>
</tr>
<tr>
<td>write</td>
<td>&lt;Text&gt;</td>
<td>Writes &lt;Text&gt; strings to the HTML page. &lt;Text&gt; can also be the return value of functions.</td>
</tr>
<tr>
<td>WriteVar</td>
<td>&lt;Variable Name&gt;, &lt;Variable Source&gt;, &lt;Format String&gt;</td>
<td>Output of a variable value. The syntax is identical to the GetVar() function.</td>
</tr>
<tr>
<td>WriteXMLData</td>
<td>&lt;DATA&gt;, &lt;TEMPLATE&gt;</td>
<td>Outputs the data directly in contrast to ProcessXMLData().</td>
</tr>
</tbody>
</table>

Table 3-12 MWSL process variables

<table>
<thead>
<tr>
<th>Process variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeIndex (Page 299)</td>
<td>Parse process variable for the template. This variable outputs the number of nodes that have already been run through.</td>
</tr>
<tr>
<td>NodeLevel (Page 299)</td>
<td>Parse process variable for the template. This variable outputs the hierarchy level of the current node.</td>
</tr>
</tbody>
</table>
3.5.7.16 Mode of operation of the template mechanism

A Template is applied to data elements of a data source. This mechanism allows separate implementation of data and processing.

The Template mechanism is started by the `ProcessXMLData()` and `WriteXMLData()` commands.

![Diagram](image)

The data file contains structured data that can be output as transformed data using the Template mechanism.

An XML file consists of a series of XML nodes. A name as well as a set of attributes are assigned to each XML node. An attribute, in turn, consists of a name and a value.

The Template file contains a set of transformation operations.

A transformation operation can be assigned to an XML node (with a certain name).

During the transformation process, the data fragment is read node-by-node from top to bottom. If a Template can be assigned to a node, this Template is executed and the attributes of the current node are available to the Template as variables.
3.5.7.17 Structure of the template file

The Template file is an XML file whose data nodes contain the transformation operations for processing a data file.

Example
```xml
<?xml version="1.0" ?>
<TEMPLATES>
    [...]
    <TEMPLATE NAME="Variable">
        [...]
        <POSITION NAME="LINE">
            <![CDATA[
                [...]
                <MWSL>Name()</MWSL>
                [...]
            ]]> 
        </POSITION>
        [...]
    </TEMPLATE>
    [...]
</TEMPLATES>
```

The Template file evaluates the data nodes of the data file.

The individual Template tags are defined in the `<TEMPLATES>` tag (in the example above, only one template tag is defined).

The line `<TEMPLATE NAME="Variable">` defines that this Template (the subsequent code) is only run for data notes of the "Variable" type. The type of a data node is specified either through the name of the data node or through a special attribute named "Template".

```
<POSITION NAME="LINE">
    [...]
</POSITION>
```

This tag is used to specify the area of the web page in which the content of the template is to be output.

The positions `HEAD`, `LINE`, and `FOOT` are available and are run in that order during processing.

The content of the `POSITION` is encapsulated in a `CDATA`-block in order to protect it from the XML parser:
```
<![CDATA[ [...]]]>
```

MWSL can now be used to access the attributes of the current data node and to output them or use them in other operations.
Example:

```html
<MWSL> WriteVar("Name") </MWSL>
```

### 3.5.7.18 Structure of a data source

A data source is an XML fragment, in which the nodes of the XML fragment form the data elements.

**Note**

If the XML fragment does not contain a root node, MWSL itself generates a root node so that the data source conforms to XML.

The XML fragment can also be a complete XML document.

Example:

```xml
<?xml version="1.0" standalone="yes"?>
[...]
<Variable Name="ZUFUEHRUNG.STATE"
Type="String"
InitialValue="good"
Behavior="Manual"
Description="Status of part infeed."
/>
[...]
<Variable Name="Language"
Type="String"
InitialValue="de_DE"
Behavior="Manual"
Description="Language setting of web page"
/>
[...]
```

The example defines data nodes (in the example: `Variable`) with the associated attributes. The different nodes can be evaluated using a corresponding template file. A further option is the creation of a data structure (hierarchy). That is, data nodes can be created that, in turn, contain other data nodes.
Example:

```xml
<?xml version="1.0" standalone="yes"?>
 [...]  
 <StructVariable Name="Farbe"
     Type="String"
     InitialValue="gelb"
     Behavior="Manual"
     Description="Fictitious value">
 [...]  
 <Subvar Name="Rotteil"
     Type="Integer"
     InitialValue="128"
     Behavior="Manual"
     Description="The red proportion of the color"
 />
 <Subvar Name="Blauteil"
     Type="Integer"
     InitialValue="128"
     Behavior="Manual"
     Description="The blue proportion of the color"
 />
 <Subvar Name="Gruenteil"
     Type="Integer"
     InitialValue="128"
     Behavior="Manual"
     Description="The green proportion of the color"
 />
 [...]  
</StructVariable>  
 [...]  
```

This example uses the data node of type `StructVariable`. This data node contains several data nodes of type `Subvar`.

Different templates can be used for the different types of data nodes.
3.5.7.19 Template transformation

The `ProcessXMLData()` command dynamically generates an HTML file (or just a text fragment) from an XML data file and an XML template file.

For this purpose, the parser runs through the data file step by step, from top to bottom.

It reads in a data node. Following this, a search is performed in the template file for a matching template for this data node. If a template is found, the template is applied to the data node. The template is an MWSL fragment. The sole difference is that the attributes of the XML data node in the template are available as standard process variables if a variable source is not specified. If there are identical names, the XML attributes overlay the corresponding process variables. If the variable source PROCESS is specified explicitly, the process variables are always used.

Example:
```
Prozessvariable
Color Value "Green"
```

Data file:
```
<?xml version="1.0" standalone="yes"?>
[...]
<Variable Name="ZUFUEHRUNG.STATE"
Farbe="Red"
/>
[...]
<Variable Name="Language"
/>
[...]
```
Template file:
```xml
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Variable">
    <POSITION NAME="LINE">
      <![CDATA[
        <MWSL>write (GetVar("Name") + ":");</MWSL>
        <MWSL>write (GetVar("Color") + 
"\r\n");</MWSL>
      ]]>                
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
```

Output:
ZUFUHRUNG.STATE: Red
Language: Green

Sequence of a Template-process:

The figure shows the sequence of the TemplateParse-process.
The Template is run through one time for each data element (yellow and green blocks). The attributes of the current data element can thereby be accessed as variables.
Chronological sequence:

1. The `<APPMENU>` tag is found in the Source and the Template is run. No processing operation is found for this tag in the Template, therefore, no data element is written to the output.
2. The first `<BUTTON>` tag is found and processed by the Template.
3. The second `<BUTTON>` tag is found and processed by the Template.
4. No other tags are found; processing is complete.

Another example:

**Calling MWSL file**

```html
<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>
  <BODY>
    <MWSL>
      <!--
      write(ProcessXMLData{
      "<EXTERNAL SRC="/MWSL/variables.xml "/>",
      "<TEMPLATES><EXTERNAL SRC="/MWSL/variablesTemplate.xml="/>
      </TEMPLATES>"
      });
      -->
    </MWSL>
  </BODY>
</HTML>
```
variables.xml
<?xml version="1.0" standalone="yes"?>
<Provider Name="MyVarProvider">
  <Variable Name="ZUFUEHRUNG.STATE"
    Type="String"
    InitialValue="good"
    Behavior="Manual"
    Description="Status of part infeed."
  />
  <Variable Name="Language"
    Type="String"
    InitialValue="de_DE"
    Behavior="Manual"
    Description="Language setting of web page"
  />
</Provider>
<!-- End of File -->

variablesTemplate.xml
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Provider">
    <POSITION NAME="HEAD">
      <![CDATA[
        <TABLE BORDER="1">
          <TR>
            <TH>VariableName</TH>
          </TR>
        </TABLE>
      ]]>
    </POSITION>
    <POSITION NAME="FOOT">
      <![CDATA[
        <TR>
          <TD>!MWSL!--GetVar("Name");--</MWSL>
        </TD>
      ]]>
    </POSITION>
  </TEMPLATE>
  <TEMPLATE NAME="Variable">
    <POSITION NAME="LINE">
      <![CDATA[
        <TR>
      ]]>
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
Call the ProcessXMLData command to start the parser with the data file variables.xml. The <Provider> tag is found first.

In the variablesTemplate.xml template file, a search is performed in order to ascertain whether a template has been defined for this type.

The Position="HEAD" area is executed.

The parser reads the next tag from the data file and generates the additional lines of the HTML file to be output based on the appropriate template in the template file.

The same happens with the next tag.

The footer part of the template provider is executed with the end tag </Provider>.

The generated portion is written in bold print in the following routine.
Generated file:

```html
<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>
  <BODY >
    <TABLE BORDER="1">
      <TR>
        <TH>Variablenname</TH>
      </TR>
      <TR>
        <TD>ZUFUEHRUNG.STATE</TD>
      </TR>
      <TR>
        <TD>Language</TD>
      </TR>
      <TR>
        <TD>Hello</TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

### 3.5.7.20 MWSL in XML attributes

As part of template parsing, it is also useful to write MWSL-statements to XML attributes of the data file, which are then evaluated at the time of parsing.

Example:

```xml
<?xml version="1.0" standalone="yes"?>
<Motor Name="M1"
    Nummer="1"
    Type="Dreh"
    Nennleistung = "7"
    Drehzahl="3"
    Alter="2"
    Farbe="RED"
    Prozess="&MWSL;WriteVar(&quot;CPULoad&quot;,&quot;PROCESS&quot;,&quot;%e&quot;); &END_MWSL;"
/>
```
The example uses the following MWSL command:

"&MWSL;WriteVar("CPULoad", &quot;PROCESS&quot;, &quot;%e&quot;); &END_MWSL;"

The command would appear as follows in a template file or an MWSL file:

<MWSL>WriteVar("CPULoad", "PROCESS", "%e");</MWSL>

This command outputs the value of the CPULoad variables from the PROCESS variable source. Please note that <MWSL> must be replaced with &MWSL;, and </MWSL> with &END_MWSL. In addition, double quotation marks " must be replaced with &quot;.

The rest corresponds to the MWSL syntax.

3.5.7.21 Examples

Examples of how MWSL can be used

The examples shown here outline the options for using MWSL.

Setting variable values using the SetVar function

<MWSL>
    SetVar(GetVar("VARNAME"), GetVar(GetVar("VARNAME"), "URL"));
</MWSL>

In this example, the variable whose name is saved in the VARNAME process variable is initialized with the value of the VARNAME URL variable.

For illustration:

GetVar("VARNAME") supplies the content of the VARNAME process variable. This value will be viewed, in turn, as a variable name.

If we assume that the content of the VARNAME process variable is "Jack", the overall call already looks much simpler:

Overall call: SetVar ("Jack", GetVar ("Jack", "URL"));

If we were to assume that the URL variable "Jack" had the content "is a great guy", this would be equivalent to the following expression:

SetVar("Jack", "is a great guy");
TestTemplate.mcs

This example will be used to briefly explain the template mechanism again. For illustrative purposes, a few passages in the respective files are marked.

The template produces a table.

The called file

TestTemplate.mcs

<HTML>
  <HEAD>
    <TITLE>
      MWSL Template Test Page
    </TITLE>
  </HEAD>
  <BODY>
    <MWSL>
      <!--
        write(ProcessXMLData("<EXTERNAL SRC="/MWSL/variables.xml"/>",
                          "<TEMPLATES><EXTERNAL SRC="/MWSL/variablesTemplate.xml"/>
                          </TEMPLATES>"))
        -->
    </MWSL>
  </BODY>
</HTML>
VariablesTemplate.xml

```xml
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Provider">
    <POSITION NAME="HEAD">
      <![CDATA[
        <TABLE BORDER="1">
          <TR>
            <TH>Varname</TH>
            <TH>Type</TH>
            <TH>Description</TH>
            <TH>Value</TH>
            <TH>NI/NL</TH>
          </TR>
        </TABLE>
      ]]>}
    </POSITION>
    <POSITION NAME="FOOT">
      <![CDATA[
        <TR><TD COLSPAN="5"></TD></TR>
      ]]>}
    </POSITION>
  </TEMPLATE>

  <TEMPLATE NAME="Variable">
    <POSITION NAME="LINE">
      <![CDATA[
        <TR align=center><A HREF="<MWSL>Link()"<MWSL>"><MWSL>Name()"<MWSL></A>
          <TD>Type()"<MWSL></TD>
          <TD>Description()"<MWSL></TD>
        </TR>
      ]]>}
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
```
In this case, the template file consists of multiple templates. The Provider template places the header and footer parts for the data (table header and footer).

The Variable template is appointed for the lines of data output (one table row per variable entry).

The data are inserted after the corresponding attributes.

The data file variables.xml

```xml
<?xml version="1.0" standalone="yes"?>
<Provider Name ="MyVarProvider">
  <Variable Name="ZUFUEHRUNG.STATE"
    Type="String"
    InitialValue="good"
    Behavior="Manual"
    Description="Status of part infeed"
  />
  <Variable Name="Language"
    Type="String"
    InitialValue="de_DE"
    Behavior="Manual"
    Description="Language setting of web page"
  />
</Provider>
```
### Generated HTML file:

```html
<html>
<head>
    <title>
        MWSL Template Test Page
    </title>
</head>
<body>
    <table border="1">
        <tr>
            <th>Varname</th>
            <th>Type</th>
            <th>Description</th>
            <th>Value</th>
            <th>NI/NL</th>
        </tr>
        <tr>
            <td align=center>
                <a href="">ZUFUEHRUNG.STATE</a>
            </td>
            <td>String</td>
            <td>Status of part infeed</td>
            <td>good</td>
            <td>2 / 2</td>
        </tr>
        <tr>
            <td align=center>
                Language
            </td>
            <td>String</td>
        </tr>
    </table>
</body>
</html>
```
MainNavigation.mcs

This file contains the actual body of the HTML page to be generated.
The dynamic part is generated with the ProcessXMLData() command and inserted in <table>.
MainNavigation.xml contains the data part for the generation.

```xml
<?xml version="1.0" standalone="yes"?>
<MAINNAVIGATION>
  <APPLICATION NAME = "Entrance"
    CLIENTAREA = "/Portal/Entrance.mcs"
    TITLE = "Back to Entrance Page." />
  <APPLICATION NAME = "MWSL Test"
    CLIENTAREA = "/MWSL/Start.mcs"
    TITLE = "Test environment for MWSL."/>
  <APPLICATION NAME = "File Browser"
    SECUREGROUP = "Administrator"
    CLIENTAREA = "/www"
    TITLE = "Browse the Filesystem" />
  [...]  
  <APPLICATION NAME = "CSSA"
    SECUREGROUP = "User"
    CLIENTAREA = "/CSSA/Main.mcs"
    TITLE = "PKI Interface."/>
  <APPLICATION NAME = "VarSimulator"
    CLIENTAREA = "/Simulator/Simulator_index.mcs"
    TITLE = "Simulate several variables."/>
</MAINNAVIGATION>
```

/ Templates/MainNavigation.xml

```xml
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
  <TEMPLATE NAME="APPLICATION">
    <POSITION NAME="LINE">
      <![CDATA[
      <td>
        <input class = "MainMenu"
          type = "BUTTON"
          title = "<MWSL> WriteVar("TITLE")</MWSL>"
          value = "<MWSL> WriteVar("NAME")</MWSL>"
          OnClick =
            "NavigateApp('<MWSL>WriteVar("CLIENTAREA")</MWSL>')"
        />
        
      </td>
    ]]>  
  </POSITION>
  </TEMPLATE>
</TEMPLATES>
```

This file is run for each data node and the appropriate variables are inserted.
3.5 User-defined pages

Generated HTML file

```html
<html>
<head>
    <title>
        MiniWeb Main Navigation
    </title>
</head>
<body>
    <table>
        <tr>
            <td>
                <input class="MainMenu" type="BUTTON" title="Back to Entrance Page." value="Entrance" OnClick="NavigateApp('/Portal/Entrance.mcs')"
            </td>
            <td>
                <input class="MainMenu" type="BUTTON" title="Test environment for MWSL." value="MWSL Test" OnClick="NavigateApp('/MWSL/Start.mcs')"
            </td>
            <td>
                <input class="MainMenu" type="BUTTON" title="Browse the Filesystem" value="File Browser" OnClick="NavigateApp('/www')"
            </td>
            [...]  
            </td>
            <td>
                <input class="MainMenu" type="BUTTON" title="PKI Interface." value="CSSA" OnClick="NavigateApp('/CSSA/Main.mcs')"
            </td>
            <td>
                <input class="MainMenu" type="BUTTON" title="Simulate several variables." value="VarSimulator" OnClick="NavigateApp('/Simulator/Simulator_index.mcs')"
            </td>
        </tr>
    </table>
</body>
</html>
```


**AppNavigation.mcs**

**Call:**

```xml
<MWSL>
    WriteXMLData("<EXTERNAL SRC=" + GetVar("XML", "URL") + "/">", 
    "<TEMPLATES><EXTERNAL SRC=" + 
    GetVar("TEMPLATE", "URL") + "/"></TEMPLATES>");
</MWSL>
```

When the call is made, the data and the template file are transferred from the URL.

In this example, the transferred template file is assumed to be AppNavigation.xml and the transferred data file MWSLTestMenu.xml.

**MWSLTestMenu.xml**

```xml
<?xml version="1.0" ?>
<APPMENU>
    <MENU>
        <BUTTON NAME = "Menu1"
            LABEL = "Variablentest"
            TITLE = "Tooltip"
            STATUS = "Statusline"
            CLIENTAREA = "/MWSL/Variablentest.mcs?Parameter=4711"
            />
        
        <BUTTON NAME = "Menu9"
            LABEL = "Versuch"
            TITLE = "Tooltip"
            STATUS = "Statusline"
            CLIENTAREA = "/MWSL/Versuch.mcs" />
    </MENU>
</APPMENU>
```

In the following template file, some attributes are queried as to their existence and the corresponding lines executed.

In this case, only the condition for **CLIENTAREA** is run since no other queried attribute is present.
AppNavigation.xml

```xml
<?xml version="1.0" standalone="yes"?>
<TEMPLATES>
  <TEMPLATE NAME="BUTTON">
    <POSITION NAME="LINE">
      <![CDATA[
        <TR>
          <TD>
            <INPUT CLASS = "MainMenu"
              TYPE = "BUTTON"
              TITLE = "<MWSL>TITLE()</MWSL>"
              VALUE = "<MWSL>LABEL()</MWSL>"
            >
            <MWSL>
              if ( ExistVariable( "CLIENTAREA" ) )
              {
                write("OnClick=
                  "top.ClientArea.window.navigate('{}'
                    + GetVar("CLIENTAREA") + '{}")");
              }
              if ( ExistVariable ( "TOP" ) )
              {
                write("OnClick = "top.window.navigate('{}' +
                    GetVar("TOP") + '{}")");
              }
              if ( ExistVariable ( "WIN" ) )
              {
                write("OnClick = "window.open('{}' +
                    GetVar("WIN") + '{}")");
              }
              if ( ExistVariable ( "ACTION" ) )
              {
                write("OnClick = "top.ClientArea.window." +
                    GetVar("ACTION") + '{}")");
              }
            </MWSL>
          </TD>
        ]]> />
      </TR>
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
```

The template file contains `if` conditions, in which attributes from the data file are queried. This results in different statements for each data node during runtime. The generated HTML file will then only contain the line corresponding to the respective data node.
3.5 User-defined pages

generierte Ausgabe

```html
<TD>
  <INPUT CLASS = "MainMenu"
    TYPE = "BUTTON"
    TITLE = "Tooltip"
    VALUE = "Variablentest"
    OnClick = "top.ClientArea.window.navigate('MWSL/Variablentest.mcs?Parameter=4711')"
  />
</TD>

[…]

<TD>
  <INPUT CLASS = "MainMenu"
    TYPE = "BUTTON"
    TITLE = "Tooltip"
    VALUE = "Versuch"
    OnClick = "top.ClientArea.window.navigate('/MWSL/Versuch.mcs')"
  />
</TD>
```

3.5.8 **User-defined directory page**

In IT DIAG, various tasks that can be configured using the `<DEFAPP>` tag in WebCfgFrame.xml are compiled in the default application.

The default application contains the following tasks:

- Sending of files
  - HTML pages
  - Images
  - Etc.

- Receiving a new WebCfg.xml configuration file
  - Parsing of this file
  - Restarting the web server

- Generating the directory browser
  - Creating the directory contents
  - Deleting of files
  - Creating or deleting of directories
  - Loading of files

For some of the tasks, the default application itself must generate HTML pages and return them to the client, e.g. for the directory browser or the confirmation if a new file has been received.

These pages should be designed by the user so that they can be integrated into the desired appearance of the target system.

For this purpose, multiple HTML fragments are defined, that the default application uses to compile the response pages. These HTML fragments are in the configuration area.

The fragments must be encapsulated in `<![CDATA[ ... ]]>` blocks since they may not always exhibit valid XML syntax.
Overview:

<SERVERPAGES>
[...]
<CONFIGURATION_DATA>
<DEFAPP>
<DIRHEAD>
[...]
</DIRHEAD>
<DIRTAIL>
[...]
</DIRTAIL>
<DIRLINE>
[...]
</DIRLINE>
<LOADSUCCEED>
[...]
</LOADSUCCEED>
<SENDCOMPLETE>
[...]
</SENDCOMPLETE>
</DEFAPP>
[...]
</CONFIGURATION_DATA>
[...]
</SERVERPAGES>

Five of these HTML fragments or HTML pages are defined:

- <DIRHEAD> : Directory header
- <DIRLINE> : Directory entry (one line)
- <DIRTAIL> : Directory footer
- <LOADSUCCEED>: Page confirming that a new configuration XML file was received
- <SENDCOMPLETE> : Page confirming that a new file was received
Operation (software)

3.5 User-defined pages

Structure of a directory page

The directory browser is made up of 3 sections:

- A header area ① <DIRHEAD>
- A data area (containing the directory entries) ② <DIRLINE>
- A footer area ③ <DIRTAIL>

When put together, the three fragments produce an HTML page.

This means that the header area contains an <HTML>-tag but no closing </HTML>-tag, as this is located in the footer area. Otherwise, the header and footer areas can contain any HTML constructs.

One <DIRLINE> fragment is inserted for each directory entry.

The directory browser provides a few variables that can be used in the HTML fragments:
### 3.5 User-defined pages

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%DIRPATH%</td>
<td>Provides the absolute path of the currently displayed directory. Here, the path is displayed from the browser view, i.e., as it appears in the URL and not the physical path in the local file system. The latter is not possible for reasons of security and feasibility.</td>
</tr>
<tr>
<td>%ICONPATH%</td>
<td>Can only be used in the &lt;DirLine&gt; area. Generates an &lt;IMG Src=&quot;...&quot;&gt; for the current file. A matching icon from the MIME type table is established as the icon (based on the file extension). See file types &lt;MIME_TYPES&gt;. In the example screenshot above, the Explorer ball is the icon for the HTML pages, while a folder symbol is the icon for the directories. These are the result of the %ICONPATH% entry.</td>
</tr>
<tr>
<td>%DESTINATIONPATH%</td>
<td>Can only be used in the &lt;DirLine&gt; area. Generates a URL via which the current file can be requested. This variable can be used, for example, to create a link to this file.</td>
</tr>
<tr>
<td>%FILENAME%</td>
<td>Can only be used in the &lt;DirLine&gt; area. Returns the name of the current file.</td>
</tr>
<tr>
<td>%FILESIZE%</td>
<td>Can only be used in the &lt;DirLine&gt; area. Returns the size of the current file.</td>
</tr>
</tbody>
</table>
| %FILEATTRIBUTES% | Can only be used in the <DirLine> area. Returns a character string with the attributes of the current file: 

- R: The file is write-protected.
- H: The file is hidden.
- S: The file is a system file.
- A: The file must be archived; this means it has been changed since the last backup.
- F: The file is in the ROM (fix). 

If an attribute for the file is not set, a "." is entered at the appropriate location. |
3.5 User-defined pages

| %=TRASHBINPATH% | Generates an `<IMG>` tag with a trash can icon (the path is created from the MIME type table, entry: "trashbin"). If the icon is clicked, a script function "Delete()" is called.

This contains the URL to the current file, extended to include the parameter "?DELETE", as a parameter. In this script function (which can be defined in the `<DIRHEAD>` part, for example), it would be possible to request this URL after a confirmation prompt, for instance, whereupon the file would be deleted from the default application. (assuming the user has sufficient rights.)

If a file is sent to the server, the default application must confirm this upload with an HTML response. This HTML response is stored under the `<SENDCOMPLETE>` tag. This HTML file can contain a corresponding confirmation or even the request to redisplay the modified directory.

| %=FORWARDURL% | Can only be used in the `<SENDCOMPLETE>` area.

Returns a URL that can be used to display the modified directory (expanded to include the file that was just loaded).

For example, if only:
```
<SCRIPT> location.replace("%=FORWARDURL%"); </SCRIPT>
```

is entered in the body of the HTML page, this is where the directory page is automatically loaded.

MWSL statements can also be used in the fragments.

**Confirmation pages**

The response for a successful WebCfg.xml upload is stored in the `<LOADSUCCEED>` tag.

The `<SENDCOMPLETE>` tag can be used to control further processing following an Upload or Delete action.
3.5.9 Server Side Includes (SSI)

3.5.9.1 Integration of process values

You can include process values in the user-defined HTML pages using Server Side Includes (SSI).

Note

In SIMOTION controls V4.1 and higher, HTML pages with SSI must be available as a binary file to display process values. A standard HTML page can be converted to a binary file using the supplied conversion tool (Page 119).

HTML pages with static content only do not have to be converted.

Integration of process values

The variables are integrated in the HTML page using the `<%= IDENTIFIER %>` character string. IDENTIFIER is a placeholder, which you must replace with variables from the variable providers. For example, the variable `<%= DeviceInfo.Board %>` returns the name of the control. On a D435, for example, the value is "D435".

Details about the variables and syntax can be found in the Variable providers (Page 235) section.

The source text below shows an example for integrating the variable `userData.user1`. First, the value of the variable is output (system variable `userData.user1`:

```html
<html>
<head>
    <title>Demo Seite</title>
</head>
<body background="#FFFFFF" link="#000000" vlink="#000000">

Demoseite<br>
Systemvariable userData.user1 : <%= var(userData.user1) %>
<br>
<form method="post" action="/VarApp">
    SIMOTION C: userData.user1:
    <input type="TEXT" name="var(userData.user1)" value="%= var(userData.user1)" /></form>
</body>
</html>
```
3.6 OPC XML DA server

3.6.1 Overview
The SIMOTION IT OPC XML DA server enables access via Ethernet to data and operating modes of the SIMOTION device.

What is OPC XML DA?
OPC is the abbreviation for OLE for Process Control and describes a standard interface for communication in automation systems.

With OPC XML DA, it is possible to communicate with a control using Ethernet-based standard message frames. Commands are transmitted via the SOAP (Simple Object Access Protocol) communication protocol.

The interface is defined in a configuration file using a description language based on XML vocabulary. It defines the format of the request and response message streams with which function calls are transmitted (see OPC XML DA R1.0 specification http://www.opcfoundation.org/Downloads.aspx).

This interface can only be used by client applications.

The following figure illustrates a client made available by the OPC Foundation.

The client enables browsing via the system, interface, IO, and global device variables.
Purpose and benefits

The purpose and benefits of the SIMOTION IT OPC XML DA server are as follows:

- Addressable via Ethernet interface in accordance with the OPC XML DA V1.0 standard.
- The server can be addressed by any client application which conforms to the OPC XML DA V1.0 standard, regardless of its operating system (e.g. Linux).

What previous knowledge is required?

In order to understand the SIMOTION IT OPC XML DA server described in this chapter, it is assumed that the user is familiar with OPC XML DA terminology (see OPC XML DA R1.0 specification).
3.6.2 Comparison of OPC XML DA / SIMATIC NET OPC DA

Comparison

The "SIMATIC NET OPC Server for SIMOTION" product exists in addition to the SIMOTION IT OPC XML DA server. This package also allows access to data and operating modes of the SIMOTION device via SIMATIC NET OPC DA.

The following table compares the two packages and describes the basic procedure:

Table 3-13 Basic procedure for accessing data

<table>
<thead>
<tr>
<th>SIMOTION IT OPC XML DA</th>
<th>SIMATIC NET OPC DA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No configuration (OPC export) necessary with SIMOTION SCOUT. Program variable can be activated via a switch.</td>
<td>OPC export with SIMOTION SCOUT required, which has to be repeated for every project change.</td>
</tr>
<tr>
<td>Symbols are resolved in the SIMOTION device, communication by means of text format (XML).</td>
<td>Symbols are resolved during OPC export and stored in the OPC server on the Windows system in binary format; binary communication -&gt; higher data throughput.</td>
</tr>
<tr>
<td>At present only SIMOTION with OPC XML DA. Access to S7 devices not possible at present.</td>
<td>Simultaneous access to SIMOTION and S7 devices.</td>
</tr>
<tr>
<td>Client can run on any operating system.</td>
<td>Based on Windows COM/DCOM technology; client and server can only run on Windows operating systems.</td>
</tr>
<tr>
<td>Communication with standard protocols (TCP/IP, XML, SOAP), no manufacturer-specific (SIEMENS) tools, drivers required on the client system.</td>
<td>S7 protocol used for communication, appropriate manufacturer-specific drivers required on the client.</td>
</tr>
<tr>
<td>Communication only possible via Ethernet.</td>
<td>Communication via PROFIBUS/MPI and Ethernet is possible.</td>
</tr>
<tr>
<td>Direct addressing via firewalls is possible.</td>
<td>Generally, DCOM communication not released for firewalls.</td>
</tr>
</tbody>
</table>
3.6.3 Schematic diagram at the design stage

Example arrangement

The figure below shows an arrangement example of the relevant software for the creation of a client application on a PC. The PC and the SIMOTION device are networked via Ethernet.

![Diagram showing software arrangement](image-url)

Figure 3-81  Design stage overview (example)
3.6.4 Schematic diagram at runtime

Example arrangement

The figure below shows an arrangement example of the client application on a PC and the OPC XML DA server on the SIMOTION device at runtime. Both are networked via the Ethernet.

Figure 3-82 Overview at runtime (example)
3.6.5 Installation

3.6.5.1 Hardware and software requirements at the design stage

Hardware requirements at the design stage

Note
You can freely select the programming environment. The following requirements are an example for Microsoft Visual Studio .NET, but they are not binding.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Minimum requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel Pentium III or compatible,</td>
</tr>
<tr>
<td></td>
<td>800 MHz</td>
</tr>
<tr>
<td>Main memory</td>
<td>128 MB RAM</td>
</tr>
</tbody>
</table>

Software requirements at the design stage

Note
You can freely select the programming environment. The following requirements are an example for Microsoft Visual Studio .NET, but they are not binding.

- **Microsoft Visual Studio .NET:**
- Configuration file (WSDL), in accordance with OPC XML R1.0 specification.
3.6.5.2 Configuring SIMOTION device interface at runtime

Configuring the interface

In order to establish a connection between a PC and a SIMOTION device when the system is running, you must carry out the following steps for the configuration of the Ethernet interface:

Table 3-15 Configuring the interface

<table>
<thead>
<tr>
<th>Step</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>You must activate the functionality in the SIMOTION project when configuring the control hardware via the &quot;Ethernet extended&quot; properties in the &quot;OPC XML/diagnostic pages&quot; function.</td>
</tr>
<tr>
<td>2</td>
<td>The server address must be known. This can be configured using the interface settings in HW Config.</td>
</tr>
</tbody>
</table>

3.6.6 Making unit variables available

To make variables available on the SIMOTION IT OPC XML DA server, you have to declare them as VAR_GLOBAL.

Declaring unit variables in the interface

In the declaration table, you define the data type for each variable. Only variables declared as VAR_GLOBAL are available in OPC XML DA.

The following figure shows an example of unit variable declarations in an MCC program.

Figure 3-83 Declaring global variables
Permit OPC XML

To activate the variables for OPC XML DA, proceed as follows:

1. Open the Program properties dialog.
2. Select the Compiler tab.
3. Activate Permit OPC-XML, if it is not already activated (standard setting).

The following figure shows how to activate a unit variable from an MCC program.

![Figure 3-84 Making variables available for OPC XML DA](image)

**Note**

The OPC XML activation applies also to variables in LAD/FBD and ST programs. To make variables available for OPC XML DA in an ST program, they have to be defined in a global variable block. This must be located in the interface section.
3.6.7 Example of a client application

Example

The following example describes the most important programming steps for the "Read" method with the Microsoft Visual Studio .NET2003 tool.

The OPC client executes the following application:

The application example displays a Read button in a dialog box. When the button is activated, the client connects to the SIMOTION IT OPC XML DA server and reads a variable. The result is displayed in the output field of the dialog box.

The dialog box of the application example is shown in the following figure:

![Demo client](image)

Figure 3-85 Demo client

Programming steps

The following programming steps are needed:

1. Create a new project with Microsoft Visual Studio .NET and import the WSDL file as the interface description ("Add Web Reference" menu).
2. Create a dialog box with a text field and a Read button.
3. Enter the name assigned for the reference, e.g. "OPCXMLServer", in the program using DemoClient.OPCXMLServer.
4. Declare the server URL in the program as follows:
   ```
   http://<IP address>/soap/opcxml
   ```
   Enter the IP address of your SIMOTION control in place of <IP address>.
5. Instantiate the server proxy object in accordance with the code example and provide the call-up with the required parameters.
6. The required data is returned.
Program section

using DemoClient.OPCXMLServer;
private void ReadButton_Click(object sender, System.EventArgs e)
{
    Service SIMOTION_C_Server = new Service();

    RequestOptions ReadOptions = new RequestOptions();
    ReadOptions.ClientRequestHandle = "";
    ReadOptions/localeID = "DE-AT";
    ReadOptions.RequestDeadlineSpecified = false;
    ReadOptions.ReturnDiagnosticInfo = false;
    ReadOptions.ReturnErrorText = false;
    ReadOptions.ReturnItemName = false;
    ReadOptions.ReturnItemPath = false;
    ReadOptions.ReturnItemTime = false;

    SIMOTION_C_Server.Url = "http://simotion/soap/opcxml";
    SIMOTION_C_Server.Credentials = myCredentials;
    SIMOTION_C_Server.PreAuthenticate = true;
    System.Net.ServicePointManager.Expect100Continue = false;
    SIMOTION_C_Server.Read(ReadOptions,ReadItemList,out ReadReplyList,out ReadErrorList);

    if ((ReadReplyList.items[0] != null) &&
        (ReadReplyList.Items[0].Value != null) &&
        (ReadReplyList.Items[0].Value.GetType().Name != "XmlNode[]")
    Output.Text = ReadReplyList.Items[0].Value.ToString();
    else Output.Text = "<Error>";
}

Note

If the client application has been created with Microsoft Visual Studio .NET, then an "application configuration file" must be added to the solution with the "Add New Item" menu. This text file is required to deactivate the debug information in the SOAP message frame.

The "application configuration file" must have the following contents:

**Application configuration file**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.diagnostics>
        <switches>
            <add name="Remote.Disable" value="1"/>
        </switches>
    </system.diagnostics>
</configuration>
```
3.6.8 SIMOTION IT OPC XML DA server interface

3.6.8.1 Overview

Introduction
This section describes the methods you can run across the interface to the OPC XML DA V1.0 server. The server itself is integrated in the SIMOTION device.

Prior to Version 4.2, the server had to be enabled with a license.

This is just a brief overview. A detailed description of these methods can be found in the document titled "OPC XML DA Specification R1.0" from the OPC Foundation.

You can find an up-to-date and detailed interface description on the home page of the OPC Foundation: http://www.opcfoundation.org

3.6.8.2 Methods which can be called synchronously

The SIMOTION IT OPC XML DA server provides the following methods, which can be called synchronously, under the "OpcXmlDaService" type:

Description of methods

Browse
The "Browse" method allows you to navigate through the available variables.

GetProperties
The "GetProperties" method can query the settings for a specific variable (e.g. access rights, time stamp, data type).

getStatus
The "GetStatus" method supplies information about the server status, the program version and the supported interface version.

Read
The "Read" method reads out variable lists.

Subscribe
The "Subscribe" method passes a list of variable names and receives a handle for the subscription. This handle can be used in the SubscriptionPolledRefresh method to poll the values of the previously defined variables again. See Basics of subscriptions (Page 222).

The buffering property, which corresponds to the "EnableBuffering" attribute, is not supported. This has an effect on the "SubscribeRequestItem" and "SubscribeRequestItemList" methods.
**SubscriptionPolledRefresh**

The "SubscriptionPolledRefresh" method returns the values of the variables written beforehand using the Subscribe method. The handle specifying the subscription is used as a parameter.

The "Holdtime" parameter defines the earliest possible response time. This limits the frequency of data transmission.

The "ReturnAllterms" parameter determines how the "WaitTime" parameter is used.

- **True**
  "WaitTime" is ignored, all requested values are returned immediately.

- **False**
  For the period set in the "WaitTime" parameter, the server checks whether one of the requested values has changed since the last call.
  
  If the specified time expires without a value having been changed, an empty response is returned.

  If values change during the specified time, the changed values are returned immediately and the polling ended.

**SubscriptionCancel**

The "SubscriptionCancel" method cancels the subscription and returns the subscription handle.

Which subscription is to be canceled, must be specified at the call.

If an asynchronous call form is used, the client is informed later of which subscription has been canceled, via a client handle.

---

**Note**

Once the subscription has been canceled, the subscription handle ceases to be valid for the client.

---

**Write**

The "Write" method writes variable lists.
3.6.8.3 Access to variables

Variable access using methods

Variables can be accessed via the methods which can be called synchronously and asynchronously.

---

Note

Information for accessing unit variables in an MCC program can be found in the SIMOTION MCC programming manual.

Information for access in an ST program can be found in the SIMOTION ST programming manual.

---

See also

Making unit variables available (Page 210)
3.7 Trace interface via SOAP (TVS)

Introduction

The SOAP-based service provides a trace service option.

The WebTrace is identical to the SIMOTION SCOUT trace. The only difference is in the format of the output data.

Trace-Service

The "Trace Interface via SOAP" function package enables variable values to be written to a buffer. The values are packed in files and can be retrieved asynchronously via an HTTP request.

This interface can only be used by client applications. The client enables the time characteristic of variables to be traced.

A WSDL file is available for creating the application.
3.7 Trace interface via SOAP (TVS)

3.7.1 Trace Interface via SOAP

Introduction

When working with a trace, the trace can assume various states. The following graphic shows the possible states and transitions. The methods named are described in chapter "Trace interface".

![WebTrace Diagram](image)

**Figure 3-86 WebTrace**

States

After a trace has been created with "InitializeWebTrace", it is in the STOPPED state. With StartTrace, it starts up and writes the desired data to the buffer. Accordingly, it can be stopped again with "StopTrace". After the start, the trace switches to RUNNING state. If the time specified in the call has expired, the trace assumes the FINISHED state. A trace can be deleted at any time with "CancelTrace" in order to create a new trace, for example.
3.7 Trace interface via SOAP (TVS)

3.7.2 Procedure/terms

HTTP methods - data exchange

The trace data are stored as compiled data on the RAM disk using the ReadData method. This data must be retrieved via ordinary HTTP requests.

Note

The data are not deleted by retrieval alone! To prevent the RAM disk from overflowing, an HTTP DELETE call to this URL must follow an HTTP GET call. (Reason: The use case under consideration is one in which a client may have to request the same trace data more than once, e.g. to compare traces that have already been executed.) These temporary data are completely deleted only after a CancelTrace operation, regardless of whether they have already been retrieved or not.

TRIGGERED

The trace offers a triggering option. Depending on the trigger method, different constants or variable symbols must be specified for this. The trace starts with:

- A rising edge (RE), if the variable exceeds the value of a constant.
- A falling edge (FE), if the variable falls below the value of a constant.
- Within a tolerance band (WIB), if the variable lies between two constants.
- Outside of a tolerance band (OOB), if the variable lies outside of a tolerance band.
- Bit mask has value (BHV), if the variable has a specified value after masking with a constant.

If the trace is set up in TRIGGERED mode, a trigger condition as described below must be specified. This trigger acts as a SingleShot. However, the MatchCountTriggerPoint parameter can be used to set the trigger for repeated occurrences (e.g. five: triggering only takes place the fifth time the trigger condition appears).

In this case, the trace takes place only after the trigger. The data are recorded for the duration specified during setup.
IMMEDIATE / ENDLESS

The counterpart to the TRIGGERED Trace is the IMMEDIATE Trace, which begins the trace immediately after the StartTrace call has occurred. In this case as well, the data are recorded for the duration specified during setup.

The ENDLESS Trace uses a ring buffer trace. Trigger conditions are not evaluated. ENDLESS Trace starts as soon as the StartTrace event arrives. However, it is terminated only when StopTrace is called explicitly. The size of the ring buffer must also be specified using the duration for the initialization call. Thus, an appropriate value must be found that uses fewer resources, but is sufficient to retrieve data in a timely manner via HTTP.

The size of the ring buffer \( B \) is determined from the number of variables \( N \), their size \( S \), the transferred time duration \( t \) and the cycle clock \( T \) in which they are traced.

\[
B = \frac{n-1}{t/T} \sum_{i=0}^{n-1} S_i
\]

Within the transferred time duration, the buffer must be discharged at least once by calling the "readData" function in order to prevent the oldest trace data from being overwritten each time.

If the parameters require a larger memory area, the recording duration is reduced so that no more than 256 KB are occupied.

3.7.3 Error handling

All implemented methods of the TVS (trace via SOAP) supply either the requested data or status information, or an SOAP_FAULT. This behavior enables the use of the SoapFaultError in the .NET framework. The Try-Catch mechanism enables convenient error handling.
3.7 Trace interface via SOAP (TVS)

3.7.4 Basics of subscriptions

Introduction

"GetStatus" must be called in order to query the status of a trace. The fastest possible detection of a status change requires extremely frequent polling, which places an unnecessary load on the CPU in the control and causes heavy traffic on the network.

To optimize this operation, OPC XML DA provides "subscriptions". With subscriptions, a query does not receive a response until the required variable changes or a timeout occurs. Thus, the connection is kept open without causing traffic. As soon as relevant data are available for the client, these data are sent to the client.

The trace supports this mechanism via the SOAP web service also. However, in this case, only the status of the trace object is checked, as this is the only valuable information in this environment.

As soon as the status changes (e.g. RUNNING -> FINISHED), the clients that issued the query receive a response accordingly. In essence, any number of clients is possible (as long as there are sufficient resources).

Operational sequence

The operational sequence of a subscription is as follows:

Figure 3-87 Subscription
First, a subscription must be created. It is answered with a unique ServerHandle, which is required for further communication.

SubscriptionRefresh can be called as often as necessary to start a new query. This request receives two time specifications in milliseconds as parameters:

- **HoldTime:**
  This time indicates the minimum hold time for the response, irrespective of whether the status has changed.

- **WaitTime:**
  The WaitTime begins after the HoldTime has expired. If the trace status has changed, the response to the current status is sent immediately. If there is no change, the response is sent once the WaitTime has expired.

The exact method calls are explained in the next section.
3.7 Trace interface via SOAP (TVS)

3.7.5 Interface

3.7.5.1 Global definitions

**TraceStateEnum**

Enumerator that indicates the status of the trace object.

**Declaration:**

```java
public enum TraceStateEnum {
    RUNNING, STOPPED, ERROR, EMPTY, FINISHED, TRIGGERED
}
```

**TraceDataCycleEnum**

Enumerator that specifies the cycle clock in which the data are to be traced. It must be noted here that large traces may cause a level overflow.

**Declaration:**

```java
public enum TraceDataCycleEnum {
    IPO1, IPO2, SERVO,
}
```
Structure VDSC

Structure that contains information about the traced variables. These are:

- **Variable name** `VarName`
- **Variable type** `VarType` in S7 notation (e.g. DINT or BYTE)
- **VarOffset** specifies the offset of the variable within the data stream (relative to the start of the I/O container)
- **Variable length** `VarLen` (optional)

Declaration:

```csharp
public class VDSC
{
    public string VarName;
    public string VarType;
    public System.UInt32 VarOffset;
    public System.UInt32 VarLen;
}
```

TriggerCondition

Structure indicating the trigger of a trace. It contains the variable to be compared in symbolic names according to VarProvider notation.

- **Variable** refers to the variable for comparison.
- **Constant1** is a constant which is set according to the trigger type.
- **Constant2** is a constant which is set according to the trigger type.
- **Operation** indicates the comparison type as enumerator `TriggerOperationType`.
- **MatchCountTriggerPoint** indicates how many times the trigger condition must apply before the trigger is activated.
- **GlobalTriggerID** is optional and contains a unique ID, which is generated by the browser when a distributed trace is set up.
Declaration:

```java
public class TriggerCondition
{
    public string Variable;
    public string Constant1;
    public string Constant2;
    public TriggerOperationType Operation;
    public System.UInt32 MatchCountTriggerPoint;
    public System.UInt32 GlobalTriggerID;
}
```

For this purpose, the comparison type:

**Call:**

```java
public enum TriggerOperationType
{
    RE,
    FE,
    WIB,
    OOB,
    BHV
}
```

The following table gives an overview of various possible combinations of trigger types and constants.

<table>
<thead>
<tr>
<th>Trigger type</th>
<th>Symbol</th>
<th>Description</th>
<th>Constant1</th>
<th>Constant2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rising Edge</td>
<td>RE</td>
<td>Triggered if variable exceeds the value of Constant1 in positive direction</td>
<td>Limit exceeded</td>
<td>-Not used-</td>
</tr>
<tr>
<td>Falling Edge</td>
<td>FE</td>
<td>Triggered if variable exceeds the value of Constant1 in negative direction</td>
<td>Limit exceeded</td>
<td>-Not used-</td>
</tr>
<tr>
<td>Within a tolerance Band</td>
<td>WIB</td>
<td>Triggered if variable is located within the interval spanned by Constant1 and Constant2</td>
<td>Lower limit</td>
<td>Upper limit</td>
</tr>
<tr>
<td>Out of tolerance Band</td>
<td>OOB</td>
<td>Triggered if variable is located outside the interval spanned by Constant1 and Constant2</td>
<td>Lower limit</td>
<td>Upper limit</td>
</tr>
<tr>
<td>Bit pattern</td>
<td>BHV</td>
<td>Triggered if variable logically rounded with Constant1 produces a value equal to Constant2, ((v&amp;c1)==c2)</td>
<td>Bit mask</td>
<td>Result of comparison</td>
</tr>
</tbody>
</table>

Overview of trigger types and constants
3.7 Trace interface via SOAP (TVS)

### 3.7.5.2 Methods

#### StartTrace

The `StartTrace` method starts an initialized trace. The SoapFault "No Trace available" is returned if a trace has not yet been initialized. `StartTrace` is ignored (with a positive result) if the trace is already in progress.

The `GlobalTriggerID` must be transferred for a distributed trace. The trigger ID is a unique ID, which unambiguously identifies the station currently responsible for triggering; the same ID is used for all stations.

When a trace is restarted, the trigger ID must be reassigned so that the trigger events can be distinguished from one another.

**Call:**

```csharp
public TVS_Client.TVS.StartTrace_Response StartTrace
(
    uint GlobalTriggerID
)
```

```csharp
public class StartTrace_Response
{
    public TraceStateEnum TraceState;
}
```

#### StopTrace

The `StopTrace` method stops a trace in progress. The "No Trace available" SoapFault is returned when a trace has not yet been initialized. This is ignored (with a positive result) if the trace has already stopped.

```csharp
public TVS_Client.TVS.StopTrace_Response StopTrace
(
    uint GlobalTriggerID
)
```

```csharp
public class StopTrace_Response
{
    public TraceStateEnum TraceState;
}
```
Call:

```java
public TVS_Client1.TVS.StopTrace_Response StopTrace ()
{
    public class StopTrace_Response
    {
        public TraceStateEnum TraceState;
    }
}
```

**CancelTrace**

The `CancelTrace` method deletes an active trace. The traces switches to EMPTY status, and all trace data are deleted. (Note: Data blocks of the WebTrace that have been requested but have not yet been retrieved are also deleted (see WebTrace::ReadData())  

The "No Trace available" SoapFault is returned when a trace has not yet been initialized.

Call:

```java
public TVS_Client.TVSIO.CancelTrace_Response CancelTrace ()
{
    public class CancelTrace_Response
    {
        public TraceStateEnum TraceState;
    }
}
```

**GetStatus**

The `GetStatus` method returns the current status of the trace. When a trace object is deleted or has become invalid, then TraceIsValid will contain "false". In this case, the trace must be deleted via `CancelTrace`.

Call:

```java
public TVS_Client.TVSIO.GetStatus_Response GetStatus ()
{
    public class GetStatus_Response
    {
        public bool TraceIsValid;
        public TraceStateEnum TraceState;
    }
}```
ReadData

The `readData` method saves trace data on the RAM disk and supplies the URLs of the files in the return value. These data can be retrieved from the client with an HTTP-GET request. The "No Tracedata available" SoapFault is returned if no trace data are available.

Call:

```java
public ReadData_Response ReadData()

public class ReadData_Response
{
    public TraceStateEnum TraceState;
    public string[] URL;
}
```

InitializeWebTrace

A trace is created with `InitializeWebTrace`. `VariablesToTrace` is the list of symbolic names in accordance with VarProvider notation. `TraceDataCycle` determines the cycle clock in which the data is to be recorded. `TraceStartType` determines the type of trace. `Duration` specifies the duration of the recording in milliseconds. With an endless trace, this parameter specifies the size of the ring buffer in milliseconds.

`MatchCountTriggerPoint` in `TriggerInformation` determines how often the trigger must occur before it actually performs a trigger operation and, as a result, starts the recording. `Pretrigger` specifies the number of values which are to be recorded prior to triggering ("history").

Call:

```java
public InitializeWebTrace_Response InitializeWebTrace
(
    string[] VariablesToTrace,
    TraceDataCycleEnum TraceDataCycle,
    TraceStartTypeEnum TraceStartType,
    uint Pretrigger,
    uint Duration,
    TriggerCondition TriggerInformation
    string[] DevicesInvolved
)
```
3.7 Trace interface via SOAP (TVS)

public class InitializeWebTrace_Response
{
    public VDSC[] CurrentlyTracedVariables;
    public TraceStateEnum TraceState;
    public string UID;
    public string[] DevicesInvolved;
}

InitializeWebTraceEx

InitializeWebTraceEx is identical to InitializeWebTrace apart from the return value, where the variables are sorted according to their particular offset.

GetTraceParameters

GetTraceParameters can be used to read out an existing trace configuration.

Only one WebTrace is returned (if one actually exists).

Call:

public GetTraceParameters_Response GetTraceParameters
{
    string UID
}

public class GetTraceParameters_Response
{
    public TraceTypeEnum TraceType;
    public VDSC[] CurrentlyTracedVariables;
    public TraceStateEnum TraceState;
    public TraceDataCycleEnum TraceDataCycle;
    public string UID;
    public TraceStartTypeEnum TraceStartType;
    public unsignedInt Pretrigger;
    public unsignedInt Duration;
    public TriggerCondition TriggerInformation;
    public unsignedInt IOContainerOffset;
    public unsignedInt IOContainerLength;
    public hexBinary ClientHandle;
    public string[] DevicesInvolved;
}
EnableTrigger

Only for the distributed trace.

When a distributed trace has been set up and started, this activates the triggers. The TriggerID must be unique so that the stations can differentiate between them and do not lose their way on the network. The sequence is important: All traces should be running before the trigger is activated, to avoid loss of trigger events.

Return value: TriggerState activated or not activated.

Call:

```java
public EnableTrigger_Response EnableTrigger
{
    uint TriggerID
}
```

```java
public class EnableTrigger_Response
{
    public TraceStateEnum TraceState;
}
```

ReadData

With ReadData, the TVS service is requested to read out the trace buffer and pack the data in temporary files. These can then be accessed via HTTP under the relative paths specified in URL. If the buffer is empty, a response is made to the request with the "No Tracedata available" SoapFault. Currently, a maximum of 8 compiled files with a maximum of 8,192 recording points are provided for each request.

ReadDataArchive

If a trace is in the STOPPED state, this function can be used to request the recorded data.

The function supplies a URL, from which a WTRC data archive can be downloaded and then displayed in the WebTraceViewer.

---

**Note**

The WTRC file is deleted as soon as it has been downloaded.
3.7 Trace interface via SOAP (TVS)

Call:

```csharp
public ReadDataArchive_Response ReadDataArchive ()
```

```csharp
public class ReadDataArchive_Response
{
    public TraceStateEnum TraceState;
    public string URL
}
```

**ReadDataArchives**

*Only for the distributed trace.*

ReadDataArchives automatically retrieves the trace data for all stations involved in a distributed trace and combines them in a WTRC file. The `URLField` array is used to transfer a list of the URLs for all the stations involved in the trace. The trace must be in the STOPPED state in order to use this method.

The return value is identical to that for ReadDataArchive.

Call:

```csharp
public ReadDataArchives_Response ReadDataArchives (public string[] URLField;
```

```csharp
public class ReadDataArchives_Response
{
    public TraceStateEnum TraceState;
    public string URL
}
```
3.7.5.3 **Subscriptions**

**Introduction**

The subscription methods are listed below.

**Subscribe**

A subscription is created using the `Subscribe` method. The response is a `ServerHandle` that can be used to uniquely identify a subscription operation. In addition, the current `TraceStatus` is supplied.

**Call:**

```java
public TVS_Client.TVS.Subscribe_Response Subscribe ()

public class Subscribe_Response {
    public System.UInt32 ServerHandle;
    public TraceStateEnum TraceState;
}
```

**SubscriptionRefresh**

`SubscriptionRefresh` is used to query the status of the trace again; the trace was previously queried with the `Subscribe` method. The server response is received after `HoldTime` (milliseconds) + `WaitTime` (milliseconds), if the status has not changed during this time, or the response is received (at the earliest) after the `HoldTime` has expired and before the `WaitTime` has expired, if the status of the trace changes during the `WaitTime`. As such, the response is never expected before the `HoldTime` has elapsed.

In the response, `StateChanged` indicates whether the status has changed between request and response (`true`) or whether the `TraceState` status matches the status during the request (`false = WaitTime expired`).
3.7 Trace interface via SOAP (TVS)

Call:

```csharp
public TVS_Client.TVS.SubscriptionRefresh_Response SubscriptionRefresh
{
    System.UInt32 ServerHandle ,
    System.UInt32 WaitTime ,
    System.UInt32 HoldTime
}

public class SubscriptionRefresh_Response {
    public bool StateChanged;
    public TraceStateEnum TraceState;
}
```

SubscriptionCancel

With SubscriptionCancel, a subscription is canceled and the resource is enabled. The response indicates whether the Cancel operation was successful. Any current SubscriptionRefreshes are cancelled and responses sent immediately.

Call:

```csharp
public TVS_Client.TVS.SubscriptionCancel_Response SubscriptionCancel
{
    System.UInt32 ServerHandle
}

public class SubscriptionCancel_Response {
    public bool SubscriptionCanceled;
}
```
3.8 Variables providers

3.8.1 Overview

Variable providers

The data of the SIMOTION device can be accessed via the "variable providers". Each provider enables access to certain variables.

At present there are four variable providers; these are described in the section below.

- MiniWeb
- SIMOTION
- SIMOTION diagnostics
- UserConfig

You can access the data supplied by the variable providers from SIMOTION IT OPC XML-DA, SIMOTION IT DIAG (standard diagnostics pages), or, if necessary, from user-defined HTML pages.

3.8.2 SIMOTION

You can access SIMOTION process variables via the "SIMOTION" provider. As of V4.1, you can also change the operating mode, initiate backups with RamToRom and ActiveToRam, and access drive parameters and technological alarms.

Note

An exact list with description can be found in the online help for SIMOTION SCOUT in "System Functions, System Variables and Configuration Data".
Variables syntax of the "SIMOTION" provider

With OPC XML DA V1.0, access to the variables of the SIMOTION device is via the terms "ItemPath" and "ItemName". In user-defined HTML pages, they are accessed via the "ItemName".

**ItemPath**
The name for "ItemPath" is always "SIMOTION" for SIMOTION process variables.

```
ItemPath="SIMOTION"
```

**Note**
The "ItemPath" is only needed for access via OPC XML-DA, not for SIMOTION IT DIAG or user-defined HTML pages.

### 3.8.2.1 Accessing system variables/technology object system variables

For **system variables**, the **ItemName** syntax is:

```
ItemName="var/name"
```

Example: `ItemName="var/userData.user3"`

For **technology object system variables**, the **ItemName** syntax is:

```
ItemName="to/name.variable"
```

Example: `ItemName="to/Axis_1.positioningState.actualPosition"`

**Note**
The names of the system variables and technology object system variables to be used can be found in the online help for SIMOTION SCOUT in "System Functions, System Variables and Configuration Data".

For unit variables in the interface, the **ItemName** syntax is:

```
ItemName=" unit/name.variable"
```

Example: `ItemName=" unit/prog_1.var_1"

**Note**
The names to be used for the unit variables in the interface correspond to the program and variable names in **lower case characters**.
3.8.2.2 Accessing technology object configuration data (V4.1 and higher)

For technology object configuration data, the ItemName syntax is:

\[ \text{ItemName} = \text{cfg/TOName.activeConfigData|setConfigData.variable} \]

- activeConfigData: Currently valid configuration files, read-only
- setConfigData: Data set image, write access possible

The data can be write-accessed if the “effectiveness” property has the "CHANGEABLE_WITH_RESTART" or "CHANGEABLE_WITHOUT_RESTART" value.

In the case of "CHANGEABLE_WITH_RESTART", the change does not take effect until the respective technology object has been restarted.

Example: \[ \text{ItemName} = \text{cfg/Axis_0.setConfigData.Restart.restartActivationSetting} \]

---

Note

The names of the TO configuration data to be used can be found in the online help for SIMOTION SCOUT in “System Functions, System Variables and Configuration Data”.

3.8.2.3 Accessing drive parameters (V4.1 and higher)

For drive parameters, the ItemName syntax is:

\[ \text{ItemName} = \text{drv/TOName|LogAddr.Params.ParamNo} \]

- TOName: Specifies the technology object name (possible if an Axis technology object exists for the drive object)
- LogAddr: Specifies the logical drive address
- ParamNo: Parameter number

If an attempt is made to write-access a read-only drive variable, the drive issues a feedback message (error code) to this effect.

Example 1: \[ \text{ItemName} = \text{drv/Axis_0.Params.105} \]
Example 2: \[ \text{ItemName} = \text{drv/256.Params.5} \]
3.8 Variables providers

3.8.2.4 Accessing technological alarms (V4.1 and higher)

For **technological alarms**, the **ItemName** syntax is:

```
ItemName="dev/Alarm.Variable|Values-Array
```

**Variable:**
- **State**
  - Status of query:
    - READY
    - BUSY
    - ERROR
- **Version**
  - Incremented each time the alarm buffer is modified. By entering this variable in a subscription, you can be notified each time a change is made to the alarm buffer.
- **EventCount**
  - Number of currently pending alarms
- **QuitAll**
  - Acknowledges all pending alarms

**Values array:**
- **Array with the currently pending alarms**
  - This array contains as many elements as are entered in EventCount.

Example:
```
ItemName="dev/Alarm.Version"
```

For a currently pending alarm, the **ItemName** syntax is:

```
ItemName="dev/Alarm.Values[ValueNumber].ArrayElement"
```

**ValueNumber:**
- Index of an alarm in the list of currently pending technological alarms

**ArrayElement:**
- **AlarmNo**
  - Alarm number
- **To**
  - Name the technology object that generated the alarm
- **Time**
  - Time of the alarm entry
- **Text**
  - Alarm text
- **Quit**
  - Acknowledges the alarm
- **Type**
  - Classification of the technological alarm:
    - ALARM
    - WARNING
    - INFORMATION

Example:
```
ItemName="dev/Alarm.Values[0].AlarmNo"
```
### 3.8.2.5 Changing the operating mode (V4.1 and higher)

For setting the operating mode, the **ItemName** syntax is:

```
ItemName="dev/Service.BZU.Variable"
```

**Variable:**
- **Value**
  Writing one of the following values changes the operating mode accordingly:
  - STOP
  - STOPU
  - RUN
- **State**
  Displays the execution states during an operating mode change
  The states change from IDLE to ACTIVE to READY.
- **Result**
  Shows the result of the operating mode change (when State = READY)
  Result = OK if the operating mode has been changed successfully. Otherwise, Result = Error

**Example:**
```
ItemName="dev/Service.BZU.Value"
```

### 3.8.2.6 RamToRom (V4.1 and higher)

For execution of **RamToRom**, the **ItemName** syntax is:

```
ItemName="dev/Service.RamToRom.Variable"
```

**Variable:**
- **Value**
  Save operation starts with Value = 0
- **State**
  Displays the status of the save operation
  The display starts with 0% and continues to 100%.
- **Result**
  Shows the result of the save operation (when State = 100%)
  Result = OK if the save operation has been completed successfully. Otherwise, Result = Error

**Example:**
```
ItemName="dev/Service.RamToRom.Value"
```
3.8.2.7 **ActiveToRam (V4.1 and higher)**

For execution of **ActiveToRam** (after changing the configuration data), the **itemName** syntax is:

```
itemName="dev/Service.ActToRam.Variable"
```

**Variable:**
- **Value**
  - Save operation starts with Value = 0
- **State**
  - Displays the status of the save operation
  - The display starts with 0% and continues to 100%.
- **Result**
  - Shows the result of the save operation (when State = 100%)
  - Result = OK if the save operation has been completed successfully. Otherwise, Result = Error ID

**Example:** `itemName=" dev/Service.ActToRam.Value"`

---

*Operation (software)*

*3.8 Variables providers*

---

SIMOTION IT Ethernet-based HMI and Diagnostic Functions

Diagnostics Manual, 11/2010

240
### 3.8.2.8 Accessing the global variables (V4.2 and higher)

The way to access the control's "global device variables" created by the user in SCOUT is via /glo/.

For the **global device variables**, the **ItemName** syntax is:

```
ItemName="glo/name"
```

The symbol information must be loaded to the control in order to make these variables visible. For this purpose, a check mark must be made under **Device > Properties > Settings** in SCOUT.

![Figure 3-88 'Global device variables' setting in the SCOUT device properties](image)

SIMOTION IT Ethernet-based HMI and Diagnostic Functions
Diagnostics Manual, 11/2010
241
3.8.2.9 Accessing the IO variables (V4.2 and higher)

There are three different ways to access the address list of the control's I/O variables that have been created in SCOUT:

- `/io/_direct/` addresses the direct I/O access (current value) of the I/O variables. This form of access is offered for all I/O variables.

- `/io/_image/` addresses the process image for the I/O variables. Only the I/O variables in address area 0-63 that have addresses in the format PI.../PQ..., are displayed. I/O symbols in the format %I... / %Q... are not currently addressable.

- `/io/_quality/` addresses the quality of I/O variables, i.e. the I/O status of the subslot (from HW Config) which contains this I/O variable. This is a 32-bit pattern. An overview of the possible bit pattern values can be found in the 'SIMOTION ST Structured Text' manual, in the section entitled 'Access to I/O variables (as of V4.2)'. The quality is the same for all I/O variables in a subslot. The quality is given as an integer for the individual I/O variables of the basic data types (BIT, BYTE, WORD, DWORD) and for arrays. It is not given for array elements (i.e. arrays cannot be expanded).

For the IO variables, the ItemName syntax is:

```
ItemName="io/_direct|_image|_quality/name"
```

The symbol information must be loaded to the control in order to make these variables visible. For this purpose, a check mark must be made under Device > Properties > Settings in SCOUT.
3.8.2.10 Accessing the AlarmS messages (V4.2 and higher)

How to access the AlarmS messages created by the user in SCOUT and triggered by the control.

For the **AlarmS messages**, the **ItemName** syntax is:

\[
\text{ItemName} = \text{"dev/Alarm.Values[ValueNumber].ArrayElement"}
\]

- **ValueNumber**: Index of an AlarmS in the list of currently pending technological alarms
- **ArrayElement**:  
  - AlarmNo: Alarm number  
  - AddInfo: Additional information  
  - Time: Time of AlarmS entry  
  - Text: AlarmS text  
  - Quit: Acknowledgement of AlarmS  
  - Type: S/SQ

Example:  
\[
\text{ItemName} = \text{"dev/Alarm.Values[0].AlarmNo"}
\]
3.8 Variables providers

3.8.3 SIMOTION diagnostics

3.8.3.1 Introduction

Access to diagnostics variables

The diagnostics variables of a SIMOTION control can be accessed via the "SIMOTION diagnostics" provider.

Most of the variables have read-only access and a few (e.g. operating mode) also have write access. All variables are of the string type. Therefore, numerical values are converted into strings by the provider.

The variable management area is dynamic and depends on the current configuration of the SIMOTION control. The provider supports browsing via OPC XML DA V1.0, meaning that the current variable management area can be viewed.

Variables groups of the "SIMOTION diagnostics" provider

The diagnostics variables of the "SIMOTION diagnostics" provider are combined into groups. A variable name is made up of the group name and variable name:

For example: Group.Variable

3.8.3.2 DeviceInfo group

General information on the SIMOTION device

The DeviceInfo group contains general information on the SIMOTION device. The 10 variables of this group are always available.

Table 3-16 Variables of the DeviceInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceInfo.Board</td>
<td>Specifies the system being used, read only</td>
</tr>
<tr>
<td>DeviceInfo.License-Serial-No</td>
<td>License serial number for this device, read-only</td>
</tr>
<tr>
<td>DeviceInfo.BZU</td>
<td>Access to the operating state, read and write, valid values for writing: STOP, STOPU, RUN</td>
</tr>
<tr>
<td>DeviceInfo.Systemtime</td>
<td>Access to the system time, read and write, the time must always be specified as in the following example: “Tue Aug 05 17:00:00 2003”; no other formats are accepted.</td>
</tr>
<tr>
<td>DeviceInfo.Timezone</td>
<td>Time offset in minutes, read and write, valid values are -720 to +720</td>
</tr>
<tr>
<td>DeviceInfo.Active-MAC</td>
<td>Active MAC address, read-only</td>
</tr>
<tr>
<td>DeviceInfo.Remanent-MAC</td>
<td>Retentive MAC address, read-only</td>
</tr>
<tr>
<td>DeviceInfo.IP-Address</td>
<td>IP configuration data (address, subnet mask and gateway), read-only</td>
</tr>
<tr>
<td>DeviceInfo.Subnet-Mask</td>
<td></td>
</tr>
<tr>
<td>DeviceInfo.Gateway</td>
<td></td>
</tr>
</tbody>
</table>
Additional variables of the DeviceInfo group

The following variables supply HTML color values ("#XXXXXX") which correspond to the colors of the DC5V, RUN, STOPU, and STOP LEDs of the SIMOTION device. It is, therefore, possible to display the operating mode as "traffic light information" via an HTML table (by means of the "background" attribute in the cells), for example, similar to the display in SIMOTION SCOUT, as for "Operating mode ...".

Access to these values is read-only.

Table 3-17 Variables of the DeviceInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceInfo.LEDColor.DC5V</td>
<td>Color for the DC5V LED; as the server can only be addressed when a voltage is applied, the corresponding HTML color is always green (&quot;#00FF00&quot;)</td>
</tr>
<tr>
<td>DeviceInfo.LEDColor.RUN</td>
<td>Color for the RUN LED; green in the RUN operating mode (&quot;#00FF00&quot;), otherwise gray (&quot;#C0C0C0&quot;)</td>
</tr>
<tr>
<td>DeviceInfo.LEDColor.STOPU</td>
<td>Color for the STOPU LED; amber in the STOPU operating mode (&quot;#FF9900&quot;), otherwise gray (&quot;#C0C0C0&quot;)</td>
</tr>
<tr>
<td>DeviceInfo.LEDColor.STOP</td>
<td>Color for the STOP LED; amber in the STOP operating mode (&quot;#FF9900&quot;), otherwise gray (&quot;#C0C0C0&quot;)</td>
</tr>
</tbody>
</table>

3.8.3.3 CompInfo group

This group supplies information about the components of the device. The number of variables varies in this group depending on the number of technology packages or additional hardware components.

Access to all variables is read-only.

Information on the CPU

The following variables supply information on the CPU:

Table 3-18 Variables of the CompInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompInfo.Cpu.MLFB</td>
<td>CPU MLFB / order number</td>
</tr>
<tr>
<td>CompInfo.Cpu.Serial-No</td>
<td>CPU serial number</td>
</tr>
<tr>
<td>CompInfo.Cpu.Revision-No</td>
<td>Revision number</td>
</tr>
<tr>
<td>CompInfo.Cpu_KERNELNAME</td>
<td>Kernel name</td>
</tr>
<tr>
<td>CompInfo.Cpu.Build-No</td>
<td>Build number</td>
</tr>
<tr>
<td>CompInfo.Cpu.User-Version</td>
<td>User version (firmware)</td>
</tr>
</tbody>
</table>

Information on the technology packages (TPs) and hardware

The number of available TPs or hardware components can be determined with the following variables.
Table 3-19 Variables of the CompInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompInfo.TP-Count</td>
<td>Number of available technology packages</td>
</tr>
<tr>
<td>CompInfo.HW-Count</td>
<td>Number of available hardware components</td>
</tr>
</tbody>
</table>

If TPs are available, information on the individual TPs can be obtained with CompInfo.TPx.Variable-Name (whereby x stands for the TP number).

The first TP is allocated the number 1 (not 0), for example: CompInfo.TP1.Name

The following information is available:

Table 3-20 Variables of the CompInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompInfo.TPx.Name</td>
<td>Name of the TP</td>
</tr>
<tr>
<td>CompInfo.TPx.User-Version</td>
<td>User version of the TP</td>
</tr>
<tr>
<td>CompInfo.TPx.Build-No</td>
<td>Build number of the TP</td>
</tr>
</tbody>
</table>

If additional hardware components are available, information on the individual hardware components can be obtained with CompInfo.HWx.Variable-Name (whereby x stands for the HW number).

The first hardware component is allocated the number 1 (not 0), for example: CompInfo.HW1.MLFB

The following information is available:

Table 3-21 Variables of the CompInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompInfo.HWx.MLFB</td>
<td>MLFB / order number</td>
</tr>
<tr>
<td>CompInfo.HWx.Serial-No</td>
<td>Serial number</td>
</tr>
<tr>
<td>CompInfo.HWx.Revision-No</td>
<td>Revision number</td>
</tr>
<tr>
<td>CompInfo.HWx.Firmwarename</td>
<td>Firmware name</td>
</tr>
<tr>
<td>CompInfo.HWx.Build-No</td>
<td>Build number</td>
</tr>
<tr>
<td>CompInfo.HWx.User-Version</td>
<td>User version</td>
</tr>
</tbody>
</table>
As the information is dynamic and the scope is not known beforehand, the following variables also exist to simplify the display of hardware components and TPs in HTML:

Table 3-22 Variables of the CompInfo group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompInfo.TableHead.TP</td>
<td>Supplies the header of an HTML table with all the information about the TPs, e.g. &quot;&lt;tr&gt;&lt;th&gt;TP Name&lt;/th&gt;&lt;th&gt;User Ver.&lt;/th&gt;&lt;th&gt;Build No.&lt;/th&gt;&lt;/tr&gt;&quot;</td>
</tr>
<tr>
<td>CompInfo.Table.TP</td>
<td>Supplies an HTML table with all the information about all the available TPs</td>
</tr>
<tr>
<td>CompInfo.TableHead.HW</td>
<td>Supplies the header of an HTML table with all the information about the hardware components, e.g. &quot;&lt;tr&gt;&lt;th&gt;MLFB&lt;/th&gt;&lt;th&gt;Serial No.&lt;/th&gt;&lt;th&gt;Revision No.&lt;/th&gt;&lt;th&gt;Firmware Name&lt;/th&gt;&lt;th&gt;User Ver.&lt;/th&gt;&lt;th&gt;Build No.&lt;/th&gt;&lt;/tr&gt;&quot;</td>
</tr>
<tr>
<td>CompInfo.Table.HW</td>
<td>Supplies an HTML table with all the information about all the available hardware components</td>
</tr>
</tbody>
</table>

Note
Separate access to the table and the table header enables separate formatting.

3.8.3.4 CPULoad group

Information on CPU load

The CPULoad group supplies information on the load of the CPU. Access to all variables is read-only.

Table 3-23 Variables of the CPULoad group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPULoad.Percent</td>
<td>CPU load in percent</td>
</tr>
<tr>
<td>CPULoad.Mintime</td>
<td>Minimum runtime of the BackgroundTask (free cycle) in ms with 5 decimal places</td>
</tr>
<tr>
<td>CPULoad.Acttime</td>
<td>Actual runtime of the BackgroundTask (free cycle) in ms with 5 decimal places</td>
</tr>
<tr>
<td>CPULoad.Maxtime</td>
<td>Maximum runtime of the BackgroundTask (free cycle) in ms with 5 decimal places</td>
</tr>
</tbody>
</table>
3.8 Variables providers

3.8.3.5 MemoryLoad group

Information on memory load

The MemoryLoad group provides information about the load on memory devices in bytes or as a percentage. Variables can only be accessed in read-only mode.

Table 3-24 Variables of the MemoryLoad group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryLoad.Flash-Size</td>
<td>Size of the Flash memory</td>
</tr>
<tr>
<td>MemoryLoad.Flash-Used</td>
<td>Currently occupied flash memory</td>
</tr>
<tr>
<td>MemoryLoad.RAM-Size</td>
<td>Size of the RAM</td>
</tr>
<tr>
<td>MemoryLoad.RAM-Used</td>
<td>Currently occupied RAM</td>
</tr>
<tr>
<td>MemoryLoad.RAMDisk-Size</td>
<td>Size of the RAM disk</td>
</tr>
<tr>
<td>MemoryLoad.RAMDisk-Used</td>
<td>Currently occupied RAM disk memory</td>
</tr>
<tr>
<td>MemoryLoad.Remanent-Size</td>
<td>Size of the retentive memory</td>
</tr>
<tr>
<td>MemoryLoad.Remanent-Used</td>
<td>Currently occupied retentive memory.</td>
</tr>
<tr>
<td>MemoryLoad.Flash-Percent</td>
<td>Percentage of external Flash memory used</td>
</tr>
<tr>
<td>MemoryLoad.RAM-Percent</td>
<td>Percentage of RAM memory used</td>
</tr>
<tr>
<td>MemoryLoad.RAMDisk-Percent</td>
<td>Percentage of RAM disk used</td>
</tr>
<tr>
<td>MemoryLoad.Remanent-Percent</td>
<td>Percentage of internal Flash memory used</td>
</tr>
</tbody>
</table>

3.8.3.6 TaskRT group

Variables of the TaskRT group

The TaskRT group supplies information about the task runtimes and the task states of the SIMOTION device. The same values are supplied as in the SIMOTION SCOUT under device diagnostics, task runtimes. Access to all values is read-only. The number of variables varies and depends on the current configuration of the execution system in SIMOTION SCOUT.

Table 3-25 Variables of the TaskRT group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskRT.TaskCnt</td>
<td>Supplies the number of currently available tasks</td>
</tr>
</tbody>
</table>

Task names

The following information can be obtained for the individual tasks via TaskRT.Task-name.Variable-Name. The tasks have the same name in IT DIAG and SCOUT.

The same information can be obtained for every task; here is an example of the first MotionTask.
Example:

TaskRT.MotionTask_1.Status

Current task status, can be an appropriate combination of the following values:
STOP_PENDING, STOPPED, RUNNING, STOP_UNCOND, WAITING, SUSPENDED,
WAITING_FOR_NEXT_CYCLE, WAITING_FOR_NEXT_INTERRUPT, LOCKED,
SUSPENDED_BY_DEBUG_MODE

Additional variables of the TaskRT group

Table 3-26 Variables of the TaskRT group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskRT.MotionTask_1.Actual</td>
<td>Current runtime of the task in ms, with 5 decimal places</td>
</tr>
<tr>
<td>TaskRT.MotionTask_1.Min</td>
<td>Minimum runtime of the task in ms, with 5 decimal places</td>
</tr>
<tr>
<td>TaskRT.MotionTask_1.Max</td>
<td>Maximum runtime of the task in ms, with 5 decimal places</td>
</tr>
<tr>
<td>TaskRT.MotionTask_1.Average</td>
<td>Average runtime of the task in ms, with 5 decimal places</td>
</tr>
</tbody>
</table>

As the information is dynamic and the scope is not known beforehand, the following
variables also exist to simplify the display of task information in HTML:

Table 3-27 Variables of the TaskRT group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaskRT.TableHead</td>
<td>Supplies the header of an HTML table with all the information about the tasks,</td>
</tr>
<tr>
<td></td>
<td>e.g. &quot;&lt;tr&gt;&lt;th&gt;Taskname&lt;/th&gt;&lt;th&gt;Status&lt;/th&gt;&lt;th&gt;Actual&lt;/th&gt;&lt;th&gt;Min&lt;/th&gt;&lt;th&gt;Max&lt;/th&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;th&gt;Average&lt;/th&gt;&lt;/tr&gt; &quot;</td>
</tr>
<tr>
<td>TaskRT.Table</td>
<td>Supplies an HTML table with all the information about the available tasks; all runtime values are entered with the unit as, unlike the individual value query, they can vary between s and ms. Three decimal places are displayed.</td>
</tr>
</tbody>
</table>
3.8.3.7 DiagBuffer group

The DiagBuffer group supplies information about the events in the DiagBuffer. Access to all variables is read-only.

Events can be output in English, French, German, Italian, and Spanish text.

Requirements

Text is output in English by default. To display event text in a different language, a file in the relevant language must be downloaded to the SIMOTION control memory card.

<table>
<thead>
<tr>
<th>Language</th>
<th>File name</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>DGBUFTXT-EN.EDB</td>
</tr>
<tr>
<td>German</td>
<td>DGBUFTXT-DE.EDB</td>
</tr>
<tr>
<td>French</td>
<td>DGBUFTXT-FR.EDB</td>
</tr>
<tr>
<td>Italian</td>
<td>DGBUFTXT-IT.EDB</td>
</tr>
<tr>
<td>Spanish</td>
<td>DGBUFTXT-ES.EDB</td>
</tr>
</tbody>
</table>

Language-specific file names of the DiagBuffer texts

Procedure

1. Open the \3_Diag_Buf_Messages\Diag_Buf_Messages directory on the SIMOTION IT DIAG DVD.
2. Insert the SIMOTION control memory card in a reader/writer.
3. Copy the DGBUFTXT-XX.EDB file for the required language into the \USER\SIMOTION\HMICFG directory. You must create the directory if it does not already exist.
4. Insert the memory card in the SIMOTION device again.

Procedure for the SIMOTION P350

1. Shut down the SIMOTION P control.
2. Open the AddOn\4_Accessories\Simotion_IT\3_Diag_Buf_Messages\Diag_Buf_Messages directory on the SIMOTION SCOUT Add-Ons DVD.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the F:\SIMOTION\USER\CARD\USER\SIMOTION\HMICFG directory (for the default installation).
4. Start the SIMOTION P control.
Procedure for the SIMOTION P320

1. Shut down the SIMOTION P control.
2. Open the AddOn\4_Accessories\Simotion_IT\3_Diag_Buf_Messages\Diag_Buf_Messages directory on the SIMOTION SCOUT Add-Ons DVD.
3. Copy the DGBUFTXT-XX.EDB file for the required language to the D:\Card\USER\SIMOTION\HMICFG directory (for the default installation).
4. Start the SIMOTION P control.

**Note**

Only one language can be stored on the SIMOTION control at any given time.

On delivery and following a firmware update, the English version will be present on the device in all cases.

For reasons of compatibility, a DGBUFTXT.EDB file is recognized, even if no DGBUFTXT-XX.EDB file is found. If both files are present, priority is given to DGBUFTXT-XX.EDB.

### Variables of the DiagBuffer group

The following variables are available for enhancing the display:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagBuffer.TableHead</td>
<td>Supplies the header of an HTML table with all events. The contents are: &lt;tr&gt;&lt;th&gt;Nr&lt;/th&gt;&lt;th&gt;Time&lt;/th&gt;&lt;th&gt;Date&lt;/th&gt;&lt;th&gt;Event&lt;/th&gt;/tr&gt;</td>
</tr>
<tr>
<td>DiagBuffer.Table</td>
<td>Supplies the contents of an HTML table with all events. The structure of each row is as follows: &lt;tr&gt;&lt;td&gt;NUMBER&lt;/td&gt;&lt;td&gt;TIME&lt;/td&gt;&lt;td&gt;DATE&lt;/td&gt;&lt;td&gt;EVENT&lt;/td&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td></td>
<td>Note: The NUMBER, TIME, DATE, and EVENT texts specified in this format are replaced with the corresponding value of each event.</td>
</tr>
<tr>
<td>DiagBuffer.ExtendedTable</td>
<td>Supplies the contents of the HTML table with all events, including the extended entries displayed via the Info button.</td>
</tr>
<tr>
<td>DiagBuffer.ExtendedBufferJScript</td>
<td>Supplies the dynamically generated JavaScript fragment required to display the table.</td>
</tr>
<tr>
<td>DiagBuffer.LText[]</td>
<td>Supplies an array that enables access to the entire text of the diagnostics buffer entry. The index matches the index of the diagnostics buffer entry. The individual elements of a diagnostics buffer entry (time, date, text, extended entry text) are separated by &quot;/@@/&quot;.</td>
</tr>
</tbody>
</table>
The following variables can be used for direct access to the data of certain events in the diagnostics buffer:

Table 3-29 Variables of the DiagBuffer group - direct access

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagBuffer.EventCnt</td>
<td>Number of events currently in the diagnostics buffer</td>
</tr>
<tr>
<td>DiagBuffer.CplEventCnt</td>
<td>Event counter beyond the circular buffer limit</td>
</tr>
<tr>
<td></td>
<td>During ramp-up, the buffer is initialized with the current number of diagnostics buffer entries. Each time an entry is made, the value is incremented, even beyond the maximum number of diagnostics buffer entries.</td>
</tr>
<tr>
<td>DiagBuffer.Time_1 bis DiagBuffer.Time_n</td>
<td>Time of each event</td>
</tr>
<tr>
<td>DiagBuffer.Date_1 bis DiagBuffer.Date_n</td>
<td>Date of each event</td>
</tr>
<tr>
<td>DiagBuffer.Text_1 bis DiagBuffer.Text_n</td>
<td>Text of each event</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If the event text number and its parameters cannot be resolved, the number and parameters are output in HEX format. The variable in HEX format is a string of 20 hexadecimal characters (without separators).</td>
</tr>
</tbody>
</table>

Example of an HTML page

```html
<html>
<head>
  <title>SIMOTION <%=DeviceInfo.Board%> - Diagnostics</title>
  <script type="text/javascript">
    <%=DiagBuffer.ExtendedBufferJScript%>
  </script>
</head>
<body style="font-family: Arial">
  <h2>Diag Buffer (extended)</h2>
  <table border="2" cellspacing="1" cellpadding="5">
    <font size="4">
      <%=DiagBuffer.TableHead%>
      <%=DiagBuffer.ExtendedTable%>
    </font>
  </table>
</body>
</html>
```
3.8 Variables providers

DiagBufferDrv group

The DiagBufferDrv group provides information about the drive diagnostics buffer. Access to all variables is read-only.

Variables of the DiagBufferDrv group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagBufferDrv.TableHead</td>
<td>Supplies the header of an HTML table with all events. The contents are:</td>
</tr>
<tr>
<td></td>
<td>&lt;tr&gt;&lt;th&gt;Nr&lt;/th&gt;&lt;th&gt;Time&lt;/th&gt;&lt;th&gt;Date&lt;/th&gt;&lt;th&gt;Event&lt;/th&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td>DiagBufferDrv.Table</td>
<td>Supplies the contents of an HTML table with all events. The structure of each row is as follows:</td>
</tr>
<tr>
<td></td>
<td>&lt;tr&gt;&lt;td&gt;NUMBER&lt;/td&gt;&lt;td&gt;TIME&lt;/td&gt;&lt;td&gt;DATE&lt;/td&gt;&lt;td&gt;EVENT&lt;/td&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The NUMBER, TIME, DATE, and EVENT texts specified in this format are replaced with the corresponding value of each event.</td>
</tr>
<tr>
<td>DiagBufferDrv.ExtendedTable</td>
<td>Supplies the contents of the HTML table with all events, including the extended entries displayed via the Info button.</td>
</tr>
<tr>
<td>DiagBufferDrv.ExtendedBufferJavaScript</td>
<td>Supplies the dynamically generated JavaScript fragment required to display the table.</td>
</tr>
<tr>
<td>DiagBufferDrv.LText[]</td>
<td>Supplies an array that enables access to the entire text of the diagnostics buffer entry. The index matches the index of the diagnostics buffer entry. The individual elements of a diagnostics buffer entry (time, date, text, extended entry text) are separated by &quot;/@@/&quot;.</td>
</tr>
</tbody>
</table>
The following variables can be used for direct access to the data of certain events in the drive diagnostics buffer:

### Table 3-30 Variables of the DiagBufferDrv group - direct access

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiagBufferDrv.EventCnt</td>
<td>Number of events currently in the drive diagnostics buffer</td>
</tr>
<tr>
<td>DiagBufferDrv.CplEventCnt</td>
<td>Event counter beyond the circular buffer limit. During ramp-up, the counter is initialized with the current number of drive diagnostics buffer entries. Each time an entry is made, the value is incremented, even beyond the maximum number of drive diagnostics buffer entries.</td>
</tr>
<tr>
<td>DiagBufferDrv.Date[1] bis DiagBufferDrv.Date[n]</td>
<td>Date of each event</td>
</tr>
<tr>
<td>DiagBufferDrv.Text[1] bis DiagBufferDrv.Text[n]</td>
<td>Text of each event <strong>Note</strong>: If the event text number and its parameters cannot be resolved, the number and parameters are output in HEX format. The variable in HEX format is a string of 20 hexadecimal characters (without separators).</td>
</tr>
</tbody>
</table>

### 3.8.3.9 Alarms group

#### Information about alarm table

The Alarms group provides information about the pending alarms. Access to all variables is read-only.

### Table 3-31 Variables of the Alarms group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarms.AlarmCnt</td>
<td>Number of alarms</td>
</tr>
<tr>
<td>Alarms.Table</td>
<td>HTML table with all pending alarms</td>
</tr>
<tr>
<td>Alarms.TableHead</td>
<td>Table header for the HTML table of pending alarms</td>
</tr>
<tr>
<td>Alarms.TableHeadBuffer</td>
<td>HTML table (header only) of the alarm buffer</td>
</tr>
<tr>
<td>Alarms.TableHeadUser</td>
<td>HTML table (header only) of the AlarmS</td>
</tr>
<tr>
<td>Alarms.TableBodyBuffer</td>
<td>HTML table (content only) of the alarm buffer</td>
</tr>
<tr>
<td>Alarms.TableBodyUser</td>
<td>HTML table (content only) of the AlarmS</td>
</tr>
<tr>
<td>Alarms.TableBuffer</td>
<td>HTML table of the alarm buffer</td>
</tr>
<tr>
<td>Alarms.UserAlarmCnt</td>
<td>Number of AlarmS</td>
</tr>
</tbody>
</table>
3.8.3.10 AlarmsDrv group

Information about drive alarm table

The AlarmsDrv group provides information about the pending drive alarms. Access to all variables is read-only.

Table 3-32 Variables of the AlarmsDrv group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmsDrv.AlarmCnt</td>
<td>Number of drive alarms</td>
</tr>
<tr>
<td>AlarmsDrv.Table</td>
<td>HTML table with all pending drive alarms</td>
</tr>
<tr>
<td>AlarmsDrv.TableHead</td>
<td>Table header for the HTML table of pending drive alarms</td>
</tr>
<tr>
<td>AlarmsDrv.TableHeadBuffer</td>
<td>HTML table (header only) of the drive alarm buffer</td>
</tr>
<tr>
<td>AlarmsDrv.TableHeaderUser</td>
<td>HTML table (header only) of the drive alarms</td>
</tr>
<tr>
<td>AlarmsDrv.TableBodyBuffer</td>
<td>HTML table (content only) of the drive alarm buffer</td>
</tr>
<tr>
<td>AlarmsDrv.TableBodyUser</td>
<td>HTML table (content only) of the drive alarms</td>
</tr>
<tr>
<td>AlarmsDrv.TableBuffer</td>
<td>HTML table of the drive alarm buffer</td>
</tr>
<tr>
<td>AlarmsDrv.UserAlarmCnt</td>
<td>Number of drive alarms which are AlarmS</td>
</tr>
</tbody>
</table>

3.8.3.11 ActiveTraces group

Variables of the ActiveTraces group

The ActiveTraces group returns the number of active traces and a list of the active traces. Access to all variables is read-only.

Table 3-33 Variables of the ActiveTraces group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveTraces.TraceCnt</td>
<td>Number of active traces</td>
</tr>
<tr>
<td>ActiveTraces.TableHead</td>
<td>Supplies the header of an HTML table with all active traces. The contents are:&lt;tr&gt;&lt;th&gt;Name&lt;/th&gt;&lt;th&gt;State&lt;/th&gt;&lt;/tr&gt;</td>
</tr>
<tr>
<td>ActiveTraces.Table</td>
<td>Supplies the contents of an HTML table with all active traces. The structure of each row is as follows: &lt;tr&gt;&lt;td&gt;NAME&lt;/td&gt;&lt;td&gt;STATE&lt;/td&gt;&lt;/tr&gt; Note: The NAME and STATE placeholders specified in the format are replaced with the corresponding value of each trace.</td>
</tr>
</tbody>
</table>
3.8 Variables providers

3.8.3.12 Watch group

Variables of the Watch group

The Watch group provides access to saved watch tables. Access to all variables is read-only.

Table 3-34 Variables of the Watch group

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Watch.TableNames</td>
<td>List of watch table names separated by commas</td>
</tr>
<tr>
<td>Watch.TableHead</td>
<td>Table header for the HTML table of a watch table</td>
</tr>
<tr>
<td>Watch.TablesCount</td>
<td>Number of watch tables</td>
</tr>
<tr>
<td>Watch.Tables.TableName.csv</td>
<td>Exports the specified watch table (TableName) as CSV file</td>
</tr>
<tr>
<td>Watch.Tables.TableName.xml</td>
<td>Exports the specified watch table (TableName) as an XML file for transfer to other controls</td>
</tr>
<tr>
<td>Watch.Tables.TableName.html</td>
<td>Provides the specified watch table (TableName) in HTML format</td>
</tr>
</tbody>
</table>
3.8.3.13 Comparison with the device diagnostics of SIMOTION SCOUT

Comparison with device diagnostics in SIMOTION SCOUT

The variables described in this chapter are based on the view of the device diagnostics in SIMOTION SCOUT. The following figures show the connection between the "SIMOTION diagnostics" variables and the device diagnostics in SIMOTION SCOUT.

Figure 3-90 "General" device diagnostics
3.8 Variables providers

MemoryLoad.RAM-Size
MemoryLoad.RAM-Disk-Size
MemoryLoad.RAM-Used
MemoryLoad.RAM-Disk-Used

Figure 3-91 "System load" device diagnostics

TaskRT.MotionTask_11.Status
TaskRT.MotionTask_11.Actual
TaskRT.MotionTask_11.Min
TaskRT.MotionTask_11.Average
TaskRT.MotionTask_11.Max

Figure 3-92 "Task runtimes" device diagnostics
3.8.4 UserConfig

3.8.4.1 User-defined variables

The user-defined variables can be declared in the WebCfg.xml (Page 270) file and read in the variable provider.

For the variable provider, the ItemName syntax is:

```
ItemName=" constants/VariableName"
```

Some constant variables are preinstalled in IT DIAG:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceUserMsgLanguageID</td>
<td>Integer (LCID)</td>
<td>Specifies the language to be used when importing user-defined messages (diagnostics buffer or AlarmS). Setting the language for AlarmS and user-defined diagnostics buffer messages (Page 22)</td>
</tr>
<tr>
<td>WatchWritable</td>
<td>YES/NO default: YES</td>
<td>Specifies whether watch tables can be edited and deleted on the standard pages.</td>
</tr>
<tr>
<td>BasicWatchWritable</td>
<td>YES/NO default: YES</td>
<td>Specifies whether watch tables can be edited and deleted on the basic pages.</td>
</tr>
<tr>
<td>UserArea</td>
<td>Character string: Embedded, EmbeddedSimple, StandAlone</td>
<td>User's Area display mode. Embedded, user-defined pages (Page 122)</td>
</tr>
<tr>
<td>UserDir</td>
<td>Character string</td>
<td>Directory for user pages: &quot;/FILES&quot; + &lt;UserDir&gt;</td>
</tr>
</tbody>
</table>

Overview of preinstalled constant variables

See also

IT DIAG configuration data (Page 78)
3.9 Secure Socket Layer

Introduction

The Secure Socket Layer protocol (SSL) enables encrypted data transfer between a client and SIMOTION. HTTPS access between the browser and the SIMOTION control is based on the Secure Socket Layer protocol.

Encrypted access to SIMOTION can take place via both SIMOTION IT OPC XML DA and SIMOTION IT DIAG.

This section tells you which steps you need to follow to enable encrypted data communication between a client and SIMOTION. The possibilities are as follows:

1. You have a Certification Authority (CA) in your organization and the necessary key files are available. In this case, "Transferring key files to SIMOTION" is the relevant section.

2. You do not have a CA in your organization. In this case, you will need to create the key files yourself. An example of how to do this is described in Creating key files (Page 262). When you have created the key files, they will need to be transferred to SIMOTION.

Note

HTTPS connections are supported in SIMOTION V3.2 and higher.
3.9 Secure Socket Layer

3.9.1 Key files

Encryption methods
You need two key files for the encryption method used by the Secure Socket Layer protocol. You need a public certificate and a private key. The pair of keys is created individually for each SIMOTION control. This ensures that the address requested matches the SIMOTION control accessed during HTTPS communication.

Note
Encrypted access to the SIMOTION control is only possible with the control identifier (name/IP address) specified when the key was created.

You can find further information about Secure Socket Layer certificates at http://www.verisign.de.

As delivered
To ensure that you can access the SIMOTION control via HTTPS with the as-delivered version of SIMOTION IT DIAG (standard diagnostic pages), the system is supplied with two key files already integrated in the WebCfg.xml file.

When you attempt HTTPS access using the key files supplied with the system, you will be warned that the certificate is unknown and that the current address of the control does not match the name of the control in the certificate.

3.9.2 Transferring the key files to the SIMOTION control
To enable IT DIAG access to the SIMOTION device via HTTPS, you will need to transfer the key files "MWSSLCer.pem" and "MWSSLKey.pem" to the memory card of the SIMOTION control.

In order to copy the key files onto the memory card you need a memory card reader/writer.

The procedure for transferring key files to the SIMOTION P350 control is described separately.

Procedure

1. Insert the SIMOTION control memory card in the card reader/writer.

2. Copy the "MWSSLCer.pem" and "MWSSLKey.pem" files on the memory card to directory \USER\SIMOTION\HMICFG. You must create the directory if it does not already exist.

3. Insert the memory card in the SIMOTION control and switch the control on.

Error-free OPC XML DA access via HTTPS is available after the SIMOTION control ramps up.
3.9 Secure Socket Layer

3.9.3 Creating key files with script (V4.1 and higher)

Overview

Note
HTTPS connections are supported in SIMOTION V3.2 and higher.

If no Certification Authority (CA) is available in your organization, we recommend that you follow the steps described in the following section. The certificate and the key files are created with the OpenSSL tool and a Perl script.

Carry out the following steps:

<table>
<thead>
<tr>
<th>No.</th>
<th>Working step</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Install the Perl runtime environment</td>
<td>If Perl is not installed</td>
</tr>
<tr>
<td>2.</td>
<td>Install OpenSSL</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Create the certificate and key files</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with Perl script</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Copy WebCfg.xml to this folder</td>
<td>In SIMOTION SCOUT V4.1, for example, a standard WebCfg.xml can be found in the AddOn\4_Accessories\Simotion_IT2_Configuration folder on the &quot;SIMOTION SCOUT Add-On&quot; DVD.</td>
</tr>
<tr>
<td>5.</td>
<td>Execute the Perl script</td>
<td>Call the SIMOTION Perl script with the relevant options.</td>
</tr>
<tr>
<td>6.</td>
<td>Download the created WebCfg.xml file to the control with a browser</td>
<td>This step must be performed once for each control.</td>
</tr>
<tr>
<td>7.</td>
<td>Import the created certificate to the PC browser</td>
<td>This step must be performed once for each PC.</td>
</tr>
</tbody>
</table>

HTTPS access is available after the SIMOTION control ramps up.
Installing the Perl runtime environment

Install Perl if the Perl runtime environment is not present on your PC. You can download a free setup for Windows from the following websites, for example:

- http://www.activestate.com
- http://www.perl.org

Installing OpenSSL

You can download a free setup for Windows from the following websites, for example:

- http://openssl.org

Note

It is assumed that OpenSSL is installed after C:\OpenSSL. If a different path has been selected, line 5 of the Perl script must be adapted accordingly.

Creating the certificate and key files with Perl script

- Create a folder with any name on your local drive, e.g., "c:\SimotionSSL".
- Copy the Perl script file "cert.pl" to the folder you created.
  You can find the Perl script file in the AddOn\4_Accessories\Simotion_IT\6_Tools folder on the "SIMOTION SCOUT Add-Ons" DVD.
- Copy a standard "WebCfg.xml" file to the folder you created.
  You can find a standard template of the "WebCfg.xml" file on the "Utilities & Applications" DVD or you can use the one that is created automatically on the memory card when you ramp up the control.
- Execute the Perl script with the following options:
  "perl cert.pl -c <IP address> -p"

A CA is created, after which a server key is generated and the certificate is signed.

The following files are saved in the folder (e.g. "c:\SimotionSSL"): 
- "c:\SimotionSSL\CA\cakey.pem"
- "c:\SimotionSSL\CA\cacert.pem"
- "c:\SimotionSSL\out\<IP address>\MWSSLKey.pem"
- "c:\SimotionSSL\out\<IP address>\MWSSLCert.pem"
- "c:\SimotionSSL\out\<IP address>\WebCfg.xml"

Note

For help with calling, use option -h: "perl cert.pl -h"
• Use a browser to download the created "WebCfg.xml" file to the control.
  The WebCfg.xml file contains the pair of keys (server key and certificate) for the HTML
  server of the control.
• Import the certificate to the browser
  The cacert.pem certificate can be made known to the PC by importing it in the browser. If
  the certificate is not imported, a message indicating that the signed CA is unknown is
  displayed when you open the browser.
  See also Importing the certificate into the browser (Page 264).

3.9.3.1 Importing the certificate into the browser
If you use SSL with your own certification authority, you will need to prepare your PCs for
communication with the SIMOTION control. To do this, the "SIMOTION.cer" certificate must
be included in the list of root certificates.

Procedure
You first need to edit the cacert.pem file as follows:
1. Copy the "cacert.pem" file (in the example in "My Documents\OpenSSL\demoCA").
2. Insert the copied file in "My Documents\OpenSSL\", for example.
3. Rename the file to "SIMOTION.cer".
   Please follow the instructions in your browser when importing the certificate.
### List of abbreviations

#### 4.1 List of abbreviations

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>Certification Authority</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CSV</td>
<td>Character Separated Values</td>
</tr>
<tr>
<td>DO</td>
<td>Drive Object (Drive object)</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>ECMA</td>
<td>European Computer Manufacturers Association</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GMT</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Secure HTTP</td>
</tr>
<tr>
<td>JS</td>
<td>Javascript</td>
</tr>
<tr>
<td>MWSL</td>
<td>MiniWeb Server Language</td>
</tr>
<tr>
<td>OPC</td>
<td>Denotes a standard interface for communication in automation technology. <a href="http://www.opcfoundation.org/">http://www.opcfoundation.org/</a></td>
</tr>
<tr>
<td>OPC XML-DA</td>
<td>OPC XML Data Access</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>TO</td>
<td>Technology Object (Technology object)</td>
</tr>
<tr>
<td>TVS</td>
<td>Trace Via SOAP</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UTC</td>
<td>Universal Time Coordinated</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSL</td>
<td>Extensible Stylesheet Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>XSL Transformation</td>
</tr>
</tbody>
</table>
List of abbreviations

4.1 List of abbreviations
5.1 WebCfg.xml

5.1.1 <ALTERNATE_PORTNUMBER>

Tag

Additional port for requests for the web server.
Every TCP/IP server (or service) has what is known as a well-known port number which can be used by a client to address it. For the web server, this is normally port number 80.
The web server can also "listen" to a second port number. If a request comes on this port number, the WEB_FLAG_ALTERNATIVE_PORT flag is set in the ECB.
For example, by adding a firewall you can establish a firewall-controlled security concept. Another application of this alternative port uses the DAV module to detect whether a request is a DAV request or a web request.

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
[...]
<BASE>
[...]
</BASE>
<SERVEROPTIONS>
  <ALTERNATE_PORTNUMBER>
    81
  </ALTERNATE_PORTNUMBER>
[...]
</SERVEROPTIONS>
[...]
</SERVERPAGES>
```

In this example, the alternative port number of the web server is set to 81.

See also

Overview (Page 104)
5.1.2 <ALTERNATE_SSL_PORTNUMBER>

Tag

For the SSL protocol (Secure Socket Layer), an additional well-known port number is needed. This is normally port number 443. The web server can also "listen" to a second port number. If a request comes on this port number, the MWEB_FLAG_ALTERNATIVE_PORT flag is set in the ECB. For example, by adding a firewall you can establish a firewall-controlled security concept. Another application of this alternative port uses the DAV module to detect whether a request is a DAV request or a web request. This is the alternative SSL port.

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]  
    <BASE>
        [...]  
        <SERVEROPTIONS>
            <ALTERNATE_SSL_PORTNUMBER>
                5443
            </ALTERNATE_SSL_PORTNUMBER>
            [...]  
            </SERVEROPTIONS>
            [...]  
        </BASE>
    [...]  
</SERVERPAGES>
```

In this example, the alternative port number for SSL is set to 5443.

5.1.3 <BASE>

Tag

The link lists for user-defined HTML pages are stored in the <BASE> tag of WebCfg.xml.

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]  
    <BASE LOCALLINK="/"  
        [...]  
        <index.mbs LOCALLINK="mydir/index.mbs" PREFER_EXTERNAL="TRUE"/>
        [...]  
    </BASE>
    [...]  
</SERVERPAGES>
```
5.1.4  <BROWSEABLE>

<table>
<thead>
<tr>
<th>Tag Values</th>
<th>&lt;BROWSEABLE&gt;</th>
<th>TRUE, FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enable and disable browsing and displaying of directories.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This tag can be used to allow browsing globally for all directories. In this case, the individual BROWSEABLE attributes are of no relevance for the LOCALLINKS.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        <BROWSEABLE>
            FALSE
        </BROWSEABLE>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```

In this example, global browsing is disabled and can be explicitly enabled for individual links. This is the default behavior.

**See also**

Browsing of directories (Page 109)
5.1.5 <CONFIGURATION_DATA>

Tag | <CONFIGURATION_DATA>
--- | ---

Each module provides the option of defining module-specific configuration data within this tag. These data can be read out again using the default service. The format of the individual items of configuration data depends exclusively on the modules. Therefore, it cannot be described in general terms.

Example

```xml
<SERVERPAGES>
    [...]
    <CONFIGURATION_DATA>
    <USERCONFIG>
        <UserArea>EmbeddedSimple</UserArea>
        <UserDir/>
        <IncludeScriptsDirectly>NO</IncludeScriptsDirectly>
    </USERCONFIG>
    <!-- Add your constants here -->
    <ForceUserMsgLanguageID>1031</ForceUserMsgLanguageID>
    </USERCONFIG>
    </CONFIGURATION_DATA>
    [...]
</SERVERPAGES>
```

5.1.6 <DEFAULTDOCUMENT>

Tag | <DEFAULTDOCUMENT>
--- | ---

Specification of the document that is to be displayed if the URL received from the browser does not contain explicit page information. This is often called Default.mcs or Index.mcs. There can be only one default document. If no default document is found and file browsing is permitted, the directory itself is returned.

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        <DEFAULTDOCUMENT>
            Default.mcs
        </DEFAULTDOCUMENT>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```

If, for example, the URL http://Servername/MyDir is used to query a directory, the web server appends the file name "Default.mcs" to the URL (http://Servername/MyDir/Default.mcs) and then attempts to resolve the URL:

- If this succeeds, Default.mcs is returned to the client.
- If this is not successful, either a directory view is returned or an HTTP 404 "Not Found" error message is issued (depending on configuration).
## 5.1.7 `<MIME_TYPES>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;MIME_TYPES&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>With the MIME type table, the web server offers the option of mapping the file extension of a particular file to an associated MIME type.</td>
<td></td>
</tr>
<tr>
<td>In addition, a different icon can be saved for each file extension for the directory browser.</td>
<td></td>
</tr>
</tbody>
</table>

| Explanation | The content of a file is designated in the file system by its extension (e.g. "txt" for text files). This type of extension is not mandatory in a transport protocol such as HTTP. For this reason, an HTTP header named "MIME type" is inserted, which contains this information about the content type. |

| Example | `<xml version="1.0" standalone="yes"?>
<SERVERPAGES>
[...]
<BASE>
[...]
</BASE>
<SERVEROPTIONS>
  <MIME_TYPES>
    <FILE EXTENSION="htm" MIMETYPE="text/html" ICON="/Images/www.gif" FILTER="TRUE"/>
    <FILE EXTENSION="html" MIMETYPE="text/html" ICON="/Images/www.gif" FILTER="FALSE"/>
    [...]
  </MIME_TYPES>
  [...]
</SERVEROPTIONS>
[...]
</SERVERPAGES>` |

For the "htm" and "html" extensions, the MIME type "text/html" is specified. The icon with the URL "!/Images/www.gif" is used to designate this data type in the directory browser. The FILTER attribute can be used to specify whether or not files with this extension are filtered. For more information about MIME types, refer to the RFCs 2045 ff.
## 5.1.8 `<PORTNUMBER>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;PORTNUMBER&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Every TCP/IP server (or service) has what is known as a well-known port number which can be used by a client to address it. For the web server, this is normally port number 80. This port number can be set in the <code>&lt;PORTNUMBER&gt;</code> tag. If nothing is set, the number 5001 is set automatically in order to prevent a collision with any existing web server.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        <PORTNUMBER>
            80
        </PORTNUMBER>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```

In this example, the port number of the web server is set to 80.

## 5.1.9 `<SERVEROPTIONS>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;SERVEROPTIONS&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The &quot;Server Options&quot; tag includes all basic parameters of the web server. The settings made within the tag affect the core of the web server.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```
## 5.1.10 `<SSLPORTNUMBER>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;SSLPORTNUMBER&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For the SSL Protocol (Secure Socket Layer), an additional well-known port number is needed. This is normally port number 443. If SSL is used in the web server, the port number for SSL can be set here. If nothing is set, the number 5443 is set automatically in order to prevent a collision with any existing web server.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        <SSLPORTNUMBER>443</SSLPORTNUMBER>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```

In this example, the port number for SSL is set to 443.

## 5.1.11 `<TIMEZONE>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;TIMEZONE&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sets the time zone of the web server. To enable time zones to be synchronized with other partners (in other words, to enable the local time-of-day setting of the web server to be converted to UTC), the web server must know which time zone has been set for the control's local clock. The value specified here represents the deviation from UTC +/- minutes.</td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE>
        [...]
    </BASE>
    <SERVEROPTIONS>
        <TIMEZONE>60</TIMEZONE>
        [...]
    </SERVEROPTIONS>
    [...]
</SERVERPAGES>
```

In this example, the time zone is set to "UTC + 60 minutes". This corresponds to MET winter time.
### 5.1.12 `<USERDATABASE>`

<table>
<thead>
<tr>
<th>Tag</th>
<th><code>&lt;USERDATABASE&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Access protection can be provided for a wide range of areas of the web server, starting with HTML pages, directories, etc., right up to individual actions of applications.</td>
<td></td>
</tr>
<tr>
<td>The security system is structured as follows:</td>
<td></td>
</tr>
<tr>
<td>- There are users (User)</td>
<td></td>
</tr>
<tr>
<td>- Each User has a password</td>
<td></td>
</tr>
<tr>
<td>- There are secure areas (SecureGroups and/or Realms)</td>
<td></td>
</tr>
<tr>
<td>- Each secure area has a group of users who are authorized to access it</td>
<td></td>
</tr>
<tr>
<td>- A user can have access to different secure areas.</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```xml
<UserDataBase>
    <USER NAME="Gast" PASSWORD="MyPassword">
        <DESCRIPTION>Gast User</DESCRIPTION>
        <GROUP Name="User" />
    </USER>
    <USER NAME="Administrator">
        <DESCRIPTION>Administrator</DESCRIPTION>
        <GROUP NAME="MiniWeb Administratoren" />
        <GROUP NAME="Administrator" />
        <GROUP NAME="User" />
        <GROUP NAME="FileAdministrator" />
        <GROUP NAME="NoAccess" />
    </USER>
</UserDataBase>
```

The following link structure is provided:

```xml
<BASE>
    <Trap1 LINK="/Trap2" SECUREGROUP="Member_Trap1"/>
    <Trap2 SECUREGROUP="Member_Trap2"/>
    <Winner.mcs>
        [ . . . ]
    </Winner.mcs>
</Trap2></BASE>
```

A user who is a member of both the "Member_Trap1" and "Member_Trap2" SecureGroups cannot request the URL, as Trap1 requests the SecureGroup "Member_Trap1" and if it is available, references Trap2 (this is a security violation). However, this now requires "Member_Trap2". While the user belongs to this second group, he has already logged on in the role "Member_Trap1". For this reason, the second Request is rejected.

Direct access to http://Server/Trap2/Winner.mcs is possible, on the other hand, because the user is in the SecureGroup "Member_Trap2" and only it is required.
### 5.1.13 BROWSEABLE attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>LOCALLINK or as a global switch via the tag <code>&lt;BROWSEABLE&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>BROWSEABLE</td>
</tr>
</tbody>
</table>

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]  
  <BASE>
    <!-- LOCALLINK="/UserData" BROWSEABLE="TRUE" REALM="Bediener"/>
    <Test.mcs LINK="/Tests/Test.mcs/"/>
    <Default.mcs>
      <!-- <![CDATA[  
        <HTML>  
          <!-- HEAD>  
            [...]  
          <!-- ]]>
        </HTML>  
      -->
      </Default.mcs>
  </BASE>
  [...]  
</SERVERPAGES>
```

### 5.1.14 LOCALLINK attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>Any node: <code>&lt;BASE&gt;</code> etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>LOCALLINK</td>
</tr>
</tbody>
</table>
5.1.15 MODIFY attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>Any node: BASE, MainDir, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>MODIFY</td>
</tr>
<tr>
<td></td>
<td>If a directory has a MODIFY attribute and the logged-in user is a member of one of the specified groups, the user may carry out all write operations in this directory.</td>
</tr>
<tr>
<td></td>
<td>He may</td>
</tr>
<tr>
<td></td>
<td>• Create new directories</td>
</tr>
<tr>
<td></td>
<td>• Overwrite files</td>
</tr>
<tr>
<td></td>
<td>• Delete files</td>
</tr>
<tr>
<td></td>
<td>• Create new files</td>
</tr>
<tr>
<td></td>
<td>The user must, of course, have READ rights as well (otherwise, he/she would not have access to the directory to start with).</td>
</tr>
</tbody>
</table>

5.1.16 PREFER_EXTERNAL attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>LOCALLINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>PREFER_EXTERNAL</td>
</tr>
<tr>
<td></td>
<td>The PREFER_EXTERNAL attribute can be set only in conjunction with LOCALLINK.</td>
</tr>
<tr>
<td></td>
<td>If PREFER_EXTERNAL is set, the external file system is accessed (read/write) preferentially.</td>
</tr>
<tr>
<td></td>
<td>If a file with the same access path exists simultaneously in the XML and the external file system, the file from the XML file system is taken by default.</td>
</tr>
<tr>
<td></td>
<td>If PREFER_EXTERNAL is set, the file from the external file system is taken. The same applies to directories (with write access).</td>
</tr>
</tbody>
</table>
### 5.1.17 READ attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>Any node: BASE, MainDir, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>READ</td>
</tr>
</tbody>
</table>
| Example      | `<xml version="1.0" standalone="yes"?>
<SERVERPAGES>
  [...]
  <BASE LOCALLINK="/">
  <MainDir REALM="USER" LOCALLINK="/Base/" >
  <www LOCALLINK="/WebSeiten/
    BROWSEABLE="TRUE"
    READ="Administrator"
    WRITE="FileAdministrator" />
  </MainDir>
  <Test.mcs LOCALLINK="/Tests/Test.mcs="/>
  </XMLDir>
  <Default.mcs>
    <![CDATA[
      <HTML>
        <HEAD>
          [...]
        ]]>  
      </Default.mcs>
    </XMLDir>
    <![CDATA[
      <Default.mcs>
        <![CDATA[
          <HTML>
            <HEAD>
              [...]
          ]]>  
        </Default.mcs>
      </XMLDir>
    ]]>  
  </BASE>
  [...]
</SERVERPAGES>` | If a READ attribute is specified for a directory, the user must be a member of one of the groups specified for the READ attribute. With READ, several groups can be specified. These must be separated with commas and no Whitespace characters may be used. |
5.1.18 REALM attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any node:</td>
<td>REALM</td>
<td>BASE, MainDir, etc.</td>
</tr>
</tbody>
</table>

The REALM attribute is used to set up a secure area. REALM may only contain one group name. The REALM attribute enables one login for all users of a group. For all users that do not belong to this group, access is blocked.

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE LOCALLINK="/"/>
    <MainDir REALM="USER" LOCALLINK="/Base/" >
        <www LOCALLINK="/WebSeiten/"
            BROWSEABLE="TRUE"
            READ="Administrator"
            WRITE="FileAdministrator" />
    </MainDir>
    <Test.mcs LOCALLINK="/Tests/Test.mcs="/>
    <XMLDir>
    </XMLDir>
    <Default.mcs>
        <![CDATA[
            <HTML>
                <HEAD>
                    [...]
                </HEAD>
            </HTML>
        ]]>}
    </Default.mcs>
    </BASE>
    [...]
</SERVERPAGES>
```
5.1.19 WRITE attribute

<table>
<thead>
<tr>
<th>Tag</th>
<th>Any node: BASE, MainDir, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>WRITE</td>
</tr>
<tr>
<td></td>
<td>If a directory has a WRITE attribute and the logged-in user is a member of one of the specified groups, the user may only create new files in this directory. He may:</td>
</tr>
<tr>
<td></td>
<td>• Not create any new directories</td>
</tr>
<tr>
<td></td>
<td>• Not overwrite any files</td>
</tr>
<tr>
<td></td>
<td>• Not delete any files</td>
</tr>
<tr>
<td></td>
<td>• Create new files</td>
</tr>
<tr>
<td></td>
<td>The user must, of course, have READ rights as well (otherwise, he/she would not have access to the directory to start with).</td>
</tr>
</tbody>
</table>

Example

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
[...]  
<BASE LOCALLINK="/">
  <MainDir REALM="USER" LOCALLINK="/Base/">
    <www LOCALLINK="/WebSeiten/" 
      BROWSEABLE="TRUE"
      READ="Administrator"
      WRITE="FileAdministrator" />
  </MainDir>
  <Test.mcs LOCALLINK="/Tests/Test.mcs/"/>
</XMLDir>
</SERVERPAGES>
```
### 5.2 WebCfgFrame.xml

#### 5.2.1 <BASE>

<table>
<thead>
<tr>
<th>Tag</th>
<th></th>
</tr>
</thead>
</table>
| The link lists for standard pages (HTML and XML) are stored in the `<BASE>`-tag of `WebCfgFrame.xml`.

**Example**

```xml
<?xml version="1.0" standalone="yes"?>
<SERVERPAGES>
    [...]
    <BASE LOCALLINK="/">
        [...]
        <alarms.mbs LOCALLINK="html/standard/alarms.mbs"
            PREFER_EXTERNAL="TRUE"/>
    [...]
</BASE>
    [...]
</SERVERPAGES>
```
5.2.2  <CONVERSION>

The conversion service offers various options for changing data from one format to another. Conversion is possible in both directions. For example, UTF8 can be converted to ASCII, URLs and HTML pages can be decoded and encoded, and typical data type conversions, e.g. from double to string, can be performed. When converting from UTF-8 to ASCII, a code page is used to enable ASCII characters >128 to be converted into any UTF-8 characters.

Example

```
<SERVERPAGES>
    [...]
    <CONVERSION>
        <CODEPAGE Name="Standard">
            <CHAR UCS="F6" ASCII="F5" />
        </CODEPAGE>
    </CONVERSION>
    [...]
</SERVERPAGES>
```

<CODEPAGE> can have as many <CHAR> entries as you like. However, there is no point in having more than 128 entries, as ASCII only makes 128 entries available in the range 128 to 255. A line is inserted in the code page for each mapped letter:

```
<CHAR UCS="F5" ASCII="A0" />
```

UCS is the UTF-8 character, which can consist of a maximum of 4 bytes. ASCII is the entry in the ASCII table, which is to correspond to this character. A DOS to UTF-8 code page is supplied as a “standard” code page. If the target system uses ANSI, no code page is required (under Windows, for example).
## 5.2.3 <DEFAPP>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The default application has the following tasks:</td>
</tr>
<tr>
<td></td>
<td>• Sending of files</td>
</tr>
<tr>
<td></td>
<td>– HTML pages</td>
</tr>
<tr>
<td></td>
<td>– Images</td>
</tr>
<tr>
<td></td>
<td>– Etc.</td>
</tr>
<tr>
<td></td>
<td>• Receiving a new WebCfg.xml configuration file</td>
</tr>
<tr>
<td></td>
<td>– Parsing of this file</td>
</tr>
<tr>
<td></td>
<td>– Restarting the web server</td>
</tr>
<tr>
<td></td>
<td>• Generating the directory browser</td>
</tr>
<tr>
<td></td>
<td>– Creating the directory contents</td>
</tr>
<tr>
<td></td>
<td>– Deleting of files</td>
</tr>
<tr>
<td></td>
<td>– Creating or deleting of directories</td>
</tr>
<tr>
<td></td>
<td>– Loading of files</td>
</tr>
</tbody>
</table>

For some of these tasks, the default application itself must generate HTML pages and return them to the client, e.g. for the directory browser or the confirmation if a new file has been received. These pages should be designed by the user so that they can be integrated into the desired appearance of the target system.

For this purpose, multiple HTML fragments are defined, that the default application uses to compile the response pages. These HTML fragments are in the configuration area and will be explained in more detail next. As has already been seen in the `<BASE>` section (see the chapter titled Virtual file system (Page 106)), the HTML fragments must be encapsulated in `<![CDATA[ ... ]]>` blocks, since they may not exhibit valid XML syntax.

### Example

```xml
<SERVERPAGES>
  ...
  <CONFIGURATION_DATA>
    <DEFAPP>
      <DIRHEAD>
        ...
      </DIRHEAD>
      <DIRTAIL>
        ...
      </DIRTAIL>
      <DIRLINE>
        ...
      </DIRLINE>
      <LOADSUCCEED>
        ...
      </LOADSUCCEED>
      <SENDCOMPLETE>
        ...
      </SENDCOMPLETE>
    </DEFAPP>
    ...
  </CONFIGURATION_DATA>
  ...
</SERVERPAGES>
```
See also

User-defined directory page (Page 198)
5.2.4  **<HTTP_RESULT_CODES>**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&lt;HTTP_RESULT_CODES&gt;</strong></td>
<td>All error messages that the web server generates can be customized with this tag.</td>
</tr>
</tbody>
</table>

**Example**

```
<SERVERPAGES>
  ...
  <HTTP_RESULT_CODES>
  <RESULT NAME="200" CODE="OK" LOCALLINK="/ErrorCodes/200_OK.mcs">
  <![CDATA[
    <HTML>
    ...
    </HTML>
  ]]>
  ...
  </RESULT>
  <RESULT NAME="404" CODE="NOT FOUND">
  <![CDATA[
    <HTML>
    ...
    </HTML>
  ]]>
  ...
  </RESULT>
  ...
  </HTTP_RESULT_CODES>
  ...
</SERVERPAGES>
```

The structure is based on the error numbers (here, error number 404). The `<RESULT>` tag includes all entries for this error number.

The `<CODE>` tag contains the brief error line specified by HTTP. This is standardized and must always be the same, irrespective of the language. For 404, this text is always "NOT FOUND".

RFC 2068 describes the available error messages and their associated brief texts and meanings.

The `<![CDATA[ ...]]>` block contains an HTML page that is sent to the client in the event of an error and that should contain a description of the error. There are no restrictions regarding the structure of this page. For example, it could contain only a reference to the home page of the web server, which is popular in the case of Error 404 in order not to show the client that it has accessed a non-existent page.

---

**Figure 5-1**  Output of an error page

---

SIMOTION IT Ethernet-based HMI and Diagnostic Functions

Diagnostics Manual, 11/2010
5.2.5  <SOAPAPP>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;SOAPAPP&gt;</td>
<td>Enables SOAP services to be activated and deactivated. A service can be deactivated permanently by deleting the corresponding &lt;WEBSERVICE&gt; tag.</td>
</tr>
</tbody>
</table>
| Example             | <SERVERPAGES>
|                     | [...]
|                     | <SOAPAPP>
|                     |   <STATIC>
|                     |     <WEBSERVICE NAME="OpcXml" URL="/SOAP/OPCXML" />
|                     |     <WEBSERVICE NAME="TVS" URL="/SOAP/TVS"/>
|                     |   </STATIC>
|                     | </SOAPAPP>
|                     | [...]
|                     | </SERVERPAGES>                                                                         |

5.2.6  <SSL>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;SSL&gt;</td>
<td>SSL provides channel encryption for TCP/IP applications. It is thus not restricted to HTTP. However, familiarity with SSL (or its successor TLS) would have been made possible just through its use as channel encryption for HTTP. Keys are required for SSL encryption (just as for every other method of encryption). In SSL, the key material is provided by the server. The &lt;CERTIFICATES&gt; entry in the configuration indicates where this key material can be found.</td>
</tr>
</tbody>
</table>
| Example             | <SERVERPAGES>
|                     | [...]
|                     | <CONFIGURATION_DATA>
|                     |   <SSL>
|                     |     <CERTIFICATES DefaultCert="cert.pem" DefaultKey="key.pem"
|                     |             TrustedCAPath="CA" />
|                     |   </SSL>
|                     | [...]
|                     | </CONFIGURATION_DATA>
|                     | [...]
|                     | </SERVERPAGES>                                                                         |

The key material comprises two parts:
1. The certificate, which contains additional information about the server, such as the server name. This contains the public key of the server. Only the certificate is transferred to the client.
2. The private key, which the server uses to encrypt data or to decrypt data that was encrypted with the public key.
Data encrypted with the private key can be decrypted using the public key.
The certificate is stored in the file that is stored under the DefaultCert attribute. The private key is stored in the file that is stored under the DefaultKey attribute.
For more information about SSL, refer also to secondary references (http://www.openssl.org/).
The SSL module can evaluate and verify client certificates. The TrustedCAPath attribute indicates the locations of the CA certificates in the system.
For further information, refer to the separate SSL documentation.
5.3 MWSL functions

5.3.1 AddHTTPHeader

<table>
<thead>
<tr>
<th>Syntax</th>
<th>AddHTTPHeader(&lt;Http-Header&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This command can be used to add HTTP headers from MWSL. These are then not transferred as part of the document but rather in the protocol portion of HTTP.</td>
</tr>
</tbody>
</table>

| Parameters | <Http-Header> | Character string that ends with \r\n. If multiple HTTP headers are to be entered, the individual headers must be separated by \r\n. |

| MWSL example | <MWSL> <!-- 
var strCookie;
strCookie = "Set-cookie: siemens_automation_language=";
strCookie = strCookie + GetVar("Language","URL");
strCookie = strCookie + ", path=/\r\n";
AddHTTPHeader( strCookie );
--> </MWSL> |
5.3.2 CacheVar

Syntax

CacheVar(<Variable Names>)

This function stores process variables in a cache memory. In the case of repeated access to the variables, they do not have to be retrieved again from the variable source. In addition, this allows a process image to be generated at a certain time. If CacheVar() is called again, the value of the variable is updated. The cache memory is only valid for the respective MWSL page.

Parameters

| <Variable Name> | String with the variable name. The pipe character (" | ") is the separator |

Example

<!-- MWSL -->

CacheVar("MiniWeb_Build|WWWRoot");
// The MiniWeb_Build and WWWRoot process variables are added to the cache.

WriteVar("MiniWeb_Build");
WriteVar("WWWRoot");

CacheVar("SystemTime");
// SystemTime is added to the cache.

WriteVar("SystemTime");

CacheVar("SystemTime");
// The value of SystemTime is updated.

WriteVar("SystemTime");

--></MWSL>

If a value is written while the MWSL page is being processed, it is updated in the cache and in the process.
## 5.3.3 ExistVariable

**Syntax**

```
ExistVariable(<Variable Name>, <Variable Source>)
```

This command queries the presence of a variable. It returns TRUE or FALSE.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Variable Name&gt;</td>
<td>Variable name</td>
</tr>
<tr>
<td>&lt;Variable Source&gt;</td>
<td>Name of variable source. Possible sources: &quot;URL&quot;, &quot;PROCESS&quot;, &quot;HTTP&quot;, &quot;COOKIE&quot;. If this parameter is omitted, <code>ExistVariable</code> assumes the source &quot;PROCESS&quot; by default.</td>
</tr>
</tbody>
</table>

### Example

```
ExistVariable("Parameter", "URL") If the URL variable "Parameter" exists, TRUE is returned; if the variable does not exist, FALSE is returned.

ExistVariable("Color", "PROCESS") If the process variable "Color" exists, TRUE is returned; if the process variable does not exist, FALSE is returned.

ExistVariable("Color") If the process variable "Color" exists, TRUE is returned; if the process variable does not exist, FALSE is returned.
```

`ExistVariable` is usually queried in an if condition to allow an appropriate response if a variable is not present (see If (Page 174)).

```mws
<--
var fakultaet = 1;
var fakul = 0;
var zaehler = 0;
if(ExistVariable("Fakul", "URL"))
{
    fakul = GetVar("Fakul", "URL");
}
for(zaehler = 1; zaehler <= fakul; zaehler++)
{
    fakultaet = fakultaet * zaehler;
}
write ("F( " + fakul + ") = " + fakultaet );
-->
```

If the URL variable Fakul exists, the local variable fakul is assigned the value of the URL variable.

**Notice!**

The variable source URL always returns an empty string for non-existent URL parameters. Therefore, variables from the variable source URL cannot be queried with `ExistVariable()`. `ExistVariable()` always returns TRUE in this case.

However, it is possible to query the empty string:

```mws
If (GetVar("MyVar", "URL") == ")
{
    write("Url Parameter MyVar not set.");
}
```

5.3 MWSL functions

5.3.4 GetVar

**Syntax**

```
GetVar(<Variable Name>, <Variable Source>, <Format String>);
```

This function returns the value of a variable from a variable source.

If a parameter does not exist, "null" is returned.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Variable Name&gt;</td>
<td>Variable name</td>
</tr>
<tr>
<td>&lt;Variable Source&gt;</td>
<td>Name of variable source. The indicated variable sources are those supplied from the web server. Additional variable sources can be developed and must be handled analogously. If this parameter is omitted, GetVar() assumes the type PROCESS by default.</td>
</tr>
<tr>
<td>&lt;Format String&gt;</td>
<td>The handling of the format string depends on the variable source. Thus, this property is not possible for the variable sources COOKIE and URL. Syntax of an HTTP variable: Variables and HTTP header information (Page 170) Syntax of a process variable: Global variables (Page 164)</td>
</tr>
</tbody>
</table>

**Example**

```
GetVar("Color");  Returns the content of the variable Color.
Because PROCESS is the default variable source, the result corresponds to that of the following call.
GetVar("Color", "PROCESS");  Returns the content of the Color variable, the variable source PROCESS.
GetVar("Parameter", "URL");  Returns the content of the Parameter variable from the URL.
GetVar("Accept-Language", "HTTP", "?-" ) Returns the content of the HTTP variable Accept-Language. The format string "?-" indicates that all characters up to the first occurrence of the ".-" character will be returned.
GetVar("Color", "PROCESS", "[2,3]" );  Returns 3 characters, starting from position 2 of the process variable Color. The result is characters 2-5 of the process variable.
GetVar("Accept-Language", "HTTP", "[3,0]" ) Returns the content of the HTTP variable Accept-Language starting from the 3rd character.
```
### 5.3.5 InsertFile

#### Syntax

InsertFile(<File name>)

- This command allows an existing text file to be imported individually.
- This text file is interpreted with MWSL prior to insertion.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;File Name&gt;</td>
<td>Name of text file, including path.</td>
</tr>
</tbody>
</table>

#### Example

```html
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
  [...]
  [...]
  <table>
  [...]
  <tr>
    <td>
      An HTML file is now displayed on the right-hand page.
    </td>
    <td>
    </td>
    <td>
      <MWSL>InsertFile("/MWSL/Output.mcs")</MWSL>
    </td>
  </tr>
  [...]
  <table>
  [...]
  </BODY>
</HTML>
```

The HTML file `Output.mcs` is now inserted and displayed in the right-hand column of the table.
## 5.3.6 ProcessXMLData

### Syntax

```
ProcessXMLData( <DATA>, <TEMPLATE> )
```

This command can be used to generate dynamic HTML files based on data and template files. There is a file containing data (Data file) and another file in which the structure is defined (template). ProcessXMLData() combines the two files into one HTML file. In this process, the template file is run through for each data element in order to determine how it should be displayed. This produces a separation of the data from the content. By changing the template file subsequently, you can change the appearance of one or more pages without having to change the data pages. As a result, data can be added more easily, and a different structure can be integrated much more easily by simply inserting a different template file. Additional information for the template mechanism: Mode of operation of the template mechanism (Page 177)

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;DATA&gt;</td>
<td>Data for the dynamic HTML file</td>
</tr>
<tr>
<td></td>
<td>A file or a variable containing the data can be transferred as a parameter.</td>
</tr>
<tr>
<td></td>
<td>File: &quot;&quot;&lt;EXTERNAL SRC=&quot;/datafile.xml&quot;/&quot;&quot;, in which datafile.xml is the file containing the data.</td>
</tr>
<tr>
<td></td>
<td>Variable:</td>
</tr>
<tr>
<td></td>
<td>Specifies the variable name.</td>
</tr>
<tr>
<td>&lt;TEMPLATE&gt;</td>
<td>Template (how the data are displayed)</td>
</tr>
<tr>
<td></td>
<td>A file or a variable containing the templates can be transferred as a parameter.</td>
</tr>
<tr>
<td></td>
<td>File: &quot;&quot;&lt;TEMPLATES&gt;&lt;EXTERNAL SRC=&quot;/Template.xml&quot;/&quot;&gt;&quot;&quot; in which &quot;Template.xml&quot; is the file containing the templates.</td>
</tr>
<tr>
<td></td>
<td>Variable:</td>
</tr>
<tr>
<td></td>
<td>Specifies the variable name.</td>
</tr>
</tbody>
</table>
### 5.3 MWSL functions

#### Example

```
Example

ProcessXMLData("<EXTERNAL SRC="/MWSL/variables.xml "/>",
"<TEMPLATES><EXTERNAL
SRC="/MWSL/variablesTemplate.xml"/></TEMPLATES>"));
```

#### MWSL example

```
<MWSL>!---<EXTERNAL SRC="/MWSL/variablesTemplate.xml"/>
var Head = "<Provider Name="MyVarProvider">";
var Data = "<Variable Name="ZUFUEHRUNG" Type="String"
 InitialValue="good" Behavior="Manual"
 Description="Part infeed."/>";
var Foot = "</Provider>");

var XMLData = Head + Data + Foot;

var TemplateHead = "<TEMPLATES>";
var TemplateFoot = "</TEMPLATES>");
var TemplateFile = "<EXTERNAL SRC="/" + GetVar("Template", "URL")
 + "/"/>");
ProcessXMLData(XMLData, TemplateHead + TemplateFile +
 TemplateFoot);
--></MWSL>
```

The XMLData variable contain the data nodes with the associated attributes.

Note that the " must be protected from the XML Parser with \\

The same method can be used to define a variable for the templates.

In the example above, the variable for the templates consists of a file specification.

### 5.3.7 SetVar

#### Syntax

```
SetVar(<Variable Name>, <Value>)
```

This function is used to place process variables.

The variables to be written depend on the variable sources.

For security reasons, only variables starting with the prefix "USER_" or "SESSION_" can be set, e.g. "SESSION_Parameter".

#### Parameters

- `<Variable Name>`
  - Variable name
  - Note that the variable name must start with USER_ or SESSION_, e.g. "SESSION_Parameter".

- `<Value>`
  - The new variable value.

#### Example

```
SetVar("Color", "Green");
```

Places the process variable Color at the value Green.
### 5.3 MWSL functions

#### 5.3.8 ShareRealm

**Syntax**

```
ShareRealm(<Group>)
```

Indicates whether the current user is a member of the group transferred as a parameter. The return value can be **TRUE** or **FALSE**.

**Parameters**

- **<Group>**
  - The following groups are currently allowed as parameters:
    - **NO_REALM** No group association
    - **ANY_REALM** Any group association
    - **[Group Name]** Member of group **[Group Name]** Groups depend on configuration.

**Example**

```
write(ShareRealm("ANY_REALM"));
```

If the current user is in any defined group, 1 is output. Otherwise, 0 is output.

**MWSL example**

```
<MWSL>!
if( ShareRealm("ANY_REALM") )
{
  write ("<tr valign="baseline">\r\n");
  write ("<td><H2>Hello " + GetVar("Username", "HTTP").replace(/\n\n/\r\n\r\n/); you're successfully logged in.</H2></td><tr><\r\n");
}
-->
</MWSL>
```

If the user is a member of any group, the instructions in the curly brackets are carried out.
### 5.3.9 write

**Syntax**

```
write(<Text>)
```

The `write()` function writes text to the output of an HTML page.

**Parameters**

- `<Text>`: Text, return values of functions, or variable contents can be transferred.

**Example**

```
<MWSL><!-
  write("Hello World");
  // Output: Hello World

  write("Hello" + " " + "World");
  // Output: Hello World

  write(GetVar("Parameter", "URL");
  // Output of the content of variable Parameter in the URL.

  write(5+6);
  // Output: 11

  var zahl1 = 5;
  var zahl2 = 7;
  var string = "Hello";
  write(string + ": " + zahl1 + zahl2);
  // Output: Hello: 57

  write(zahl1 + zahl2);
  // Output: 12

  write("Content of Parameter: " + GetVar("Parameter", "URL");
  // Ausgabe: Content of Parameter: Hello
  // if Parameter contains the string "Hello".
--></MWSL>
```
5.3.10 WriteVar

Syntax

WriteVar(<Variable Name>, <Variable Source>, <Format String>);

This command outputs the content of a variable; it will be written to the output.

WriteVar is almost the same function as GetVar.
WriteVar is equivalent to the call:

WriteVar(...) === write( GetVar ( ... ) )

The sole difference is that WriteVar outputs the content of the specified variable, while GetVar() returns the content as a return value.

Parameters

<table>
<thead>
<tr>
<th>&lt;Variable Name&gt;</th>
<th>Variable name</th>
</tr>
</thead>
</table>
| <Variable Source> | Name of variable source
|                  | The indicated variable sources are those supplied from the web server.
|                  | Additional variable sources can be developed and must be handled analogously.
|                  | If this parameter is omitted, GetVar() assumes the type PROCESS by default. |
| <Format String> | The handling of the format string depends on the variable source. |
|                 | Thus, this property is not possible for the variable sources COOKIE and URL. |
|                 | The call syntax is equivalent to that of GetVar() (Page 290). |
Example

```mwsl
<MWSL>
<!-- write(GetVar("Parameter", "URL"));
   // The content of the variable Parameter is output here.
   // GetVar returns the value of the variable, which is then written to the output with write.
   // (See also GetVar() and write().)
   // The same output can also be achieved with the following command.
   WriteVar("Parameter", "URL");
   // WriteVar writes directly to the output and supplies no return value.

   WriteVar("Color");
   // Outputs the content of Color, which is a process variable.

   WriteVar("Accept-Language", "HTTP", "?=" )
   // Outputs the content of the HTTP variable "Accept-Language" up to the "=" character.

   WriteVar("Color", "PROCESS", "[2,3]");
   // Outputs the characters 2 - 5 of the process variable Color.

   WriteVar("Accept-Language", "HTTP", "[3,0]" )
   // Outputs the content of the HTTP variable "Accept-Language" starting from the third character.

   --> <MWSL>
```

See also

Global variables (Page 164)
## 5.3 MWSL functions

### 5.3.11 WriteXMLData

WriteXMLData() outputs the data in contrast to ProcessXMLData. Instead of write{ ProcessXMLData(...)}; you can also write WriteXMLData(...);. WriteXMLData(); is assigned the same parameters as ProcessXMLData();.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>WriteXMLData ( &lt;DATA&gt;, &lt;TEMPLATE&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameters</strong></td>
<td>Data for the dynamic HTML file</td>
</tr>
<tr>
<td>&lt;DATA&gt;</td>
<td>A file or a variable containing the data can be transferred as a parameter. File: &quot;&lt;EXTERNAL SRC=&quot;/Datendatei.xml &quot;&gt;&quot;, in which &quot;data file.xml&quot; is the file containing the data. Variable: &lt;Variable Name&gt; Specifies the variable name.</td>
</tr>
<tr>
<td>&lt;TEMPLATE&gt;</td>
<td>Template (how the data are displayed)</td>
</tr>
<tr>
<td></td>
<td>A file or a variable containing the templates can be transferred as a parameter. File: &quot;&lt;TEMPLATES&gt;&lt;EXTERNAL SRC=&quot;/Template.xml &quot;&gt;&quot;, in which &quot;Template.xml&quot; is the file containing the templates. Variable: &lt;Variable Name&gt;</td>
</tr>
</tbody>
</table>
| **Example** | WriteXMLData("&lt;EXTERNAL SRC="/MWSL/variables.xml ">", 
| | "&lt;TEMPLATES&gt;&lt;EXTERNAL SRC="/MWSL/variablesTemplate.xml ">"); |
| | Additional information about the template mechanism: Mode of operation of the template mechanism (Page 177) |
### 5.3.12 NodeIndex

<table>
<thead>
<tr>
<th>Variable</th>
<th>NodeIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NodeIndex</strong> is a process variable that is available when a template is parsed.</td>
</tr>
<tr>
<td></td>
<td>This variable outputs the number of nodes that have already been run through.</td>
</tr>
<tr>
<td></td>
<td>The access is exactly the same as for other variables of the <strong>PROCESS</strong> variable source.</td>
</tr>
<tr>
<td></td>
<td>Additional information about the template mechanism: Mode of operation of the template mechanism (Page 177)</td>
</tr>
</tbody>
</table>

#### Example

```xml
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Variable">
    <POSITION NAME="LINE">
      <![CDATA[
        <MWSL> WriteVar("NodeIndex ") </MWSL>
      ]]>  
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
```

### 5.3.13 NodeLevel

<table>
<thead>
<tr>
<th>Variable</th>
<th>NodeLevel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>NodeLevel</strong> is a process variable that is available when a template is parsed.</td>
</tr>
<tr>
<td></td>
<td>This variable outputs the hierarchy level of the current node.</td>
</tr>
<tr>
<td></td>
<td>The access is exactly the same as for other variables of the <strong>PROCESS</strong> variable source.</td>
</tr>
<tr>
<td></td>
<td>Additional information about the template mechanism: Mode of operation of the template mechanism (Page 177)</td>
</tr>
</tbody>
</table>

#### Example

```xml
<?xml version="1.0" ?>
<TEMPLATES>
  <TEMPLATE NAME="Variable">
    <POSITION NAME="LINE">
      <![CDATA[
        <MWSL> WriteVar("NodeLevel") </MWSL>
      ]]>  
    </POSITION>
  </TEMPLATE>
</TEMPLATES>
```
5.4 IT DIAG files

5.4.1 DIAGURLS.TXT

Structure of the DIAGURLS.TXT file

The file that is located in the /HMI/SYSLOG/DIAG directory contains the name of the IT DIAG pages to be backed up.

Below is an example of how this file might look:

alarms.mcs
alarmsdrv.mcs
alarmbuf.mcs
devinfo.mcs
diagbuff.mcs
diagbuffdrv.mcs
diagnost.mcs
ipconfig.mcs
mempool.mcs
start.mcs
taskrunt.mcs
timezone.mcs

Content of the DIAGURLS.TXT file

See also

Diagnostic files (Page 49)
5.5 LCID country codes

5.5.1 LCID table

Country-specific codes

<table>
<thead>
<tr>
<th>Decimal value</th>
<th>Country</th>
<th>UMC code</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1033</td>
<td>English - United States</td>
<td>B 1</td>
<td>1</td>
</tr>
<tr>
<td>2057</td>
<td>English - Great Britain</td>
<td>B 2</td>
<td>2</td>
</tr>
<tr>
<td>1031</td>
<td>German - Germany</td>
<td>A 3</td>
<td>3</td>
</tr>
<tr>
<td>1036</td>
<td>French - France</td>
<td>C 4</td>
<td>4</td>
</tr>
<tr>
<td>1034</td>
<td>Spanish - Spain (Trad.)</td>
<td>D 5</td>
<td>5</td>
</tr>
<tr>
<td>1040</td>
<td>Italian - Italy</td>
<td>E 6</td>
<td>6</td>
</tr>
<tr>
<td>3081</td>
<td>English - Australia</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>10249</td>
<td>English - Belize</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>4105</td>
<td>English - Canada</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>9225</td>
<td>English - Caribbean</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>6153</td>
<td>English - Ireland</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>8201</td>
<td>English - Jamaica</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>5129</td>
<td>English - New Zealand</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>13321</td>
<td>English - Philippines</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>7177</td>
<td>English - Southern Africa</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>11273</td>
<td>English - Trinidad</td>
<td>B 10</td>
<td>10</td>
</tr>
<tr>
<td>3079</td>
<td>German - Austria</td>
<td>A 20</td>
<td>20</td>
</tr>
<tr>
<td>5127</td>
<td>German - Liechtenstein</td>
<td>A 20</td>
<td>20</td>
</tr>
<tr>
<td>4103</td>
<td>German - Luxembourg</td>
<td>A 20</td>
<td>20</td>
</tr>
<tr>
<td>2055</td>
<td>German - Switzerland</td>
<td>A 20</td>
<td>20</td>
</tr>
<tr>
<td>2060</td>
<td>French - Belgium</td>
<td>C 30</td>
<td>30</td>
</tr>
<tr>
<td>3084</td>
<td>French - Canada</td>
<td>C 30</td>
<td>30</td>
</tr>
<tr>
<td>5132</td>
<td>French - Luxembourg</td>
<td>C 30</td>
<td>30</td>
</tr>
<tr>
<td>4108</td>
<td>French - Switzerland</td>
<td>C 30</td>
<td>30</td>
</tr>
<tr>
<td>11274</td>
<td>Spanish - Argentina</td>
<td>D 40</td>
<td>40</td>
</tr>
<tr>
<td>16394</td>
<td>Spanish - Bolivia</td>
<td>D 40</td>
<td>40</td>
</tr>
</tbody>
</table>
Appendix

5.5 LCID country codes

13322 Spanish - Chile D 40
9226 Spanish - Colombia D 40
5130 Spanish - Costa Rica D 40
7178 Spanish - Dominican Rep. D 40
12298 Spanish - Ecuador D 40
17418 Spanish - El Salvador D 40
4106 Spanish - Guatemala D 40
18442 Spanish - Honduras D 40
2058 Spanish - Mexico D 40
19466 Spanish - Nicaragua D 40
6154 Spanish - Panama D 40
15370 Spanish - Paraguay D 40
10250 Spanish - Peru D 40
20490 Spanish - Puerto Rico D 40
14346 Spanish - Uruguay D 40
8202 Spanish - Venezuela D 40
2064 Italian - Switzerland E 50
1078 Afrikaans
1052 Albanian
14337 Arabic - United Arab Emirates
15361 Arabic - Bahrain
5121 Arabic - Algeria
3073 Arabic - Egypt
2049 Arabic - Iraq
11265 Arabic - Jordan
13313 Arabic - Kuwait
12289 Arabic - Lebanon
4097 Arabic - Libya
6145 Arabic - Morocco
8193 Arabic - Oman
16385 Arabic - Qatar
1025 Arabic - Saudi Arabia
10241 Arabic - Syria
7169 Arabic - Tunisia
9217 Arabic - Yemen
Appendix

5.5 LCID country codes

1067 Armenian
1068 Azeri - Latin
2092 Azeri - Cyrillic
1069 Basque
1059 Belarusian
1026 Bulgarian
1027 Catalan
2052 Chinese - China
3076 Chinese - Hong Kong SAR
5124 Chinese - Macau SAR
4100 Chinese - Singapore
1028 Chinese - Taiwan
1050 Croatian
1029 Czech
1030 Danish
1043 Dutch - Netherlands
2067 Dutch - Belgium
1061 Estonian
1065 Farsi
1035 Finnish
1080 Faroese
2108 Gaelic - Ireland
1084 Gaelic - Scotland
1032 Greek
1037 Hebrew
1081 Hindi
1038 Hungarian
1039 Icelandic
1057 Indonesian
1041 Japanese
1042 Korean
1062 Latvian
1063 Lithuanian
1071 F.Y.R.O. Macedonia
1086 Malay - Malaysia
Appendix

5.5 LCID country codes

2110 Malay - Brunei
1082 Maltese
1102 Marathi
1044 Norwegian - Bokml
2068 Norwegian - Nynorsk
1045 Polish
2070 Portuguese - Portugal
1046 Portuguese - Brazil
1047 Raeto-Romance
1048 Romanian - Romania
2072 Romanian - Republic of Moldova
1049 Russian
2073 Russian - Republic of Moldova
1103 Sanskrit
3098 Serbian - Cyrillic
2074 Serbian - Latin
1074 Setsuana
1060 Slovenian
1051 Slovak
1070 Sorbian
1072 Southern Sotho
1089 Swahili
1053 Swedish - Sweden
2077 Swedish - Finland
1097 Tamil
1092 Tatar
1054 Thai
1055 Turkish
1073 Tsonga
1058 Ukrainian
1056 Urdu
2115 Uzbek - Cyrillic
1091 Uzbek - Latin
1066 Vietnamese
1076 Xhosa
5.5 LCID country codes

1085 Yiddish
1077 Zulu
Index

A
Abbreviations, 265
Access rights to the file system, 110
Alarm buffer, 59

B
BASIC HTML pages, 91
Browse, 215
BROWSEABLE, 109

C
CancelTrace, 228
Client application, 212
Communication package, 204
ComplInfo group, 245
Configuration
   Back up device data, 69
   Loading, 68
   Old configuration data, 70
   Restore device data, 69
   Saving, 68
   Update, 69

D
DEFAPP, 198
Default application, 198
Device Trace, 39
Diagnostic files, 49
Drive diagnostics buffer, 54
Drive faults, 58

E
Encrypted data transfer, 260
Event
   Language output, 52

F
Firmware
   Updating the device, 68
Form of delivery, 13

G
GetProperties, 215
GetStatus, 215, 228

H
Home page, 24
HTML pages
   Alarm buffer, 59
   Alarms, 55
   Device Info, 26
   Diag Buffer, 52
   Diag buffer drive, 54
   Diagnostic files, 49
   Diagnostics, 29
   Drive alarms, 58
   Files, 87
   Home, 24
   Home page, 24
   IP Config, 28
   Manage Config, 68
   Service overview, 31
   Settings, 84
   Syslog, 60
   System Trace, 42
   Task runtime, 30
   Tasktrace, 46
   Trace, 38
   Watch, 34
   HTTPS, 260

I
Installation of IT DIAG
   Ethernet interface, 17
   Hardware/software requirements, 17
   IP address, 17
   ItemName
Index

ActToRam, 240
Drive parameters, 237
Operating mode, 239
RamToRom, 239
System variables, 236, 241, 243, 259
Technological alarms, 238
TO configuration data, 237
TO system variables, 236
Unit variables, 236
ItemPath, 236

O
OPC
XML DA, 204
OPC XML server interface, 215
Operating mode, 239
RUN/STOP/STOP, 85

P
Possible uses
Standard pages, 14
User-defined page, 15
PREFER_EXTERNAL, 108

R
Read, 128
READ, 113
REALM, 112
References, 3
Runtime license, 13

S
Secure Socket Layer, 260
Server Side Includes, 203
Settings
System time/time zone, 85
Simplified HTML pages
Alarms, 97
Device Info, 92
Diag buffer, 95, 96
Diagnostic files, 99
Diagnostics, 94
Home page, 91
IP Config, 98
SOAP, 11, 204
SOAPAPP, 285
StartTrace, 227
StopTrace, 227
Subscribe, 215
SubscriptionCancel, 216
SubscriptionPolledRefresh, 216
System time/time zone, 85
System Trace, 42

T
TaskRT group, 248
Tasktrace, 46

L
LOCALLINK, 108

M
MODIFY, 114
MWSL
COOKIE variables, 170
COOKIES, 169
Error messages, 160
For, 173
Function overview, 176
Global variables, 164
HTTP header, 170
If, 174
Mode of operation, 158
Operators, 172
Processing variable values, 167
Script variables, 162
Structure, 159
Template mechanism, 177
URL parameters, 168
MWSL examples
SetVar(), 187
TestMenu, 195
TestTemplate, 188
Trace
  System Trace, 42
Trace settings, 48
TraceDataCycleEnum, 224
TraceStateEnum, 224

U
Update
  Firmware, 68
  WebCfg.xml firmware, 70
User-defined pages, 118
  Default application, 198
  Directory page, 198
  Home page, 121
  JavaScript, 127
  MWSL, 158
  OPC XML server, 128

V
Variable providers
  SIMOTION, 235
  SIMOTION diagnostics, 244
Variables groups, 244
Virtual file system
  Configuration, 106
  Different virtual and physical file system, 107
  PREFER_EXTERNAL, 108

W
Warnings, 58
Watch table, 34
WebCfg.xml
  BROWSEABLE, 109
  DEFAPP, 198
  Display of directories, 115
  Inheriting authorizations, 115
  MODIFY, 114
  READ, 113
  REALM, 112
  Security concept, 110
  Server options (overview), 104
  Settings, 103
  WRITE, 114
  XML file system, 106
WebTraceViewer, 40
Write, 216
WRITE, 114