BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Lib: Tc3_JsonXml

2022-01-06 | Version: 1.11
Table of contents

1 Foreword .......................................................................................................................... 11
  1.1 Notes on the documentation ...................................................................................... 11
  1.2 Safety instructions ........................................................................................................ 12
  1.3 Notes on information security ...................................................................................... 13

2 Overview .......................................................................................................................... 14

3 Getting started .................................................................................................................. 16

4 Function blocks ................................................................................................................ 19
  4.1 FB_JsonDomParser ...................................................................................................... 19
    4.1.1 AddArrayMember .................................................................................................. 19
    4.1.2 AddBase64Member .............................................................................................. 20
    4.1.3 AddBoolMember .................................................................................................. 20
    4.1.4 AddDateTimeMember .......................................................................................... 20
    4.1.5 AddDcTimeMember ............................................................................................. 20
    4.1.6 AddDoubleMember ............................................................................................. 21
    4.1.7 AddFileTimeMember ........................................................................................... 21
    4.1.8 AddHexBinaryMember ....................................................................................... 21
    4.1.9 AddInt64Member ................................................................................................. 22
    4.1.10 AddIntMember ................................................................................................... 22
    4.1.11 AddJsonMember ................................................................................................ 22
    4.1.12 AddNullMember ................................................................................................. 22
    4.1.13 AddObjectMember ............................................................................................. 23
    4.1.14 AddStringMember ............................................................................................. 23
    4.1.15 AddUint64Member ............................................................................................. 23
    4.1.16 AddUintMember ................................................................................................ 24
    4.1.17 ArrayBegin ......................................................................................................... 24
    4.1.18 ArrayEnd ............................................................................................................ 24
    4.1.19 ClearArray .......................................................................................................... 25
    4.1.20 CopyDocument .................................................................................................. 25
    4.1.21 CopyJson ........................................................................................................... 25
    4.1.22 CopyString ........................................................................................................ 26
    4.1.23 FindMember ....................................................................................................... 27
    4.1.24 FindMemberPath ............................................................................................... 27
    4.1.25 GetArraySize .................................................................................................... 27
    4.1.26 GetArrayValue .................................................................................................. 27
    4.1.27 GetArrayValueByIdx ......................................................................................... 28
    4.1.28 GetBase64 .......................................................................................................... 28
    4.1.29 GetBool ............................................................................................................... 28
    4.1.30 GetDateTime ...................................................................................................... 28
    4.1.31 GetDcTime .......................................................................................................... 29
    4.1.32 GetDocument ..................................................................................................... 29
    4.1.33 GetDocumentLength ......................................................................................... 29
    4.1.34 GetDocumentRoot ............................................................................................. 29
    4.1.35 GetDouble ......................................................................................................... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.36</td>
<td>GetFileTime</td>
<td>30</td>
</tr>
<tr>
<td>4.1.37</td>
<td>GetHexBinary</td>
<td>30</td>
</tr>
<tr>
<td>4.1.38</td>
<td>GetInt</td>
<td>30</td>
</tr>
<tr>
<td>4.1.39</td>
<td>GetInt64</td>
<td>30</td>
</tr>
<tr>
<td>4.1.40</td>
<td>GetJson</td>
<td>30</td>
</tr>
<tr>
<td>4.1.41</td>
<td>GetJsonLength</td>
<td>30</td>
</tr>
<tr>
<td>4.1.42</td>
<td>GetMaxDecimalPlaces</td>
<td>31</td>
</tr>
<tr>
<td>4.1.43</td>
<td>GetMemberName</td>
<td>31</td>
</tr>
<tr>
<td>4.1.44</td>
<td>GetMemberValue</td>
<td>32</td>
</tr>
<tr>
<td>4.1.45</td>
<td>GetString</td>
<td>32</td>
</tr>
<tr>
<td>4.1.46</td>
<td>GetStringLength</td>
<td>32</td>
</tr>
<tr>
<td>4.1.47</td>
<td>GetType</td>
<td>32</td>
</tr>
<tr>
<td>4.1.48</td>
<td>GetUint</td>
<td>33</td>
</tr>
<tr>
<td>4.1.49</td>
<td>GetUint64</td>
<td>33</td>
</tr>
<tr>
<td>4.1.50</td>
<td>HasMember</td>
<td>33</td>
</tr>
<tr>
<td>4.1.51</td>
<td>IsArray</td>
<td>33</td>
</tr>
<tr>
<td>4.1.52</td>
<td>IsBase64</td>
<td>34</td>
</tr>
<tr>
<td>4.1.53</td>
<td>IsBool</td>
<td>34</td>
</tr>
<tr>
<td>4.1.54</td>
<td>IsDouble</td>
<td>34</td>
</tr>
<tr>
<td>4.1.55</td>
<td>IsFalse</td>
<td>34</td>
</tr>
<tr>
<td>4.1.56</td>
<td>IsHexBinary</td>
<td>34</td>
</tr>
<tr>
<td>4.1.57</td>
<td>IsInt</td>
<td>35</td>
</tr>
<tr>
<td>4.1.58</td>
<td>IsInt64</td>
<td>35</td>
</tr>
<tr>
<td>4.1.59</td>
<td>IsISO8601TimeFormat</td>
<td>35</td>
</tr>
<tr>
<td>4.1.60</td>
<td>IsNull</td>
<td>35</td>
</tr>
<tr>
<td>4.1.61</td>
<td>IsNumber</td>
<td>35</td>
</tr>
<tr>
<td>4.1.62</td>
<td>IsObject</td>
<td>35</td>
</tr>
<tr>
<td>4.1.63</td>
<td>IsString</td>
<td>36</td>
</tr>
<tr>
<td>4.1.64</td>
<td>IsTrue</td>
<td>36</td>
</tr>
<tr>
<td>4.1.65</td>
<td>IsUint</td>
<td>36</td>
</tr>
<tr>
<td>4.1.66</td>
<td>IsUint64</td>
<td>36</td>
</tr>
<tr>
<td>4.1.67</td>
<td>LoadDocumentFromFile</td>
<td>37</td>
</tr>
<tr>
<td>4.1.68</td>
<td>MemberBegin</td>
<td>37</td>
</tr>
<tr>
<td>4.1.69</td>
<td>MemberEnd</td>
<td>37</td>
</tr>
<tr>
<td>4.1.70</td>
<td>NewDocument</td>
<td>38</td>
</tr>
<tr>
<td>4.1.71</td>
<td>NextArray</td>
<td>38</td>
</tr>
<tr>
<td>4.1.72</td>
<td>ParseDocument</td>
<td>38</td>
</tr>
<tr>
<td>4.1.73</td>
<td>PushbackBase64Value</td>
<td>38</td>
</tr>
<tr>
<td>4.1.74</td>
<td>PushbackBoolValue</td>
<td>38</td>
</tr>
<tr>
<td>4.1.75</td>
<td>PushbackDateTimeValue</td>
<td>39</td>
</tr>
<tr>
<td>4.1.76</td>
<td>PushbackDcTimeValue</td>
<td>39</td>
</tr>
<tr>
<td>4.1.77</td>
<td>PushbackDoubleValue</td>
<td>39</td>
</tr>
<tr>
<td>4.1.78</td>
<td>PushbackFileTimeValue</td>
<td>40</td>
</tr>
<tr>
<td>4.1.79</td>
<td>PushbackHexBinaryValue</td>
<td>40</td>
</tr>
<tr>
<td>4.1.80</td>
<td>PushbackInt64Value</td>
<td>40</td>
</tr>
<tr>
<td>4.1.81</td>
<td>PushbackIntValue</td>
<td>40</td>
</tr>
</tbody>
</table>
Table of contents

4.1.82 PushbackJsonValue .......................................................... 41
4.1.83 PushbackNullValue .......................................................... 41
4.1.84 PushbackStringValue ...................................................... 41
4.1.85 PushbackUint64Value ...................................................... 41
4.1.86 PushbackUintValue .......................................................... 42
4.1.87 RemoveAllMembers .......................................................... 42
4.1.88 RemoveArray ................................................................. 42
4.1.89 RemoveMember ............................................................... 43
4.1.90 RemoveMemberByName ..................................................... 43
4.1.91 SaveDocumentToFile ....................................................... 43
4.1.92 SetArray .............................................................. 44
4.1.93 SetBase64 .............................................................. 44
4.1.94 SetBool ................................................................. 44
4.1.95 SetDateTime ............................................................... 45
4.1.96 SetDcTime ................................................................. 45
4.1.97 SetDouble ................................................................. 45
4.1.98 SetFileTime ................................................................. 46
4.1.99 SetHexBinary ............................................................... 46
4.1.100 SetInt ................................................................. 46
4.1.101 SetInt64 ............................................................... 46
4.1.102 SetJson ................................................................. 47
4.1.103 SetMaxDecimalPlaces .................................................... 47
4.1.104 SetNull ................................................................. 47
4.1.105 SetObject ............................................................... 47
4.1.106 SetString ............................................................... 48
4.1.107 SetUint ................................................................. 48
4.1.108 SetUint64 ............................................................... 48

4.2 FBJsonDynDomParser ......................................................... 49

4.3 FBJsonSaxReader ............................................................... 49
  4.3.1 DecodeBase64 ............................................................ 49
  4.3.2 DecodeDateTime ........................................................ 50
  4.3.3 DecodeDcTime .......................................................... 50
  4.3.4 DecodeFileTime ......................................................... 50
  4.3.5 DecodeHexBinary ....................................................... 51
  4.3.6 IsBase64 ................................................................. 51
  4.3.7 IsHexBinary ............................................................. 51
  4.3.8 IsISO8601TimeFormat .................................................. 51
  4.3.9 Parse ................................................................. 52
  4.3.10 ParseValues ............................................................ 52

4.4 FBJsonSaxWriter ............................................................... 52
  4.4.1 AddBase64 ............................................................... 53
  4.4.2 AddBool ................................................................. 53
  4.4.3 AddDateTime ............................................................ 53
  4.4.4 AddDcTime ............................................................. 53
  4.4.5 AddDint ................................................................. 54
  4.4.6 AddFileTime ............................................................ 54
Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.7</td>
<td>AddHexBinary</td>
<td>54</td>
</tr>
<tr>
<td>4.4.8</td>
<td>AddKey</td>
<td>54</td>
</tr>
<tr>
<td>4.4.9</td>
<td>AddKeyBool</td>
<td>55</td>
</tr>
<tr>
<td>4.4.10</td>
<td>AddKeyDateTime</td>
<td>55</td>
</tr>
<tr>
<td>4.4.11</td>
<td>AddKeyDcTime</td>
<td>55</td>
</tr>
<tr>
<td>4.4.12</td>
<td>AddKeyFileTime</td>
<td>55</td>
</tr>
<tr>
<td>4.4.13</td>
<td>AddKeyLreal</td>
<td>56</td>
</tr>
<tr>
<td>4.4.14</td>
<td>AddKeyNull</td>
<td>56</td>
</tr>
<tr>
<td>4.4.15</td>
<td>AddKeyNumber</td>
<td>56</td>
</tr>
<tr>
<td>4.4.16</td>
<td>AddKeyString</td>
<td>56</td>
</tr>
<tr>
<td>4.4.17</td>
<td>AddLint</td>
<td>57</td>
</tr>
<tr>
<td>4.4.18</td>
<td>AddLreal</td>
<td>57</td>
</tr>
<tr>
<td>4.4.19</td>
<td>AddNull</td>
<td>57</td>
</tr>
<tr>
<td>4.4.20</td>
<td>AddRawArray</td>
<td>58</td>
</tr>
<tr>
<td>4.4.21</td>
<td>AddRawObject</td>
<td>58</td>
</tr>
<tr>
<td>4.4.22</td>
<td>AddReal</td>
<td>58</td>
</tr>
<tr>
<td>4.4.23</td>
<td>AddString</td>
<td>58</td>
</tr>
<tr>
<td>4.4.24</td>
<td>AddUdint</td>
<td>59</td>
</tr>
<tr>
<td>4.4.25</td>
<td>AddUlimit</td>
<td>59</td>
</tr>
<tr>
<td>4.4.26</td>
<td>CopyDocument</td>
<td>59</td>
</tr>
<tr>
<td>4.4.27</td>
<td>EndArray</td>
<td>60</td>
</tr>
<tr>
<td>4.4.28</td>
<td>EndObject</td>
<td>60</td>
</tr>
<tr>
<td>4.4.29</td>
<td>GetDocument</td>
<td>60</td>
</tr>
<tr>
<td>4.4.30</td>
<td>GetDocumentLength</td>
<td>60</td>
</tr>
<tr>
<td>4.4.31</td>
<td>GetMaxDecimalPlaces</td>
<td>61</td>
</tr>
<tr>
<td>4.4.32</td>
<td>ResetDocument</td>
<td>61</td>
</tr>
<tr>
<td>4.4.33</td>
<td>SetMaxDecimalPlaces</td>
<td>61</td>
</tr>
<tr>
<td>4.4.34</td>
<td>StartArray</td>
<td>61</td>
</tr>
<tr>
<td>4.4.35</td>
<td>StartObject</td>
<td>61</td>
</tr>
</tbody>
</table>

4.5 FB_JsonReadWriteDataType

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.1</td>
<td>AddJsonKeyPropertiesFromSymbol</td>
<td>62</td>
</tr>
<tr>
<td>4.5.2</td>
<td>AddJsonKeyValueFromSymbol</td>
<td>63</td>
</tr>
<tr>
<td>4.5.3</td>
<td>AddJsonValueFromSymbol</td>
<td>63</td>
</tr>
<tr>
<td>4.5.4</td>
<td>CopyJsonStringFromSymbol</td>
<td>64</td>
</tr>
<tr>
<td>4.5.5</td>
<td>CopyJsonStringFromSymbolProperties</td>
<td>64</td>
</tr>
<tr>
<td>4.5.6</td>
<td>CopySymbolNameByAddress</td>
<td>65</td>
</tr>
<tr>
<td>4.5.7</td>
<td>GetTypeTypeNameByAddress</td>
<td>65</td>
</tr>
<tr>
<td>4.5.8</td>
<td>GetJsonFromSymbol</td>
<td>65</td>
</tr>
<tr>
<td>4.5.9</td>
<td>GetJsonStringFromSymbol</td>
<td>66</td>
</tr>
<tr>
<td>4.5.10</td>
<td>GetJsonStringFromSymbolProperties</td>
<td>66</td>
</tr>
<tr>
<td>4.5.11</td>
<td>GetSizeJsonStringFromSymbol</td>
<td>67</td>
</tr>
<tr>
<td>4.5.12</td>
<td>GetSizeJsonStringFromSymbolProperties</td>
<td>67</td>
</tr>
<tr>
<td>4.5.13</td>
<td>GetSymbolNameByAddress</td>
<td>67</td>
</tr>
<tr>
<td>4.5.14</td>
<td>SetSymbolFromJson</td>
<td>68</td>
</tr>
</tbody>
</table>

4.6 FB_XmlDomParser

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6.1</td>
<td>AppendAttribute</td>
<td>69</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.6.2</td>
<td>AppendAttributeAsBool</td>
<td>69</td>
</tr>
<tr>
<td>4.6.3</td>
<td>AppendAttributeAsDouble</td>
<td>69</td>
</tr>
<tr>
<td>4.6.4</td>
<td>AppendAttributeAsFloat</td>
<td>70</td>
</tr>
<tr>
<td>4.6.5</td>
<td>AppendAttributeAsInt</td>
<td>70</td>
</tr>
<tr>
<td>4.6.6</td>
<td>AppendAttributeAsLInt</td>
<td>70</td>
</tr>
<tr>
<td>4.6.7</td>
<td>AppendAttributeAsUint</td>
<td>71</td>
</tr>
<tr>
<td>4.6.8</td>
<td>AppendAttributeAsUilint</td>
<td>71</td>
</tr>
<tr>
<td>4.6.9</td>
<td>AppendAttributeCopy</td>
<td>71</td>
</tr>
<tr>
<td>4.6.10</td>
<td>AppendChild</td>
<td>72</td>
</tr>
<tr>
<td>4.6.11</td>
<td>AppendChildAsBool</td>
<td>72</td>
</tr>
<tr>
<td>4.6.12</td>
<td>AppendChildAsDouble</td>
<td>72</td>
</tr>
<tr>
<td>4.6.13</td>
<td>AppendChildAsFloat</td>
<td>73</td>
</tr>
<tr>
<td>4.6.14</td>
<td>AppendChildAsInt</td>
<td>73</td>
</tr>
<tr>
<td>4.6.15</td>
<td>AppendChildAsLInt</td>
<td>73</td>
</tr>
<tr>
<td>4.6.16</td>
<td>AppendChildAsUilint</td>
<td>74</td>
</tr>
<tr>
<td>4.6.17</td>
<td>AppendChildAsUint</td>
<td>74</td>
</tr>
<tr>
<td>4.6.18</td>
<td>AppendCopy</td>
<td>74</td>
</tr>
<tr>
<td>4.6.19</td>
<td>AppendNode</td>
<td>75</td>
</tr>
<tr>
<td>4.6.20</td>
<td>Attributes</td>
<td>75</td>
</tr>
<tr>
<td>4.6.21</td>
<td>AttributeAsBool</td>
<td>75</td>
</tr>
<tr>
<td>4.6.22</td>
<td>AttributeAsDouble</td>
<td>75</td>
</tr>
<tr>
<td>4.6.23</td>
<td>AttributeAsFloat</td>
<td>75</td>
</tr>
<tr>
<td>4.6.24</td>
<td>AttributeAsInt</td>
<td>76</td>
</tr>
<tr>
<td>4.6.25</td>
<td>AttributeAsLInt</td>
<td>76</td>
</tr>
<tr>
<td>4.6.26</td>
<td>AttributeAsUilint</td>
<td>76</td>
</tr>
<tr>
<td>4.6.27</td>
<td>AttributeAsUilint</td>
<td>76</td>
</tr>
<tr>
<td>4.6.28</td>
<td>AttributeBegin</td>
<td>77</td>
</tr>
<tr>
<td>4.6.29</td>
<td>AttributeFromIterator</td>
<td>77</td>
</tr>
<tr>
<td>4.6.30</td>
<td>AttributeName</td>
<td>77</td>
</tr>
<tr>
<td>4.6.31</td>
<td>Attributes</td>
<td>78</td>
</tr>
<tr>
<td>4.6.32</td>
<td>AttributeText</td>
<td>78</td>
</tr>
<tr>
<td>4.6.33</td>
<td>Begin</td>
<td>78</td>
</tr>
<tr>
<td>4.6.34</td>
<td>BeginByName</td>
<td>79</td>
</tr>
<tr>
<td>4.6.35</td>
<td>Child</td>
<td>79</td>
</tr>
<tr>
<td>4.6.36</td>
<td>ChildByAttribute</td>
<td>79</td>
</tr>
<tr>
<td>4.6.37</td>
<td>ChildByAttributeAndName</td>
<td>79</td>
</tr>
<tr>
<td>4.6.38</td>
<td>ChildByName</td>
<td>80</td>
</tr>
<tr>
<td>4.6.39</td>
<td>Children</td>
<td>80</td>
</tr>
<tr>
<td>4.6.40</td>
<td>ChildrenByName</td>
<td>80</td>
</tr>
<tr>
<td>4.6.41</td>
<td>Compare</td>
<td>81</td>
</tr>
<tr>
<td>4.6.42</td>
<td>CopyAttributeText</td>
<td>81</td>
</tr>
<tr>
<td>4.6.43</td>
<td>CopyDocument</td>
<td>81</td>
</tr>
<tr>
<td>4.6.44</td>
<td>CopyNodeText</td>
<td>82</td>
</tr>
<tr>
<td>4.6.45</td>
<td>CopyNodeXml</td>
<td>82</td>
</tr>
<tr>
<td>4.6.46</td>
<td>FirstNodeByPath</td>
<td>82</td>
</tr>
<tr>
<td>4.6.47</td>
<td>GetAttributeTextLength</td>
<td>83</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.6.48</td>
<td>GetDocumentLength</td>
<td>83</td>
</tr>
<tr>
<td>4.6.49</td>
<td>GetDocumentNode</td>
<td>83</td>
</tr>
<tr>
<td>4.6.50</td>
<td>GetNodeTextLength</td>
<td>83</td>
</tr>
<tr>
<td>4.6.51</td>
<td>GetNodeXmlLength</td>
<td>84</td>
</tr>
<tr>
<td>4.6.52</td>
<td>GetRootNode</td>
<td>84</td>
</tr>
<tr>
<td>4.6.53</td>
<td>InsertAttributeCopy</td>
<td>84</td>
</tr>
<tr>
<td>4.6.54</td>
<td>InsertAttribute</td>
<td>84</td>
</tr>
<tr>
<td>4.6.55</td>
<td>InsertChild</td>
<td>85</td>
</tr>
<tr>
<td>4.6.56</td>
<td>InsertCopy</td>
<td>85</td>
</tr>
<tr>
<td>4.6.57</td>
<td>IsEnd</td>
<td>85</td>
</tr>
<tr>
<td>4.6.58</td>
<td>LoadDocumentFromFile</td>
<td>86</td>
</tr>
<tr>
<td>4.6.59</td>
<td>NewDocument</td>
<td>86</td>
</tr>
<tr>
<td>4.6.60</td>
<td>Next</td>
<td>86</td>
</tr>
<tr>
<td>4.6.61</td>
<td>NextAttribute</td>
<td>87</td>
</tr>
<tr>
<td>4.6.62</td>
<td>NextByName</td>
<td>87</td>
</tr>
<tr>
<td>4.6.63</td>
<td>NextSibling</td>
<td>87</td>
</tr>
<tr>
<td>4.6.64</td>
<td>NextSiblingByName</td>
<td>87</td>
</tr>
<tr>
<td>4.6.65</td>
<td>Node</td>
<td>88</td>
</tr>
<tr>
<td>4.6.66</td>
<td>NodeAsBool</td>
<td>88</td>
</tr>
<tr>
<td>4.6.67</td>
<td>NodeAsDouble</td>
<td>88</td>
</tr>
<tr>
<td>4.6.68</td>
<td>NodeAsFloat</td>
<td>88</td>
</tr>
<tr>
<td>4.6.69</td>
<td>NodeAsInt</td>
<td>89</td>
</tr>
<tr>
<td>4.6.70</td>
<td>NodeAsLInt</td>
<td>89</td>
</tr>
<tr>
<td>4.6.71</td>
<td>NodeAsUInt</td>
<td>89</td>
</tr>
<tr>
<td>4.6.72</td>
<td>NodeAsUUint</td>
<td>89</td>
</tr>
<tr>
<td>4.6.73</td>
<td>NodeName</td>
<td>90</td>
</tr>
<tr>
<td>4.6.74</td>
<td>NodeText</td>
<td>90</td>
</tr>
<tr>
<td>4.6.75</td>
<td>ParseDocument</td>
<td>90</td>
</tr>
<tr>
<td>4.6.76</td>
<td>RemoveChild</td>
<td>90</td>
</tr>
<tr>
<td>4.6.77</td>
<td>RemoveChildByName</td>
<td>90</td>
</tr>
<tr>
<td>4.6.78</td>
<td>SaveDocumentToFile</td>
<td>91</td>
</tr>
<tr>
<td>4.6.79</td>
<td>SetAttribute</td>
<td>91</td>
</tr>
<tr>
<td>4.6.80</td>
<td>SetAttributeAsBool</td>
<td>92</td>
</tr>
<tr>
<td>4.6.81</td>
<td>SetAttributeAsDouble</td>
<td>92</td>
</tr>
<tr>
<td>4.6.82</td>
<td>SetAttributeAsFloat</td>
<td>92</td>
</tr>
<tr>
<td>4.6.83</td>
<td>SetAttributeAsInt</td>
<td>92</td>
</tr>
<tr>
<td>4.6.84</td>
<td>SetAttributeAsLint</td>
<td>93</td>
</tr>
<tr>
<td>4.6.85</td>
<td>SetAttributeAsUInt</td>
<td>93</td>
</tr>
<tr>
<td>4.6.86</td>
<td>SetAttributeAsUUint</td>
<td>93</td>
</tr>
<tr>
<td>4.6.87</td>
<td>SetChild</td>
<td>93</td>
</tr>
<tr>
<td>4.6.88</td>
<td>SetChildAsBool</td>
<td>94</td>
</tr>
<tr>
<td>4.6.89</td>
<td>SetChildAsDouble</td>
<td>94</td>
</tr>
<tr>
<td>4.6.90</td>
<td>SetChildAsFloat</td>
<td>94</td>
</tr>
<tr>
<td>4.6.91</td>
<td>SetChildAsInt</td>
<td>95</td>
</tr>
<tr>
<td>4.6.92</td>
<td>SetChildAsLint</td>
<td>95</td>
</tr>
<tr>
<td>4.6.93</td>
<td>SetChildAsLInt</td>
<td>95</td>
</tr>
</tbody>
</table>
1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards. It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components. It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning. The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development. We reserve the right to revise and change the documentation at any time and without prior announcement. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany. The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations
Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability
All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification
This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols
In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="danger.png" alt="DANGER" /></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="warning.png" alt="WARNING" /></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="caution.png" alt="CAUTION" /></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="note.png" alt="NOTE" /></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer
This symbol indicates information that contributes to better understanding.
1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.
Overview

2 Overview

With the aid of the PLC library Tc3_JsonXml, SAX and DOM parser technologies can be used to create and navigate through JSON and XML documents.

System requirements

<table>
<thead>
<tr>
<th>Target System</th>
<th>Win7, WES7, WEC7, Win10 IPC or CX, (x86, x64, ARM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. TwinCAT version</td>
<td>3.1.4022.0</td>
</tr>
<tr>
<td>Min. TwinCAT level</td>
<td>TC1200 TC3 PLC</td>
</tr>
</tbody>
</table>

SAX (simple API for XML)

SAX was originally developed for handling XML documents, but can also be used for other data formats such as JSON. A SAX parser treats the data to be read or written as a sequential data stream. When reading a data stream, defined callback methods are called, which then return the corresponding contents of the data stream. A SAX parser is therefore also referred to as an event-based parser. The events occurring (callback methods) are stateless, i.e. they are not dependent on the preceding events. The advantage of this is that the XML document never has to be completely contained in the memory and the application can react "on the fly" via the callbacks.

DOM (Document Object Model)

DOM is a specification for accessing XML documents, but can also be used for other data formats such as HTML or JSON. The interface is based on a defined object model, whose validity is a prerequisite for correct use. This object model represents a document, e.g. a JSON document, in the form of a tree structure in the memory, which can then be used to navigate through the document. DOM allows navigation between the individual nodes, the creation, moving and deletion of nodes, and the reading, changing and deletion of node contents. When editing is finished a new JSON or XML document is generated from the finalized tree structure. The advantage here is that no own data housekeeping needs to be created with the read-in data, since the data exist in the DOM and can be continuously accessed.

JSON document

The following section shows a JSON document as an example:

```
{
    "VariableNameX": 0.0,
    "VariableNameY": 0.0,
    "VariableNameZ": 0.0
}
```

Metadata

The Tc3_JsonXml library contains the function block FB_JsonReadWriteDataType [61], which enables automatic generation of metadata by means of PLC attributes.

```
{
    "Values": {
        "VariableNameX": 0.0,
        "VariableNameY": 0.0,
        "VariableNameZ": 0.0
    },
    "MetaData": {
        "VariableNameX": {
            "Unit": "A"
        },
        "VariableNameY": {
            "Unit": "V"
        },
        "VariableNameZ": {
            "Unit": "mA"
        }
    }
}```
See also: Examples > Tc3JsonXmlSampleJsonDataType [p. 103].

In order to use UTF-8 characters, e.g. in the automatic generation of metadata via the function block FB_JsonReadWriteDataType [p. 61], the check box for the support of UTF-8 in the symbolism must be activated in the TwinCAT project. To do this, double-click on SYSTEM in the project tree, open the Settings tab and activate the corresponding check box.

JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard (based on RFC 7519) that defines a compact and self-describing format for securely transmitting information between communication devices in the form of a JSON object. The authenticity of the transmitted information can be verified and ensured, since a JWT is provided with a digital signature. The signature can involve a shared secret (via an HMAC algorithm) or a public/private key (via RSA).

The most common application example for JWT is the authorization of a device or user for a service. Once a user has logged into the service, all further requests to the service include the JWT. Based on the JWT, the service can then decide which additional services or resources the user may access. This means, for example, that single sign-on solutions can be implemented in cloud services.

The Tc3_JsonXml library provides a function for creating a JWT via the FB_JwtEncode [p. 96] method.
3 Getting started

This documentation article is intended as a quick-start guide to using the Tc3_JsonXml PLC library. The article is based on the following use case:

• A JSON document exists that is to be processed further.
• The JSON document is in the form of a file on the local file system.
• The article describes the steps to parse the JSON document.

For more code samples, please refer to our samples [103].

File contents

The file myJsonContent.json is located in the local file system under the following path: c:\temp\myJsonContent.json and has the following content:

```json
{
  "Timestamp":"2017-04-04T12:42:42",
  "Values": {
    "Sensor1": 42.41999816894531,
    "Sensor2": [1,2,3,4,5],
    "Sensor3": 10
  }
}
```

Declarations

The following variable declarations are required for the further procedure:

```plaintext
fbJson : FB_JsonDomParser;
jsonRoot : SJsonValue;
jsonProp : SJsonValue;
jsonIterator : SJsonValue;
jsonIteratorEnd : SJsonValue;
bLoadJsonFile : BOOL;
sTimestamp : STRING;
fSensor1 : LREAL;
aSensor2 : ARRAY[0..4] OF DINT;
nSensor3 : DINT;
i : UINT;
```

Furthermore, please make sure that you have added a reference to the PLC library Tc3_JsonXml in your PLC project.

Reading the file

The method LoadDocumentFromFile [37] from the function block FB_JsonDomParser [19] is used to read the file. The execution of the method is controlled by a rising edge at input bExec.

```plaintext
IF bLoadJsonFile = TRUE THEN
  fbJson.LoadDocumentFromFile('C:\Temp\myJsonContent.json', bLoadJsonFile);
END_IF
```

Source of the JSON document

If in your use case the JSON document already exists in the PLC, e.g. in a variable of data type STRING, you can use the method ParseDocument [38] to load the document into memory and for further processing.

Parsing the JSON document

The GetDocumentRoot [29] method can be used to reference the beginning of the JSON document in memory. The return value of the method is an interface pointer.

```plaintext
jsonRoot := fbJson.GetDocumentRoot();
```

From here, the other keys on the first level can be read, e.g. the 'Timestamp' key via the method GetString [32].
Getting started

The next key ('Values') is a nested JSON object. The child elements of this object can be read directly via the method `FindMemberPath`[^27], for example for the element 'Values/Sensor1':

```pascal
jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor1');
IF (jsonProp <> 0) THEN
  fSensor1 := fbJson.GetDouble(jsonProp);
END_IF
```

The next child element ('Values/Sensor2') is an array. This can be read by using the methods `ArrayBegin`[^24] and `ArrayEnd`[^24].

```pascal
jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor2');
IF (jsonProp <> 0) THEN
  jsonIterator := fbJson.ArrayBegin(jsonProp);
  jsonIteratorEnd := fbJson.ArrayEnd(jsonProp);
  WHILE jsonIterator <> jsonIteratorEnd DO
    IF (jsonProp <> 0) THEN
      aSensor2[i] := fbJson.GetInt(jsonIterator);
    END_IF
    jsonIterator := fbJson.NextArray(jsonIterator);
  END_WHILE
  i := 0;
END_IF
```

Handling of the next child element ('Values/Sensor3') is similar to the element 'Values/Sensor1'. Instead of the method `GetDouble`[^29] the method `GetInt`[^30] is used to read the value of the key.

```pascal
jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor3');
IF (jsonProp <> 0) THEN
  nSensor3 := fbJson.GetInt(jsonProp);
END_IF
```

Complete code sample

Below you will find the above code snippets as a complete sample.

Declaration part

```pascal
PROGRAM MAIN
  fbJson : FB_JsonDomParser;
  jsonRoot : SJsonValue;
  jsonProp : SJsonValue;
  jsonIterator : SJsonValue;
  jsonIteratorEnd : SJsonValue;
  bLoadJsonFile : BOOL;
  sTimestamp : STRING;
  fSensor1 : LREAL;
  aSensor2 : ARRAY[0..4] OF DINT;
  nSensor3 : DINT;
  i : UINT;
END_VAR
```

Implementation part

```pascal
IF bLoadJsonFile = TRUE THEN
  fbJson.LoadDocumentFromFile('C:\Temp\myJsonContent.json', bLoadJsonFile);
  jsonRoot := fbJson.GetDocumentRoot();
  jsonProp := fbJson.FindMember(jsonRoot, 'Timestamp');
  IF (jsonProp <> 0) THEN
    sTimestamp := fbJson.GetString(jsonProp);
  END_IF
  jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor1');
  IF (jsonProp <> 0) THEN
    fSensor1 := fbJson.GetDouble(jsonProp);
  END_IF
  jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor2');
  IF (jsonProp <> 0) THEN
    jsonIterator := fbJson.ArrayBegin(jsonProp);
    jsonIteratorEnd := fbJson.ArrayEnd(jsonProp);
    WHILE jsonIterator <> jsonIteratorEnd DO
      IF (jsonProp <> 0) THEN
        aSensor2[i] := fbJson.GetInt(jsonIterator);
      END_IF
      jsonIterator := fbJson.NextArray(jsonIterator);
    END_WHILE
    i := 0;
  END_IF
  jsonProp := fbJson.FindMemberPath(jsonRoot, 'Values/Sensor3');
  IF (jsonProp <> 0) THEN
    nSensor3 := fbJson.GetInt(jsonProp);
  END_IF
```
After the above code has been executed successfully, the keys that were read out are in the corresponding PLC variables, e.g. as follows:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbJson</td>
<td>FB_JsDomParser</td>
<td></td>
</tr>
<tr>
<td>jsonDoc</td>
<td>SJsonValue</td>
<td>0000000000000000...</td>
</tr>
<tr>
<td>jsonRoot</td>
<td>SJsonValue</td>
<td>fffbf209f5807508</td>
</tr>
<tr>
<td>jsonProp</td>
<td>SJsonValue</td>
<td>fffbf209f58083f8</td>
</tr>
<tr>
<td>jsonValue</td>
<td>SJsonValue</td>
<td>0000000000000000...</td>
</tr>
<tr>
<td>jsonIterator</td>
<td>SJsonValue</td>
<td>fffbf209f5808380</td>
</tr>
<tr>
<td>jsonIteratorEnd</td>
<td>SJsonValue</td>
<td>fffbf209f5808380</td>
</tr>
<tr>
<td>bHasMember</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>bLoadJsonFile</td>
<td>BOOL</td>
<td>FALSE</td>
</tr>
<tr>
<td>sTimestamp</td>
<td>STRING</td>
<td>2017-04-04T12:...</td>
</tr>
<tr>
<td>fSensor1</td>
<td>LREAL</td>
<td>42.91999815894...</td>
</tr>
<tr>
<td>arrSensor2</td>
<td>ARRAY [0..4] OF DINT</td>
<td></td>
</tr>
<tr>
<td>arrSensor2[0]</td>
<td>DINT</td>
<td>1</td>
</tr>
<tr>
<td>arrSensor2[1]</td>
<td>DINT</td>
<td>2</td>
</tr>
<tr>
<td>arrSensor2[2]</td>
<td>DINT</td>
<td>3</td>
</tr>
<tr>
<td>arrSensor2[3]</td>
<td>DINT</td>
<td>4</td>
</tr>
<tr>
<td>nSensor3</td>
<td>DINT</td>
<td>10</td>
</tr>
<tr>
<td>i</td>
<td>UINT</td>
<td>0</td>
</tr>
</tbody>
</table>
4 Function blocks

4.1 FBToJsonDomParser

This function block is derived from the same internal function block as FBJsonDynDomParser and thus offers the same interface.

The two derived function blocks differ only in their internal memory management. FBToJsonDomParser is optimized for the fast and efficient parsing and creation of JSON documents that are only little changed. The function block FBJsonDynDomParser is recommended for JSON documents to which many changes are made.

WARNING
Use of router memory
The function block occupies new memory with each change, e.g. with the methods SetObject() or SetJson(). As a result, the amount of router memory used can grow enormously after repeated actions. This allocated memory is only released again by calling the method NewDocument().

Strings in UTF-8 format
The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication as well as for JSON documents.

In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotBase and Tc3_JsonXml libraries is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

Further information on the UTF-8 STRING format and available display and conversion options can be found in the documentation for the Tc2_Utilities PLC library.

Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4022</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml</td>
</tr>
</tbody>
</table>

4.1.1 AddArrayMember

This method adds an array member to a JSON object.

Syntax

METHOD AddArrayMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR
VAR_INPUT
  reserve : UDINT;
END_VAR

Sample call:

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.AddArrayMember(jsonDoc, 'TestArray', 0);
```
4.1.2 AddBase64Member

This method adds a Base64 member to a JSON object. A structure, for example, can be addressed as an input parameter. The corresponding Base64 coding is done by the method.

Syntax

METHOD AddBase64Member : SJsonValue
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonBase64 := fbJson.AddBase64Member(jsonDoc, 'TestBase64', ADR(stStruct), SIZEOF(stStruct));

4.1.3 AddBoolMember

This method adds a Bool member to a JSON object.

Syntax

METHOD AddBoolMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : BOOL;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddBoolMember(jsonDoc, 'TestBool', TRUE);

4.1.4 AddDateTimeMember

This method adds a DateTime member to a JSON object.

Syntax

METHOD AddDateTimeMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DATE_AND_TIME;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddDateTimeMember(jsonDoc, 'TestDateTime', DT#2018-11-22-12:12);

4.1.5 AddDcTimeMember

This method adds a DcTime member to a JSON object.
4.1.6 AddDcTimeMember

This method adds a DcTime member to a JSON object.

Syntax

METHOD AddDcTimeMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DCTIME;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddDcTimeMember(jsonDoc, 'TestDcTime', 1234);
```

4.1.7 AddFileTimeMember

This method adds a FileTime member to a JSON object.

Syntax

METHOD AddFileTimeMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : FILETIME;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddFileTimeMember(jsonDoc, 'TestFileTime', ftTime);
```

4.1.8 AddHexBinaryMember

This method adds a HexBinary member to a JSON object.

Syntax

METHOD AddHexBinaryMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR
Sample call:
```perl
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddHexBinaryMember(jsonDoc, 'TestHexBinary', sHexBinary, SIZEOF(sHexBinary));
```

### 4.1.9 AddInt64Member

This method adds an Int64 member to a JSON object.

**Syntax**

```plaintext
METHOD AddFileTimeMember : SJsonValue
VAR_INPUT
    v : SJsonValue;
    value : LINT;
END_VAR
VAR_IN_OUT CONSTANT
    member : STRING;
END_VAR
```

Sample call:
```/perl
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddInt64Member(jsonDoc, 'TestInt64', 42);
```

### 4.1.10 AddIntMember

This method adds an Int member to a JSON object.

**Syntax**

```plaintext
METHOD AddIntMember : SJsonValue
VAR_INPUT
    v : SJsonValue;
    value : DINT;
END_VAR
VAR_IN_OUT CONSTANT
    member : STRING;
END_VAR
```

Sample call:
```perl
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddIntMember(jsonDoc, 'TestInt', 42);
```

### 4.1.11 AddJsonMember

This method adds a JSON member to a JSON object.

**Syntax**

```plaintext
METHOD AddJsonMember : SJsonValue
VAR_INPUT
    v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
    member : STRING;
    rawJson : STRING;
END_VAR
```

Sample call:
```perl
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddJsonMember(jsonDoc, 'TestJson', sJson);
```

### 4.1.12 AddNullMember

This method adds a NULL member to a JSON object.
Syntax
METHOD AddNullMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddNullMember(jsonDoc, 'TestJson');

4.1.13 AddObjectMember
This method adds an Object member to a JSON object.

Syntax
METHOD AddObjectMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddObjectMember(jsonDoc, 'TestObject');

4.1.14 AddStringMember
This method adds a String member to a JSON object.

Syntax
METHOD AddStringMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
  value : STRING;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddStringMember(jsonDoc, 'TestString', 'Test');

4.1.15 AddUint64Member
This method adds an UInt64 member to a JSON object.

Syntax
METHOD AddUint64Member : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : ULINT;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:
4.1.16 AddUintMember

This method adds an UInt member to a JSON object.

Syntax

METHOD AddUintMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : UDINT;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonMem := fbJson.AddUintMember(jsonDoc, 'TestUint', 42);

4.1.17 ArrayBegin

This method returns the first element of an array and can be used together with the methods ArrayEnd() and NextArray() for iteration through a JSON array.

Syntax

METHOD ArrayBegin : SJsonAIterator
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:

jsonIterator := fbJson.ArrayBegin(jsonArray);
jsonIteratorEnd := fbJson.ArrayEnd(jsonArray);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetArrayValue(jsonIterator);
  jsonIterator := fbJson.NextArray(jsonIterator);
END_WHILE

4.1.18 ArrayEnd

This method returns the last element of an array and can be used together with the methods ArrayBegin() and NextArray() for iteration through a JSON array.

Syntax

METHOD ArrayEnd : SJsonAIterator
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:

jsonIterator := fbJson.ArrayBegin(jsonArray);
jsonIteratorEnd := fbJson.ArrayEnd(jsonArray);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetArrayValue(jsonIterator);
  jsonIterator := fbJson.NextArray(jsonIterator);
END_WHILE
4.1.19 ClearArray

This method deletes the content of an array.

**Syntax**

METHOD ClearArray : BOOL
VAR_INPUT
  v : SJsonValue;
  i : SJsonAIterator;
END_VAR

**Sample call:**

The following JSON document is to be loaded into the DOM memory:

```
sMessage := '{"serialNumber"="123","batteryVoltage":1547mV","clickType":"SINGLE", "array":
["Hello",2,3]}';
```

The values of the JSON array "array" are to be deleted. First of all, the JSON document is searched iteratively for the "array" property, after which all elements of the array are deleted by calling the ClearArray() method.

```
jsonDoc := fbJson.ParseDocument(sMessage);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetMemberName(jsonIterator);
  jsonValue := fbJson.GetMemberValue(jsonIterator);
  IF sName = 'array' THEN
    jsonArrayIterator := fbJson.ArrayBegin(jsonValue);
    fbJson.ClearArray(jsonValue, jsonArrayIterator);
  END_IF
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

4.1.20 CopyDocument

This method copies the contents of the DOM memory into a variable of data type STRING, which can have any length. The method returns the length of the string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

METHOD CopyDocument : UDINT
VAR_INPUT
  nDoc : DINT;
END_VAR
VAR_IN_OUT CONSTANT
  pDoc : STRING;
END_VAR

**Sample call:**

```
nLen := fbJson.CopyDocument(sJson, SIZEOF(sJson));
```

4.1.21 CopyJson

This method extracts a JSON object from a key and stores it in a variable of data type STRING. This STRING can be of any length. The method returns the length of the copied JSON object (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

METHOD CopyJson : UDINT
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
Sample call:

The following JSON document is to be loaded into the DOM memory:

```
sMessage := '{"serialNumber":"123","meta":{"batteryVoltage":"1547mV","clickType":"SINGLE"}}';
```

The value of the JSON object "meta" is to be extracted and stored in a variable of data type STRING. First, the JSON document is searched iteratively for the property "meta", then its value or sub-object is extracted by calling the method CopyJson().

```
jsonDoc := fbJson.ParseDocument(sMessage);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetMemberName(jsonIterator);
  jsonValue := fbJson.GetMemberValue(jsonIterator);
  IF sName = 'meta' THEN
    fbJson.CopyJson(jsonValue, sString, SIZEOF(sString));
  END_IF
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

After this run, the sString variable has the following content:

```
{"batteryVoltage":"1547mV","clickType":"SINGLE"}
```

### 4.1.22 CopyString

This method copies the value of a key into a variable of the data type STRING, which can be of any length. The method returns the length of the copied string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

```
METHOD CopyString : UDINT
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  pStr : STRING;
  nStr : UDINT;
END_VAR
```

Sample call:

The following JSON document is to be loaded into the DOM memory:

```
sMessage := '{"serialNumber":"123","batteryVoltage":"1547mV","clickType":"SINGLE"}';
```

The value of the key "clickType" is to be extracted and stored in a variable of data type STRING. First, the JSON document is iteratively searched for the property "clickType".

```
jsonDoc := fbJson.ParseDocument(sMessage);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetMemberName(jsonIterator);
  jsonValue := fbJson.GetMemberValue(jsonIterator);
  IF sName = 'clickType' THEN
    fbJson.CopyString(jsonValue, sString, SIZEOF(sString));
  END_IF
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

After this run, the sString variable has the following content:

```
SINGLE
```
4.1.23  FindMember

This method searches for a specific property in a JSON document and returns it. 0 is returned if no corresponding property is found.

**Syntax**

```plaintext
METHOD FindMember : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:

```json
jsonProp := fbJson.FindMember(jsonDoc, 'PropertyName');
```
Function blocks

```plaintext
jsonIterator := fbJson.ArrayBegin(jsonArray);
jsonIteratorEnd := fbJson.ArrayEnd(jsonArray);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetArrayValue(jsonIterator);
  jsonIterator := fbJson.NextArray(jsonIterator);
END_WHILE
```

### 4.1.27 GetArrayValueByIdx

This method returns the value of an array in a specified index.

**Syntax**

```
METHOD GetArrayValueByIdx : SJsonValue
VAR_INPUT
  v : SJsonValue;
  idx : UDINT;
END_VAR
```

**Sample call:**

```plaintext
jsonValue := fbJson.GetArrayValueByIdx(jsonArray, 1);
```

### 4.1.28 GetBase64

This method decodes a Base64 value from a JSON property. If the content of a data structure, for example, is located behind the Base64 value, the decoded content can also be placed on an identical structure again.

**Syntax**

```
METHOD GetBase64 : DINT
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonBase64 := fbJson.FindMember(jsonDoc, 'base64');
nSize := fbJson.GetBase64(jsonBase64, ADR(stStruct), SIZEOF(stStruct));
```

### 4.1.29 GetBool

This method returns the value of a property of the data type BOOL.

**Syntax**

```
METHOD GetBool : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR
```

### 4.1.30 GetDateTime

This method returns the value of a property of the data type DATE_AND_TIME.

**Syntax**

```
METHOD GetDateTime : DATE_AND_TIME
VAR_INPUT
  v : SJsonValue;
END_VAR
```
4.1.31  GetDcTime

This method returns the value of a property of the data type DCTIME.

Syntax

METHOD GetDcTime : DCTIME
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
dcTime := fbJson.GetDcTime(jsonProp);

4.1.32  GetDocument

This method returns the content of the DOM memory as the data type STRING(255). With longer strings, the method will return a NULL string. In this case the method CopyDocument must be used.

Syntax

METHOD GetDocument : STRING(255)

Sample call:

sJson := fbJson.GetDocument();

4.1.33  GetDocumentLength

This method returns the length of a JSON document in the DOM memory.

Syntax

METHOD GetDocumentLength : UDINT

Sample call:

nLen := fbJson.GetDocumentLength();

4.1.34  GetDocumentRoot

This method returns the root node of a JSON document in the DOM memory.

Syntax

METHOD GetDocumentRoot : SJsonValue

Sample call:

jsonRoot := fbJson.GetDocumentRoot();

4.1.35  GetDouble

This method returns the value of a property of the data type LREAL.

Syntax

METHOD GetDouble : LREAL
VAR_INPUT
  v : SJsonValue;
END_VAR
4.1.36  GetFileTime

This method returns the value of a property of the data type DCTIME.

**Syntax**

METHOD GetFileTime : FILETIME
VAR_INPUT
  v : SJsonValue;
END_VAR

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
fileTime := fbJson.GetFileTime(jsonProp);
```

4.1.37  GetHexBinary

This method decodes the HexBinary content of a property and writes it to a certain memory address, e.g. to a data structure.

**Syntax**

METHOD GetHexBinary : DINT
VAR_INPUT
  v : SJsonValue;
p : PVOID;
n : DINT;
END_VAR

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
nLen := fbJson.GetHexBinary(jsonProp, ADR(stStruct), SIZEOF(stStruct));
```

4.1.38  GetInt

This method returns the value of a property of the data type DINT.

**Syntax**

METHOD GetInt : DINT
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.39  GetInt64

This method returns the value of a property of the data type LINT.

**Syntax**

METHOD GetInt64 : LINT
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.40  GetJson

This method returns the value of a property as data type STRING(255), if this is a JSON document itself. With longer strings, the method will return a NULL string. In this case the method `CopyJson[string]` must be used.
Syntax
METHOD GetJson : STRING(255)
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
sJson := fbJson.GetJson(jsonProp);

4.1.41  GetJsonLength

This method returns the length of a property if this is a JSON document.

Syntax
METHOD GetJsonLength : UDINT
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
nLen := fbJson.GetJsonLength(jsonProp);

4.1.42  GetMaxDecimalPlaces

This method returns the current setting for MaxDecimalPlaces. This influences the number of decimal places in the case of floating point numbers.

Syntax
METHOD GetMaxDecimalPlaces : DINT

Sample call:
nDec := fbJson.GetMaxDecimalPlaces();

4.1.43  GetMemberName

This method returns the name of a JSON property member at the position of the current iterator, e.g. during the iteration of a child element of a JSON property with the methods MemberBegin(), MemberEnd() and NextMember().

Syntax
METHOD GetMemberName : STRING
VAR_INPUT
  i : SJsonIterator;
END_VAR

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetMemberName(jsonIterator);
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
4.1.44 GetMemberValue

This method returns the value of a JSON property member at the position of the current iterator, e.g. during the iteration of a child element of a JSON property with the methods MemberBegin(), MemberEnd() and NextMember().

**Syntax**

```
METHOD GetMemberValue : SJsonValue
  VAR_INPUT
  i : SJsonIterator;
END_VAR
```

**Sample call:**

```pascal
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  jsonValue := fbJson.GetMemberValue(jsonIterator);
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

4.1.45 GetString

This method returns the value of a property of the data type STRING(255). With longer strings, the method will return a NULL string. In this case the method `CopyString` must be used.

**Syntax**

```
METHOD GetString : STRING(255)
  VAR_INPUT
  v : SJsonValue;
END_VAR
```

4.1.46 GetStringLength

This method returns the length of a property if its value is a string.

**Syntax**

```
METHOD GetStringLength : UDINT
  VAR_INPUT
  v : SJsonValue
END_VAR
```

**Sample call:**

```pascal
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
nLen := fbJson.GetStringLength(jsonProp);
```

4.1.47 GetType

This method returns the type of a property value. The return value can assume one of the values of the enum EJsonType.

**Syntax**

```
METHOD GetType : EJsonType
  VAR_INPUT
  v : SJsonValue
END_VAR
```

**TYPE EJsonType**

```pascal
{ eNullType := 0,
  eBooleanType := 1,
  eIntegerType := 2,
  eRealType := 3,
  eStringType := 4,
  eArrayType := 5,
  eObjectArray := 6,
  eObject := 7,
  eNull => eNullType,
  eBoolean => eBooleanType,
  eInteger => eIntegerType,
  eReal => eRealType,
  eString => eStringType,
  eArray => eArrayType,
  eObjectArray => eObjectArray,
  eObject => eObject,
}
eFalseType := 1,
eTrueType := 2,
eObjectType := 3,
eArrayType := 4,
eStringType := 5,
eNumberType := 6
) DINT;

Sample call:
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
eJsonType := fbJson.GetType(jsonDocProp);

### 4.1.48 GetUint

This method returns the value of a property of the data type UDINT.

**Syntax**

METHOD GetUint : UDINT
VAR_INPUT
  v : SJsonValue;
END_VAR

### 4.1.49 GetUint64

This method returns the value of a property of the data type ULINT.

**Syntax**

METHOD GetUint64 : ULINT
VAR_INPUT
  v : SJsonValue;
END_VAR

### 4.1.50 HasMember

This method checks whether a certain property is present in the DOM memory. If the property is present the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD HasMember : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

Sample call:
bHasMember := fbJson.HasMember(jsonDoc, 'PropertyName');

### 4.1.51 IsArray

This method checks whether a given property is an array. If that is the case, the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD IsArray : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR
4.1.52 IsBase64

This method checks whether the value of a given property is of the data type Base64. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD IsBase64: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.53 IsBool

This method checks whether the value of a given property is of the data type BOOL. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD IsBool: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.54 IsDouble

This method checks whether the value of a given property is of the data type Double (PLC: LREAL). If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD IsDouble: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.55 IsFalse

This method checks whether the value of a given property is FALSE. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD IsFalse: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.56 IsHexBinary

This method checks whether the value of a property is in the HexBinary format. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD IsHexBinary: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

Sample call:
4.1.57  IsInt

This method checks whether the value of a given property is of the data type Integer (PLC: DINT). If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsInt: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.58  IsInt64

This method checks whether the value of a given property is of the data type LINT. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsInt64: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.59  IsISO8601TimeFormat

This method checks whether the value of a given property has a time format according to ISO8601. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsISO8601TimeFormat: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.60  IsNull

This method checks whether the value of a given property is NULL. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsNull: BOOL
VAR_INPUT
  v: SJsonValue;
END_VAR

4.1.61  IsNumber

This method checks whether the value of a given property is a numerical value. If that is the case, the method returns TRUE, otherwise it returns FALSE.
Syntax
METHOD IsNumber : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.62 IsObject
This method checks whether the given property is a further JSON object. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsObject : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.63 IsString
This method checks whether the value of a given property is of the data type STRING. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsString : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.64 IsTrue
This method checks whether the value of a given property is TRUE. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsTrue : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.65 IsUint
This method checks whether the value of a given property is of the data type UDINT. If that is the case, the method returns TRUE, otherwise it returns FALSE.

Syntax
METHOD IsUint : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR

4.1.66 IsUint64
This method checks whether the value of a given property is of the data type ULINT. If that is the case, the method returns TRUE, otherwise it returns FALSE.
**4.1.67 LoadDocumentFromFile**

This method loads a JSON document from a file.

A rising edge on the input parameter `bExec` triggers the loading procedure. The asynchronous process is terminated as soon as the reference `bExec` is set back to FALSE from the method. When the process ends, the return value of the method indicates for one call whether the loading of the file was successful (TRUE) or failed (FALSE).

**Syntax**

```plaintext
METHOD LoadDocumentFromFile : BOOL
VAR_IN_OUT CONSTANT
  sFile : STRING;
END_VAR
VAR_INPUT
  bExec : REFERENCE TO BOOL;
END_VAR
VAR_OUTPUT
  hrErrorCode: HRESULT;
END_VAR
```

**Sample call:**

```plaintext
IF bLoad THEN
  bLoaded := fbJson.LoadDocumentFromFile(sFile, bLoad);
END_IF
```

**4.1.68 MemberBegin**

This method returns the first child element below a JSON property and can be used by a JSON property together with the methods `MemberEnd()` and `NextMember()` for iteration.

**Syntax**

```plaintext
METHOD MemberBegin : SJsonIterator
VAR_INPUT
  v : SJsonValue;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetName(jsonIterator);
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

**4.1.69 MemberEnd**

This method returns the last child element below a JSON property and can be used by a JSON property together with the methods `MemberBegin()` and `NextMember()` for iteration.

**Syntax**

```plaintext
METHOD MemberEnd : SJsonIterator
VAR_INPUT
  v : SJsonValue;
END_VAR
```
Sample call:
```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
    sName := fbJson.GetMemberName(jsonIterator);
    jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

4.1.70  **NewDocument**

This method generates a new empty JSON document in the DOM memory.

**Syntax**

METHOD NewDocument : SJsonValue

Sample call:
```plaintext
jsonDoc := fbJson.NewDocument();
```

4.1.71  **NextArray**

4.1.72  **ParseDocument**

This method loads a JSON object into the DOM memory for further processing. The JSON object takes the form of a string and is transferred to the method as an input. A reference to the JSON document in the DOM memory is returned to the caller.

**Syntax**

METHOD ParseDocument : SJsonValue

Sample call:
```plaintext
jsonDoc := fbJson.ParseDocument(sJsonString);
```

4.1.73  **PushbackBase64Value**

This method appends a Base64 value to the end of an array. A structure, for example, can be addressed as an input parameter. The corresponding Base64 coding is done by the method.

**Syntax**

METHOD PushbackBase64Value : SJsonValue

Sample call:
```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackBase64Value(jsonArray, ADR(stStruct), SIZEOF(stStruct));
```

4.1.74  **PushbackBoolValue**

This method appends a value of the data type BOOL to the end of an array.
Syntax

METHOD PushbackBoolValue : SJsonValue
VAR_INPUT
  v    : SJsonValue;
  value : BOOL;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackBoolValue(jsonArray, TRUE);

4.1.75 PushbackDateTimeValue

This method appends a value of the data type DATE_AND_TIME to the end of an array.

Syntax

METHOD PushbackDateTimeValue : SJsonValue
VAR_INPUT
  v    : SJsonValue;
  value : DATE_AND_TIME;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackDateTimeValue(jsonArray, dtTime);

4.1.76 PushbackDcTimeValue

This method appends a value of the data type DCTIME to the end of an array.

Syntax

METHOD PushbackDcTimeValue : SJsonValue
VAR_INPUT
  v    : SJsonValue;
  value : DCTIME;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackDcTimeValue(jsonArray, dcTime);

4.1.77 PushbackDoubleValue

This method appends a value of the data type Double to the end of an array.

Syntax

METHOD PushbackDoubleValue : SJsonValue
VAR_INPUT
  v    : SJsonValue;
  value : LREAL;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackDoubleValue(jsonArray, 42.42);
4.1.78 PushbackFileTimeValue

This method appends a value of the data type FILETIME to the end of an array.

Syntax

METHOD PushbackFileTimeValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : FILETIME;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackFileTimeValue(jsonArray, fileTime);

4.1.79 PushbackHexBinaryValue

This method appends a HexBinary value to the end of an array. The coding in the HexBinary format is executed by the method. A data structure, for example, can be used as the source.

Syntax

METHOD PushbackHexBinaryValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackHexBinaryValue(jsonArray, ADR(stStruct), SIZEOF(stStruct));

4.1.80 PushbackInt64Value

This method appends a value of the data type Int64 to the end of an array.

Syntax

METHOD PushbackInt64Value : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : LINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackInt64Value(jsonArray, 42);

4.1.81 PushbackIntValue

This method appends a value of the data type INT to the end of an array.

Syntax

METHOD PushbackIntValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DINT;
END_VAR
### 4.1.82 PushbackJsonValue

This method appends a JSON document to the end of an array.

**Syntax**

```
METHOD PushbackJsonValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  rawJson : STRING;
END_VAR
```

**Sample call:**

```
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackJsonValue(jsonArray, sJson);
```

### 4.1.83 PushbackNullValue

This method appends a NULL value to the end of an array.

**Syntax**

```
METHOD PushbackNullValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
```

**Sample call:**

```
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackNullValue(jsonArray);
```

### 4.1.84 PushbackStringValue

This method appends a value of the data type DCTIME to the end of an array.

**Syntax**

```
METHOD PushbackStringValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  value : STRING;
END_VAR
```

**Sample call:**

```
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackStringValue(jsonArray, sString);
```

### 4.1.85 PushbackUint64Value

This method appends a value of the data type UInt64 to the end of an array.
**4.1.86 PushbackUintValue**

This method appends a value of the data type UInt to the end of an array.

**Syntax**

```plaintext
METHOD PushbackUintValue : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : UDINT;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonArray := fbJson.FindMember(jsonDoc, 'array');
jsonValue := fbJson.PushbackUintValue(jsonArray, 42);
```

**4.1.87 RemoveAllMembers**

This method removes all child elements from a given property.

**Syntax**

```plaintext
METHOD RemoveAllMembers : BOOL
VAR_INPUT
  v : SJsonValue;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
bRemoved := fbJson.RemoveAllMembers(jsonProp);
```

**4.1.88 RemoveArray**

This method deletes the value of the current array iterator.

**Syntax**

```plaintext
METHOD RemoveArray : BOOL
VAR_INPUT
  v : SJsonValue;
  i : SJsonAIterator;
END_VAR
```

**Sample call:**

The following JSON document is to be loaded into the DOM memory:

```plaintext
sMessage := '{"serialNumber":"123","batteryVoltage":"1547mV","clickType":"SINGLE","array": ["Hello",2,3]}';
```

The first array position is to be deleted. First of all, the JSON document is searched iteratively for the "array" property, after which the first element of the array is removed by calling the RemoveArray() method.
4.1.89  **RemoveMember**

This method deletes the property at the current iterator.

**Syntax**

METHOD RemoveMember : BOOL
VAR_INPUT
  v : SJsonValue;
  i : SJsonIterator;
  keepOrder : BOOL;
END_VAR

**Sample call:**

The following JSON document is to be loaded into the DOM memory:

```javascript
sMessage := '{"serialNumber":"123","batteryVoltage":"1547mV","clickType":"SINGLE","array":["Hello",2,3]}';
```

The "array" property is to be deleted. First of all, the JSON document is searched for the "array" property, after which the property is removed.

```javascript
jsonDoc := fbJson.ParseDocument(sMessage);
jsonIterator := fbJson.MemberBegin(jsonDoc);
jsonIteratorEnd := fbJson.MemberEnd(jsonDoc);
WHILE jsonIterator <> jsonIteratorEnd DO
  sName := fbJson.GetMemberName(jsonIterator);
  IF sName = 'array' THEN
    fbJson.RemoveArray(jsonValue, jsonArrayIterator);
  END_IF
  jsonIterator := fbJson.NextMember(jsonIterator);
END_WHILE
```

4.1.90  **RemoveMemberByName**

This method removes a child element from a given property. The element is referenced by its name.

**Syntax**

METHOD RemoveMemberByName : BOOL
VAR_INPUT
  v : SJsonValue;
  keepOrder : BOOL;
END_VAR
VAR_IN_OUT CONSTANT
  member : STRING;
END_VAR

**Sample call:**

```javascript
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.RemoveMemberByName(jsonProp, 'ChildName');
```

4.1.91  **SaveDocumentToFile**

This method saves a JSON document in a file.
A rising edge at the input parameter bExec triggers the saving procedure. The asynchronous process is terminated as soon as the reference bExec is set back to FALSE from the method. When the process ends, the return value of the method indicates for one call whether saving of the file was successful (TRUE) or failed (FALSE).

**Syntax**

```plaintext
METHOD SaveDocumentToFile : BOOL
VAR_IN_OUT CONSTANT
  sFile : STRING;
END_VAR
VAR_INPUT
  bExec : REFERENCE TO BOOL;
END_VAR
VAR_OUTPUT
  hrErrorCode: HRESULT;
END_VAR

Sample call:

IF bSave THEN
  bSaved := fbJson.SaveDocumentToFile(sFile, bSave);
END_IF
```

### 4.1.92 SetArray

This method sets the value of a property to the type "Array". New values can now be added to the array with the Pushback methods.

**Syntax**

```plaintext
METHOD SetArray : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetArray(jsonProp);
```

### 4.1.93 SetBase64

This method sets the value of a property to a Base64-coded value. A data structure, for example, can be used as the source. Coding to the Base64 format takes place inside the method.

**Syntax**

```plaintext
METHOD SetBase64 : SJsonValue
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR
```

**Sample call:**

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetBase64(jsonProp, ADR(stStruct), SIZEOF(stStruct));
```

### 4.1.94 SetBool

This method sets the value of a property to a value of the data type BOOL.
Syntax

METHOD SetBool : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : BOOL;
END_VAR

Sample call:

```json
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetBool(jsonProp, TRUE);
```

### 4.1.95 SetDateTime

This method sets the value of a property to a value of the data type DATE_AND_TIME.

Syntax

METHOD SetDateTime : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DATE_AND_TIME;
END_VAR

Sample call:

```json
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetDateTime(jsonProp, dtTime);
```

### 4.1.96 SetDcTime

This method sets the value of a property to a value of the data type DCTIME.

Syntax

METHOD SetDcTime : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DCTIME;
END_VAR

Sample call:

```json
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetDcTime(jsonProp, dcTime);
```

### 4.1.97 SetDouble

This method sets the value of a property to a value of the data type Double.

Syntax

METHOD SetDouble : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : LREAL;
END_VAR

Sample call:

```json
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetDouble(jsonProp, 42.42);
```
4.1.98  SetFileTime

This method sets the value of a property to a value of the data type FILETIME.

Syntax

METHOD SetFileTime : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : FILETIME;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetFileTime(jsonProp, fileTime);

4.1.99  SetHexBinary

This method sets the value of a property to a HexBinary-coded value. A data structure, for example, can be used as the source. Coding to the HexBinary format takes place inside the method.

Syntax

METHOD SetHexBinary : SJsonValue
VAR_INPUT
  v : SJsonValue;
  p : PVOID;
  n : DINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetHexBinary(jsonProp, ADR(stStruct), SIZEOF(stStruct));

4.1.100  SetInt

This method sets the value of a property to a value of the data type INT.

Syntax

METHOD SetInt : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : DINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetInt(jsonProp, 42);

4.1.101  SetInt64

This method sets the value of a property to a value of the data type Int64.

Syntax

METHOD SetInt64 : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : LINT;
END_VAR
4.1.102 SetJson

This method inserts a further JSON document into the value of a property.

Syntax

METHOD SetJson : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  rawJson : STRING;
END_VAR

Sample call:

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetJson(jsonProp, sJson);
```

4.1.103 SetMaxDecimalPlaces

This method sets the current setting for MaxDecimalPlaces. This sets the maximum number of decimal places to be used with floating point numbers.

Syntax

METHOD SetMaxDecimalPlaces
VAR_INPUT
  dp : DINT;
END_VAR

Sample call:

```plaintext
nDec := fbJson.SetMaxDecimalPlaces();
```

4.1.104 SetNull

This method sets the value of a property to the value NULL.

Syntax

METHOD SetNull : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR

Sample call:

```plaintext
jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetNull(jsonProp);
```

4.1.105 SetObject

This method sets the value of a property to the type "Object". This enables the nesting of JSON documents.
Function blocks

Syntax

METHOD SetDouble : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : LREAL;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetObject(jsonProp);

4.1.106  SetString

This method sets the value of a property to a value of the data type STRING.

Syntax

METHOD SetString : SJsonValue
VAR_INPUT
  v : SJsonValue;
END_VAR
VAR_IN_OUT CONSTANT
  value : STRING;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetString(jsonProp, 'Hello World');

4.1.107  SetUint

This method sets the value of a property to a value of the data type UInt.

Syntax

METHOD SetUint : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : UDINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetUint(jsonProp, 42);

4.1.108  SetUint64

This method sets the value of a property to a value of the data type UInt64.

Syntax

METHOD SetUint64 : SJsonValue
VAR_INPUT
  v : SJsonValue;
  value : ULINT;
END_VAR

Sample call:

jsonDoc := fbJson.ParseDocument(sExistingJsonDocument);
jsonProp := fbJson.FindMember(jsonDoc, 'property');
jsonValue := fbJson.SetUint64(jsonProp, 42);
4.2 FB_JsonDynDomParser

This function block is derived from the same internal function block as FB_JsonDomParser [19] and thus offers the same interface.

The two derived function blocks differ only in their internal memory management. FB_JsonDynDomParser is optimized for JSON documents to which many changes are made. It releases the allocated memory again after the execution of an action, e.g. for the methods SetObject() or SetJson().

Strings in UTF-8 format

The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication as well as for JSON documents.

In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotBase and Tc3_JsonXml libraries is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

Further information on the UTF-8 STRING format and available display and conversion options can be found in the documentation for the Tc2_Utilities PLC library.

Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4024.7</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml 3.3.8.0</td>
</tr>
</tbody>
</table>

4.3 FB_JsonSaxReader

Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4022</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml</td>
</tr>
</tbody>
</table>

4.3.1 DecodeBase64

This method converts a Base64-formated string to binary data. If the conversion was successful the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD DecodeBase64 : BOOL
VAR_INPUT
sBase64 : STRING;
pBytes : POINTER TO BYTE;
nBytes : REFERENCE TO DINT;
END_VAR
VAR_OUTPUT
hrErrorCode : HRESULT;
END_VAR

Sample call:

bSuccess := fbJson.DecodeBase64('SGVsbG8gVHdpbkNBVA==', ADR(byteArray), byteArraySize);
4.3.2 DecodeDateTime

This method enables the generation of a PLC variable of the type DATE_AND_TIME or DT from a standardized ISO8601 time format (e.g. YYYY-MM-DDThh:mm:ss). DT corresponds to the number of seconds starting from the date 1970-01-01. If the conversion was successful the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD DecodeDateTime : BOOL
VAR_IN_OUT CONSTANT
  sDT : STRING;
END_VAR
VAR_OUTPUT
  nDT : DATE_AND_TIME;
  hrErrorCode : HRESULT;
END_VAR

Sample call:

bSuccess := fbJson.DecodeDateTime('2017-08-09T06:54:00', nDT => dateTime);

4.3.3 DecodeDcTime

This method enables the generation of a PLC variable of the type DCTIME from a standardized ISO8601 time format (e.g. YYYY-MM-DDThh:mm:ss). DCTIME corresponds to the number of nanoseconds starting from the date 2000-01-01. If the conversion was successful the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD DecodeDcTime : BOOL
VAR_IN_OUT CONSTANT
  sDC : STRING;
END_VAR
VAR_OUTPUT
  nDC : DCTIME;
  hrErrorCode : HRESULT;
END_VAR

Sample call:

bSuccess := fbJson.DecodeDcTime('2017-08-09T06:54:00', nDc => dcTime);

4.3.4 DecodeFileTime

This method enables the generation of a PLC variable of the type FILETIME from a standardized ISO8601 time format (e.g. YYYY-MM-DDThh:mm:ss). FILETIME corresponds to the number of nanoseconds starting from the date 1601-01-01 – measured in 100 nanoseconds. If the conversion was successful the method returns TRUE, otherwise it returns FALSE.

Syntax

METHOD DecodeDateTime : BOOL
VAR_IN_OUT CONSTANT
  sFT : STRING;
END_VAR
VAR_OUTPUT
  nFT : FILETIME;
  hrErrorCode : HRESULT;
END_VAR

Sample call:

bSuccess := fbJson.DecodeFileTime('2017-08-09T06:54:00', nFT => fileTime);
4.3.5  DecodeHexBinary

This method converts a string containing hexadecimal values into binary data. If the conversion was successful the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD DecodeHexBinary : BOOL
VAR_IN_OUT CONSTANT
  sHex : STRING;
END_VAR
VAR_INPUT
  pBytes : POINTER TO BYTE;
  nBytes : REFERENCE TO DINT;
END_VAR
VAR_OUTPUT
  hrErrorCode : HRESULT;
END_VAR

Sample call:

bSuccess := fbJson.DecodeHexBinary('ABCEF93A', ADR(byteArray), byteArraySize);

4.3.6  IsBase64

This method checks whether the transferred string corresponds to the Base64 format. If that is the case, the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD IsBase64 : BOOL
VAR_IN_OUT CONSTANT
  sBase64 : STRING;
END_VAR

Sample call:

bIsBase64 := fbJson.IsBase64('SGVsbG8gVHdpbkNBVA==');

4.3.7  IsHexBinary

This method checks whether the transferred string consists of hexadecimal values. If that is the case, the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD IsHexBinary : BOOL
VAR_IN_OUT CONSTANT
  sHex : STRING;
END_VAR

Sample call:

bSuccess := fbJson.IsHexBinary('ABCEF93A');

4.3.8  IsISO8601TimeFormat

This method checks whether the transferred string corresponds to the standardized ISO8601 time format. If that is the case, the method returns TRUE, otherwise it returns FALSE.

**Syntax**

METHOD IsISO8601TimeFormat : BOOL
VAR_IN_OUT CONSTANT
  sDT : STRING;
END_VAR
Sample call:

\[
bSuccess := fbJson.IsISO8601TimeFormat('2017-08-09T06:54:00');
\]

### 4.3.9 Parse

This method starts the SAX reader parsing procedure. The JSON object to be parsed and a reference to a function block, which was derived from the interface ITcJsonSaxHandler, are transferred as input parameters. This function block is then used for the callback methods of the SAX reader.

**Syntax**

```plaintext
METHOD Parse : BOOL
VAR_IN_OUT CONSTANT
  sJson : STRING;
END_VAR
VAR_INPUT
  ipHdl : ITcJsonSaxHandler;
END_VAR
VAR_OUTPUT
  hrErrorCode : HRESULT;
END_VAR
```

### 4.3.10 ParseValues

This method starts the SAX reader parsing procedure. The JSON object to be parsed and a reference to a function block, which was derived from the interface ITcJsonSaxValues, are transferred as input parameters. This function block is then used for the callback methods of the SAX reader. What is special about this method is that exclusively values are taken into account in the callback methods, i.e. there are no OnKey() or OnStartObject() callbacks.

**Syntax**

```plaintext
METHOD ParseValues : BOOL
VAR_IN_OUT CONSTANT
  sJson : STRING;
END_VAR
VAR_INPUT
  ipHdl : ITcJsonSaxValues;
END_VAR
VAR_OUTPUT
  hrErrorCode : HRESULT;
END_VAR
```

### 4.4 FB_JsaxWriter

- **Strings in UTF-8 format**

  The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication as well as for JSON documents.

  In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotBase and Tc3_JsonXml libraries is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

  If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

  Further information on the UTF-8 STRING format and available display and conversion options can be found in the documentation for the Tc2_Utilities PLC library.
Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4022</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml</td>
</tr>
</tbody>
</table>

### 4.4.1 AddBase64

This method adds a value of the data type Base64 to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.

**Syntax**

```plaintext
METHOD AddBase64
VAR_INPUT
    pBytes : Pointer TO BYTE;
    nBytes : DINT;
END_VAR
```

### 4.4.2 AddBool

This method adds a value of the data type BOOL to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.

**Syntax**

```plaintext
METHOD AddBool
VAR_INPUT
    value : BOOL;
END_VAR
```

**Sample call:**

```plaintext
fbJson.AddKey('bSwitch');
fJson.AddBool(TRUE);
```

### 4.4.3 AddDateTime

This method adds a value of the data type DATE_AND_TIME to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.

**Syntax**

```plaintext
METHOD AddDateTime
VAR_INPUT
    value : DATE_AND_TIME;
END_VAR
```

**Sample call:**

```plaintext
fbJson.AddKey('Timestamp');
fJson.AddDateTime(dtTime); // dtTime is of type DATE_AND_TIME
```

### 4.4.4 AddDcTime

This method adds a value of the data type DCTIME to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.

**Syntax**

```plaintext
METHOD AddDcTime
VAR_INPUT
    value : DCTIME;
END_VAR
```
Sample call:

```plaintext
fbJson.AddKey('Timestamp');
fbJson.AddDcTime(dcTime); // dcTime is of type DCTIME
```

### 4.4.5 AddDint

This method adds a value of the data type DINT to a property. Usually, a corresponding property was created beforehand with the method `AddKey()` [54].

**Syntax**

```plaintext
METHOD AddDint
VAR_INPUT
    value : DINT;
END_VAR
```

Sample call:

```plaintext
fbJson.AddKey('nNumber');
fbJson.AddDint(42);
```

### 4.4.6 AddFileTime

This method adds a value of the data type FILETIME to a property. Usually, a corresponding property was created beforehand with the method `AddKey()` [54].

**Syntax**

```plaintext
METHOD AddFileTime
VAR_INPUT
    value : FILETIME;
END_VAR
```

Sample call:

```plaintext
fbJson.AddKey('Timestamp');
fbJson.AddFileTime(ftTime); // ftTime is of type FILETIME
```

### 4.4.7 AddHexBinary

This method adds a hex binary value to a property. Usually, a corresponding property was created beforehand with the method `AddKey()` [54].

**Syntax**

```plaintext
METHOD AddHexBinary
VAR_INPUT
    pBytes : POINTER TO BYTE;
    nBytes : DINT;
END_VAR
```

Sample call:

```plaintext
fbJson.AddKey('HexBinary');
fbJson.AddHexBinary(ADR(byteHexBin), SIZEOF(byteHexBin));
```

### 4.4.8 AddKey

This method adds a new property key at the current position of the SAX writer. The value of the new property is usually set afterwards. This can be done using one of the following methods, for example: `AddBase64` [53], `AddBool` [53], `AddDateTime` [53], `AddDcTime` [53], `AddDint` [54], `AddFileTime` [54], `AddHexBinary` [54], `AddLInt` [57], `AddLreal` [57], `AddNull` [57], `AddRawArray` [58], `AddRawObject` [58], `AddReal` [58], `AddString` [58], `AddUdint` [59], `AddUlint` [59].
Syntax
METHOD AddKey
VAR_IN_OUT CONSTANT
  key : STRING;
END_VAR

Sample call:
fbJson.AddKey('PropertyName');

### 4.4.9 AddKeyBool
This method creates a new property key and at the same time a value of the data type BOOL.

Syntax
METHOD AddKeyBool
VAR_IN_OUT CONSTANT
  key : STRING;
END_VAR
VAR_INPUT
  value : BOOL;
END_VAR

Sample call:
fbJson.AddKeyBool('bSwitch', TRUE);

### 4.4.10 AddKeyDateTime
This method creates a new property key and at the same time a value of the data type DATE_AND_TIME.

Syntax
METHOD AddKeyDateTime
VAR_IN_OUT CONSTANT
  key : STRING;
END_VAR
VAR_INPUT
  value : DATE_AND_TIME;
END_VAR

Sample call:
fbJson.AddKeyDateTime('Timestamp', dtTime);

### 4.4.11 AddKeyDcTime
This method creates a new property key and at the same time a value of the data type DCTIME.

Syntax
METHOD AddKeyDcTime
VAR_IN_OUT CONSTANT
  key : STRING;
END_VAR
VAR_INPUT
  value : DCTIME;
END_VAR

Sample call:
fbJson.AddKeyDcTime('Timestamp', dcTime);

### 4.4.12 AddKeyFileTime
This method creates a new property key and at the same time a value of the data type FILETIME.
4.4.13  AddKeyFileTime

This method creates a new property key and at the same time a value of the data type FILETIME.

Syntax

METHOD AddKeyFileTime
VAR_IN_OUT CONSTANT
    key : STRING;
END_VAR
VAR_INPUT
    value : FILETIME;
END_VAR

Sample call:

fbJson.AddKeyFileTime('Timestamp', ftTime);

4.4.13  AddKeyLreal

This method creates a new property key and at the same time a value of the data type LREAL.

Syntax

METHOD AddKeyLreal
VAR_IN_OUT CONSTANT
    key : STRING;
END_VAR
VAR_INPUT
    value : LREAL;
END_VAR

Sample call:

fbJson.AddKeyLreal('PropertyName', 42.42);

4.4.14  AddKeyNull

This method creates a new property key and initializes its value with zero.

Syntax

METHOD AddKeyNull
VAR_IN_OUT CONSTANT
    key : STRING;
END_VAR

Sample call:

fbJson.AddKeyNull('PropertyName');

4.4.15  AddKeyNumber

This method creates a new property key and at the same time a value of the data type DINT.

Syntax

METHOD AddKeyNumber
VAR_IN_OUT CONSTANT
    key : STRING;
END_VAR
VAR_INPUT
    value : DINT;
END_VAR

Sample call:

fbJson.AddKeyNumber('PropertyName', 42);

4.4.16  AddKeyString

This method creates a new property key and at the same time a value of the data type STRING.
### Syntax

**METHOD AddKeyString**

VAR IN OUT CONSTANT
   key : STRING;
   value : STRING;
END_VAR

**Sample call:**

```java
fbJson.AddKeyString('PropertyName', 'Hello World');
```

### 4.4.17 AddLint

This method adds a value of the data type LINT to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.[54]

**Syntax**

**METHOD AddLint**

VAR INPUT
   value : LINT;
END_VAR

**Sample call:**

```java
fbJson.AddKey('PropertyName');
fbJson.AddLint(42);
```

### 4.4.18 AddLreal

This method adds a value of the data type LREAL to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.[54]

**Syntax**

**METHOD AddLreal**

VAR_INPUT
   value : LREAL;
END_VAR

**Sample call:**

```java
fbJson.AddKey('PropertyName');
fbJson.AddLreal(42.42);
```

### 4.4.19 AddNull

This method adds the value zero to a property. Usually, a corresponding property was created beforehand with the method `AddKey()`.[54]

**Syntax**

**METHOD AddNull**

**Sample call:**

```java
fbJson.AddKey('PropertyName');
fbJson.AddNull();
```
4.4.20 AddRawArray

This method adds a valid JSON array to a given property as a value. The array to be added must be in a valid JSON format and may only be added if the SAX writer is at a correspondingly valid position, i.e. for example, directly after a preceding AddKey() [54], StartArray() [61] or as the first call after a ResetDocument() [61].

Syntax

METHOD AddRawArray
VAR_IN_OUT CONSTANT
    rawJson : STRING;
END_VAR

Sample call:

fbJson.AddKey('PropertyName');
fbJson.AddRawArray('[1, 2, {"x":42, "y":42}, 4');

4.4.21 AddRawObject

This method adds a valid JSON object to a given property as a value. The object to be added must be in a valid JSON format and may only be added if the SAX writer is at a correspondingly valid position, i.e. for example, directly after a preceding AddKey() [54], StartArray() [61] or as the first call after a ResetDocument() [61].

Syntax

METHOD AddRawObject
VAR_IN_OUT CONSTANT
    rawJson : STRING;
END_VAR

Sample call:

fbJson.AddKey('PropertyName');
fbJson.AddRawObject('{"x":42, "y":42}');

4.4.22 AddReal

This method adds a value of the data type REAL to a property. Usually, a corresponding property was created beforehand with the method AddKey() [54].

Syntax

METHOD AddReal
VAR_INPUT
    value : REAL;
END_VAR

Sample call:

fbJson.AddKey('PropertyName');
fbJson.AddReal(42.42);

4.4.23 AddString

This method adds a value of the data type STRING to a property. Usually, a corresponding property was created beforehand with the method AddKey() [54].
4.4.24 AddUdint

This method adds a value of the data type UDINT to a property. Usually, a corresponding property was created beforehand with the method AddKey() ▶ 54.

Syntax

METHOD AddUdint
VAR_INPUT
  value : UDINT;
END_VAR

Sample call:

fbJson.AddKey('PropertyName');
fbJson.AddUdint(42);

4.4.25 AddUlint

This method adds a value of the data type ULINT to a property. Usually, a corresponding property was created beforehand with the method AddKey() ▶ 54.

Syntax

METHOD AddUlint
VAR_INPUT
  value : ULINT;
END_VAR

Sample call:

fbJson.AddKey('PropertyName');
fbJson.AddUlint(42);

4.4.26 CopyDocument

This method copies the content of the current JSON object created with the SAX Writer to a target variable of the data type STRING, which can be of any length. The method returns the length of the string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

Syntax

METHOD CopyDocument : UDINT
VAR_IN_OUT CONSTANT
  pDoc : STRING;
END_VAR
VAR_INPUT
  nDoc : UDINT;
END_VAR
VAR_OUTPUT
  hrErrorCode: HRESULT;
END_VAR

Sample call:

fbJson.CopyDocument(sTargetString, SIZEOF(sTargetString));
4.4.27 EndArray
This method generates the end of a started JSON array ("square closing bracket") and inserts it at the
current position of the SAX writer.

Syntax
METHOD EndArray : HRESULT

Sample call:
fbJson.EndArray();

4.4.28 EndObject
This method generates the end of a started JSON object ("curly closing bracket") and inserts it at the current
position of the SAX writer.

Syntax
METHOD EndObject : HRESULT

Sample call:
fbJson.EndObject();

4.4.29 GetDocument
This method returns the content of the JSON object that is currently created with the SAX Writer and returns
it as data type STRING(255).

The maximum size of the string returned by the method is 255 characters. With longer strings, the method
will return a NULL string. In this case the method CopyDocument()[59]() must be used.

Syntax
METHOD GetDocument : STRING(255)
VAR_OUTPUT
    hrErrorCode: HRESULT;
END_VAR

Sample call:
sTargetString := fbJson.GetDocument();

4.4.30 GetDocumentLength
This method returns the length of the JSON object that is currently created with the SAX Writer and returns it
as data type UDINT.

Syntax
METHOD GetDocumentLength : UDINT
VAR_OUTPUT
    hrErrorCode: HRESULT;
END_VAR

Sample call:
nLength := fbJson.GetDocumentLength();
4.4.31 GetMaxDecimalPlaces

Syntax
METHOD GetMaxDecimalPlaces : DINT

4.4.32 ResetDocument

This method resets the JSON object currently created with the SAX writer.

Syntax
METHOD ResetDocument : HRESULT

Sample call:
fbJson.ResetDocument();

4.4.33 SetMaxDecimalPlaces

Syntax
METHOD SetMaxDecimalPlaces : HRESULT
VAR_INPUT
decimalPlaces: DINT;
END_VAR

4.4.34 StartArray

This method generates the start of a new JSON array ("square opening bracket") and inserts it at the current position of the SAX writer.

Syntax
METHOD StartArray : HRESULT

Sample call:
fbJson.StartArray();

4.4.35 StartObject

This method generates the start of a new JSON object ("curly opening bracket") and inserts it at the current position of the SAX writer.

Syntax
METHOD StartObject : HRESULT

Sample call:
fbJson.StartObject();

4.5 FB_JsonReadWriteDataType

In order to use UTF-8 characters, e.g. in the automatic generation of metadata via the function block FB_JsonReadWriteDataType [p. 61], the check box for the support of UTF-8 in the symbolism must be activated in the TwinCAT project. To do this, double-click on SYSTEM in the project tree, open the Settings tab and activate the corresponding check box.
Strings in UTF-8 format

The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication as well as for JSON documents.

In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotBase and Tc3_JsonXml libraries is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

Further information on the UTF-8 STRING format and available display and conversion options can be found in the documentation for the Tc2_Utilities PLC library.

Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4022</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml</td>
</tr>
</tbody>
</table>

4.5.1 AddJsonKeyPropertiesFromSymbol

With the aid of this method, metadata can be added via PLC attributes to the JSON representation of a PLC data structure on an FB_JsonSaxWriter object. The method receives as its input parameters the instance of the FB_JsonSaxWriter function block, the desired name of the JSON property that is to contain the metadata, the data type name of the structure and a string variable sProperties, which contains a list of the PLC attributes to be extracted, separated by a cross bar.

Syntax

METHOD AddJsonValueFromSymbol : BOOL
VAR_IN_OUT
  fbWriter : FB_JsonSaxWriter;
END_VAR
VAR_IN_OUT_CONSTANT
  sKey : STRING;
  sDatatype : STRING;
  sProperties : STRING;
END_VAR
The PLC attributes can be specified in the following form on the structure elements:

```plaintext
{attribute 'Unit' := 'm/s'}
{attribute 'DisplayName' := 'Speed'}
```

Sensor1 : REAL;

A complete sample of how to use this method can be found in section Tc3JsonXmlSampleJsonDataType |

Sample call:

```plaintext
fbJsonSaxWriter.ResetDocument()
fbJsonDataType.AddJsonKeyPropertiesFromSymbol(fbJsonSaxWriter, 'MetaData','ST_Values','Unit| DisplayName');
```

### 4.5.2 AddJsonValueFromSymbol

This method generates the JSON representation of a PLC data structure on an `FB_JsonSaxWriter` object. The method receives as its input parameters the instance of the FB_JsonSaxWriter function block, the data type name of the structure, and the address and size of the source structure instance. As a result, the FB_JsonSaxWriter instance contains a valid JSON representation of the structure. Unlike the method `AddJsonValueFromSymbol()`, the elements of the source structure are nested here in a JSON sub-object whose name can be specified via the input/output parameter `sKey`.

**Syntax**

```plaintext
METHOD AddJsonValueFromSymbol : BOOL
  VAR_IN_OUT
  fbWriter : FB_JsonSaxWriter;
  END_VAR
  VAR_IN_OUT CONSTANT
  sKey : STRING;
  sDatatype : STRING;
  END_VAR
  VAR_INPUT
  nData : UDINT;
  pData : PVOID;
  END_VAR
  VAR_OUTPUT
  hrErrorCode : HRESULT;
  END_VAR
```

A complete sample of how to use this method can be found in section Tc3JsonXmlSampleJsonDataType |

Sample call:

```plaintext
fbJsonSaxWriter.ResetDocument()
fbJsonDataType.AddJsonKeyValueFromSymbol(fbJsonSaxWriter, 'Values','ST_Values',SIZEOF(stValues), ADR(stValues));
```

### 4.5.3 AddJsonValueFromSymbol

This method generates the JSON representation of a PLC data structure on an `FB_JsonSaxWriter` object. The method receives as its input parameters the instance of the FB_JsonSaxWriter function block, the data type name of the structure, and the address and size of the source structure instance. As a result, the FB_JsonSaxWriter instance contains a valid JSON representation of the structure.

**Syntax**

```plaintext
METHOD AddJsonValueFromSymbol : BOOL
  VAR_IN_OUT
  fbWriter : FB_JsonSaxWriter;
  END_VAR
  VAR_IN_OUT CONSTANT
  sKey : STRING;
  sDatatype : STRING;
  END_VAR
  VAR_INPUT
  nData : UDINT;
  pData : PVOID;
  END_VAR
  VAR_OUTPUT
  hrErrorCode : HRESULT;
  END_VAR
```

A complete sample of how to use this method can be found in section Tc3JsonXmlSampleJsonDataType |

Sample call:

```plaintext
fbJsonSaxWriter.ResetDocument()
fbJsonDataType.AddJsonKeyValueFromSymbol(fbJsonSaxWriter, 'Values','ST_Values',SIZEOF(stValues), ADR(stValues));
```
### 4.5.4 CopyJsonStringFromSymbol

This method generates the JSON representation of a symbol and copies it into a variable of the data type STRING, which can be of any length. The method returns the length of the string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

```plaintext
METHOD CopyJsonStringFromSymbol : UDINT
VAR_INPUT
    nData : UDINT;
    nDoc : UDINT;
    pData : PVOID;
END_VAR
VAR_IN_OUT CONSTANT
    pDoc : STRING;
    sDatatype : STRING;
END_VAR
VAR_OUTPUT
    hrErrorCode : HRESULT;
END_VAR
```

**Sample call:**

```plaintext
nLen := fbJsonDataType.CopyJsonStringFromSymbol('ST_Test', SIZEOF(stTest), ADR(stTest), sString, SIZEOF(sString));
```

### 4.5.5 CopyJsonStringFromSymbolProperties

This method generates a corresponding JSON representation of PLC attributes on a symbol. In contrast to the `AddJsonKeyPropertiesFromSymbol` method, the result is not written to an instance of the function block FB_JsnaWriter, but to a string variable. The method receives as its input parameters the data type name of the symbol and a string variable that represents a list of the PLC attributes to be extracted, separated by a cross bar.

The method copies this JSON representation into a variable of the data type STRING, which can be of any length. The method returns the length of the string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

```plaintext
METHOD CopyJsonStringFromSymbolProperties : UDINT
VAR_INPUT
    nDoc : UDINT;
END_VAR
VAR_IN_OUT CONSTANT
    pDoc : STRING;
    sDatatype : STRING;
    sProperties : STRING;
END_VAR
```
4.5.6 **CopySymbolNameByAddress**

This method returns the complete (ADS) symbol name of a transferred symbol. The method returns the size of the string (including null termination). If the target buffer is too small, it is emptied by a null termination and returned as length 0.

**Syntax**

METHOD CopySymbolNameByAddress : UDINT
VAR_INPUT
   nData : UDINT; // size of symbol
   pData : PVOID; // address of symbol
VAR_IN_OUT CONSTANT
   sName : STRING; // target string buffer where the symbol name should be copied to
VAR_INPUT
   nName : UDINT; // size in bytes of target string buffer
END_VAR
VAR_OUTPUT
   hrErrorCode : HRESULT;
END_VAR

**Sample call:**

```
nSymbolSize := fbJsonDataType.CopySymbolNameByAddress(nData:=SIZEOF(stValues), pData:=ADR(stValues), sName:=sSymbolName, nName:=SIZEOF(sSymbolName));
```

**Requirements**

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4024.20</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml 3.3.15.0</td>
</tr>
</tbody>
</table>

4.5.7 **GetDataTypeNameByAddress**

This method returns the data type name of a transferred symbol.

**Syntax**

METHOD GetDataTypeNameByAddress : STRING
VAR_INPUT
   nData : UDINT;
   pData : PVOID;
VAR_OUTPUT
   hrErrorCode : HRESULT;
END_VAR

**Sample call:**

```
sBuffer := fbJsonDataType.GetDataTypeNameByAddress(SIZEOF(stValues),ADR(stValues));
```

4.5.8 **GetJsonFromSymbol**

This method generates the corresponding JSON representation of a symbol. In contrast to the AddJsonValueFromSymbol() method, the result is not written to an instance of the function block FB_JSonSaxWriter, but to a string variable. The method receives as its input parameters the data type name.
of the symbol as well as the address and size of the source symbol, e.g. of a structure instance. The address and size of the destination buffer that contains the JSON representation of the symbol after the call are transferred as further input parameters.

Syntax

METHOD GetJsonFromSymbol : BOOL
VAR_IN_OUT CONSTANT
sDatatype : STRING;
END_VAR
VAR_INPUT
nData : UDINT;
pData : PVOID;
nJson : REFERENCE TO UDINT;
pJson : POINTER TO STRING;
END_VAR
VAR_OUTPUT
hrErrorCode : HRESULT;
END_VAR

4.5.9 GetJsonStringFromSymbol

This method generates the corresponding JSON representation of a symbol. In contrast to the AddJsonValueFromSymbol() method, the result is not written to an instance of the function block FB_JsonSaxWriter, but to a string variable. The method receives as its input parameters the data type name of the symbol as well as the address and size of the source symbol, e.g., of a structure instance.

The maximum size of the string returned by the method is 255 characters. With longer strings, the method will return a NULL string. In this case the method CopyJsonStringFromSymbol() must be used.

Syntax

METHOD GetJsonStringFromSymbol : STRING(255)
VAR_IN_OUT CONSTANT
sDatatype : STRING;
END_VAR
VAR_INPUT
nData : UDINT;
pData : PVOID;
END_VAR
VAR_OUTPUT
hrErrorCode : HRESULT;
END_VAR

Sample call:

sBuffer := fbJsonDataType.GetJsonStringFromSymbol('ST_Values', SIZEOF(stValues), ADR(stValues), nBufferLength, ADR(sBuffer));

4.5.10 GetJsonStringFromSymbolProperties

This method generates a corresponding JSON representation of PLC attributes on a symbol. In contrast to the AddJsonKeyPropertiesFromSymbol() method, the result is not written to an instance of the function block FB_JsonSaxWriter, but to a string variable. The method receives as its input parameters the data type name of the symbol and a string variable that represents a list of the PLC attributes to be extracted, separated by a cross bar. The result is returned directly as the return value of the method.

Sample call:

sBuffer := fbJsonDataType.GetJsonStringFromSymbolProperties('ST_Values', 'ST_Values', ADR(stValues));
The maximum size of the string returned by the method is 255 characters. With longer strings, the method will return a NULL string. In this case the method CopyJsonStringFromSymbolProperties must be used.

Syntax

METHOD GetJsonStringFromSymbolProperties : STRING(255)
VAR_IN_OUT CONSTANT
 sDatatype : STRING;
 sProperties : STRING;
END_VAR
VAR_OUTPUT
 hrErrorCode : HRESULT;
END_VAR

Sample call:

sBuffer := fbJsonDataType.GetJsonStringFromSymbolProperties('ST_Values', 'Unit|DisplayName');

4.5.11 GetSizeJsonStringFromSymbol

This method reads the size of the JSON representation of a symbol. The value is specified with null termination.

Syntax

METHOD GetSizeJsonStringFromSymbol : UDINT
VAR_INPUT
 nData :UDINT;
 pData : PVOID;
END_VAR
VAR_IN_OUT CONSTANT
 sDatatype : STRING;
END_VAR
VAR_OUTPUT
 hrErrorCode : HRESULT;
END_VAR

Sample call:

nLen := fbJsonDataType.GetSizeJsonStringFromSymbol('BOOL',SIZEOF(bBool),ADR(bBool));

4.5.12 GetSizeJsonStringFromSymbolProperties

This method reads the size of the JSON representation of PLC attributes on a symbol. The value is specified with null termination.

Syntax

METHOD GetSizeJsonStringFromSymbolProperties : UDINT
VAR_IN_OUT CONSTANT
 sDatatype : STRING;
 sProperties : STRING;
END_VAR
VAR_OUTPUT
 hrErrorCode : HRESULT;
END_VAR

Sample call:

nLen := fbJsonDataType.GetSizeJsonStringFromSymbolProperties('ST_Test','DisplayName|Unit');

4.5.13 GetSymbolNameByAddress

This method returns the complete (ADS) symbol name of a transferred symbol.

The maximum size of the string returned by the method is 255 characters. With longer strings, the method will return a null string. In this case the method CopySymbolNameByAddress must be used.
### Syntax

```plaintext
METHOD GetSymbolNameByAddress : STRING(255)
VAR_INPUT
  nData : UDINT;
  pData : PVOID;
END_VAR
VAR_OUTPUT
  hrErrorCode : HRESULT;
END_VAR

Sample call:

sBuffer := fbJsonDataType.GetSymbolNameByAddress(SIZEOF(stValues), ADR(stValues));
```

#### 4.5.14 SetSymbolFromJson

This method extracts a string containing a valid JSON message and attempts to save the contents of the JSON object to an equivalent data structure. The method receives as its input parameters the string with the JSON object, the data type name of the target structure, and the address and size of the target structure instance.

```plaintext
METHOD SetSymbolFromJson : BOOL
VAR_IN_OUT CONSTANT
  sJson : STRING;
  sDatatype : STRING;
END_VAR
VAR_INPUT
  nData : UDINT;
  pData : PVOID;
END_VAR
VAR_OUTPUT
  hrErrorCode : HRESULT;
END_VAR

Sample call:

fbJsonDataType.SetSymbolFromJson(sJson,'ST_Values',SIZEOF(stValuesReceive), ADR(stValuesReceive));
```

### 4.6 FB_XmlDomParser

#### Strings in UTF-8 format

The variables of type STRING used here are based on the UTF-8 format. This STRING formatting is common for MQTT communication as well as for JSON documents.

In order to be able to receive special characters and texts from a wide range of languages, the character set in the Tc3_IotBase and Tc3_JsonXml libraries is not limited to the typical character set of the data type STRING. Instead, the Unicode character set in UTF-8 format is used in conjunction with the data type STRING.

If the ASCII character set is used, there is no difference between the typical formatting of a STRING and the UTF-8 formatting of a STRING.

Further information on the UTF-8 STRING format and available display and conversion options can be found in the documentation for the Tc2_Utilities PLC library.

### Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4022</td>
<td>x86, x64, ARM</td>
<td>Tc3_JsonXml</td>
</tr>
</tbody>
</table>
4.6.1  AppendAttribute

This method adds a new attribute to an existing node. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttribute : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
  value : STRING;
END_VAR

Sample call:
objAttribute := fbXml.AppendAttribute(objMachine, 'Name', 'some value');

4.6.2  AppendAttributeAsBool

This method adds a new attribute to an existing node. The value of the attribute has the data type Boolean. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttributeAsBool : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : BOOL;
END_VAR

Sample call:
objAttribute := fbXml.AppendAttributeAsBool(objMachine, 'Name', TRUE);

4.6.3  AppendAttributeAsDouble

This method adds a new attribute to an existing node. The value of the attribute has the data type Double. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttributeAsDouble : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : LREAL;
END_VAR

Sample call:
objAttribute := fbXml.AppendAttributeAsDouble(objMachine, 'Name', 42.42);
4.6.4 AppendAttributeAsFloat

This method adds a new attribute to an existing node. The value of the attribute has the data type Float. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttributeAsFloat : SXmlAttribute
VAR_INPUT
   n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
   name : STRING;
END_VAR
VAR_INPUT
   value : REAL;
END_VAR
Sample call:

objAttribute := fbXml.AppendAttributeAsFloat(objMachine, 'Name', 42.42);

4.6.5 AppendAttributeAsInt

This method adds a new attribute to an existing node. The value of the attribute has the data type Integer. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttributeAsInt : SXmlAttribute
VAR_INPUT
   n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
   name : STRING;
END_VAR
VAR_INPUT
   value : DINT;
END_VAR
Sample call:

objAttribute := fbXml.AppendAttributeAsInt(objMachine, 'Name', 42);

4.6.6 AppendAttributeAsLint

This method adds a new attribute to an existing node. The value of the attribute has the data type Integer64. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax

METHOD AppendAttributeAsLint : SXmlAttribute
VAR_INPUT
   n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
   name : STRING;
END_VAR
VAR_INPUT
   value : LINT;
END_VAR
Sample call:

objAttribute := fbXml.AppendAttributeAsLint(objMachine, 'Name', 42);
4.6.7  AppendAttributeAsUint

This method adds a new attribute to an existing node. The value of the attribute has the data type Unsigned Integer. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax
METHOD AppendAttributeAsUint : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : UDINT;
END_VAR
Sample call:
objAttribute := fbXml.AppendAttributeAsUint(objMachine, 'Name', 42);

4.6.8  AppendAttributeAsUlint

This method adds a new attribute to an existing node. The value of the attribute has the data type Unsigned Integer64. The name and value of the new attribute and the existing XML node are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

Syntax
METHOD AppendAttributeAsUlint : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : ULINT;
END_VAR
Sample call:
objAttribute := fbXml.AppendAttributeAsUlint(objMachine, 'Name', 42);

4.6.9  AppendAttributeCopy

This method adds a new attribute to an existing node. The name and value of the new attribute are copied from an existing attribute. The existing attribute is transferred to the method as input parameter.

Syntax
METHOD AppendAttributeCopy : SXmlAttribute
INPUT_VAR
  n : SXmlNode;
  copy : SXmlAttribute;
END_VAR
Sample call:
xmlNewAttribute := fbXml.AppendAttributeCopy(xmlNode, xmlExistingAttribute);
4.6.10  AppendChild

This method inserts a new node below an existing node. The value of the new node has the data type STRING. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node. The input parameter cdata indicates whether the value of the node is to be encapsulated in a CDATA function block, so that certain special characters such as "<" and ">" are allowed as values.

Syntax

METHOD AppendChild : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
  value : STRING;
END_VAR
VAR_INPUT
  cdata : BOOL;
END_VAR

Sample call:

xmlNewNode := fbXml.AppendChild(xmlExisting, 'Controller', 'CX5120', FALSE);

4.6.11  AppendChildAsBool

This method inserts a new node below an existing node. The value of the new node has the data type Boolean. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendChildAsBool : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : BOOL;
END_VAR

Sample call:

xmlNewNode := fbXml.AppendChildAsBool(xmlExisting, 'SomeName', TRUE);

4.6.12  AppendChildAsDouble

This method inserts a new node below an existing node. The value of the new node has the data type Double. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendChildAsDouble : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : LREAL;
END_VAR

Sample call:
4.6.13 AppendChildAsFloat

This method inserts a new node below an existing node. The value of the new node has the data type Float. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendChildAsFloat : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : REAL;
END_VAR

Sample call:

xmlNewNode := fbXml.AppendChildAsFloat(xmlExisting, 'SomeName', 42.42);

4.6.14 AppendChildAsInt

This method inserts a new node below an existing node. The value of the new node has the data type Integer. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendChildAsInt : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : DINT;
END_VAR

Sample call:

xmlNewNode := fbXml.AppendChildAsInt(xmlExisting, 'SomeName', 42);

4.6.15 AppendChildAsLint

This method inserts a new node below an existing node. The value of the new node has the data type Integer64. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendChildAsLint : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : LINT;
END_VAR

Sample call:
xmlNewNode := fbXml.AppendChildAsLint(xmlExisting, 'SomeName', 42);

### 4.6.16 AppendChildAsUint

This method inserts a new node below an existing node. The value of the new node has the data type Unsigned Integer. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

**Syntax**

METHOD AppendChildAsUint : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : UDINT;
END_VAR
Sample call:
xmlNewNode := fbXml.AppendChildAsUint(xmlExisting, 'SomeName', 42);

### 4.6.17 AppendChildAsUlint

This method inserts a new node below an existing node. The value of the new node has the data type Unsigned Integer64. The name and value of the new node and a reference to the existing node are transferred to the method as input parameters. The method returns a reference to the newly added node.

**Syntax**

METHOD AppendChildAsUlint : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR
VAR_INPUT
  value : ULINT;
END_VAR
Sample call:
xmlNewNode := fbXml.AppendChildAsUlint(xmlExisting, 'SomeName', 42);

### 4.6.18 AppendCopy

This method inserts a new node below an existing node. The name and value of the new node are copied from an existing node. The references to the existing nodes are transferred to the method as input parameters. The method returns a reference to the newly added node.

**Syntax**

METHOD AppendCopy : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_INPUT
  copy : SXmlNode;
END_VAR
Sample call:
xmlNewNode := fbXml.AppendCopy(xmlParentNode, xmlExistingNode);
4.6.19  AppendNode

This method adds a new node to an existing node. The existing node and the name of the new node are transferred to the method as input parameters. The method returns a reference to the newly added node.

Syntax

METHOD AppendNode : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR

Sample call:
objMachines := fbXml.AppendNode(objRoot, 'Machines');

4.6.20  Attributes

This method can be used to read the attribute of a given XML node. The XML node and the name of the attribute are transferred to the method as input parameters. After the method has been called, further methods have to be called, for example to read the value of the attribute, e.g. AttributeAsInt().

Syntax

METHOD Attribute : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR

Sample call:
xmlMachine1Attribute := fbXml.Attribute(xmlMachine1, 'Type');

4.6.21  AttributeAsBool

This method returns the value of an attribute as data type Boolean. The attribute is transferred to the method as input parameter.

Syntax

METHOD AttributeAsBool : BOOL
VAR_INPUT
  a : SXmlAttribute;
END_VAR

Sample call:
bValue := fbXml.AttributeAsBool(xmlAttr);

4.6.22  AttributeAsDouble

This method returns the value of an attribute as data type Double. The attribute is transferred to the method as input parameter.

Syntax

METHOD AttributeAsDouble : LREAL
VAR_INPUT
  a : SXmlAttribute;
END_VAR
Sample call:
`lrValue := fbXml.AttributeAsDouble(xmlAttr);`

4.6.23 **AttributeAsFloat**

This method returns the value of an attribute as data type Float. The attribute is transferred to the method as input parameter.

**Syntax**

```
METHOD AttributeAsFloat : REAL
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

Sample call:
`rValue := fbXml.AttributeAsFloat(xmlAttr);`

4.6.24 **AttributeAsInt**

This method returns the value of an attribute as a data type Integer. The attribute is transferred to the method as input parameter.

**Syntax**

```
METHOD AttributeAsInt : DINT
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

Sample call:
`nValue := fbXml.AttributeAsInt(xmlAttr);`

4.6.25 **AttributeAsLint**

This method returns the value of an attribute as a data type Integer64. The attribute is transferred to the method as input parameter.

**Syntax**

```
METHOD AttributeAsLint : LINT
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

Sample call:
`nValue := fbXml.AttributeAsLint(xmlAttr);`

4.6.26 **AttributeAsUint**

This method returns the value of an attribute as data type Unsigned Integer. The attribute is transferred to the method as input parameter.

**Syntax**

```
METHOD AttributeAsUint : UDINT
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

Sample call:
nValue := fbXml.AttributeAsUint(xmlAttr);

### 4.6.27 AttributeAsUlint

This method returns the value of an attribute as data type Unsigned Integer64. The attribute is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD AttributeAsUlint : ULINT
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

**Sample call:**

```plaintext
nValue := fbXml.AttributeAsUlint(xmlAttr);
```

### 4.6.28 AttributeBegin

This method returns an iterator over all attributes of an XML node. The XML node is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD AttributeBegin : SXmlIterator
VAR_INPUT
  n : SXmlNode;
END_VAR
```

**Sample call:**

```plaintext
xmlIterator := fbXml.AttributeBegin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlAttr := fbXml.AttributeFromIterator(xmlIterator);
  nAttrValue := fbXml.AttributeAsInt(xmlAttr);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
```

### 4.6.29 AttributeFromIterator

This method converts the current position of an iterator to an XML attribute object. The iterator is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD AttributeFromIterator : SXmlAttribute
VAR_INPUT
  it : SXmlIterator;
END_VAR
```

**Sample call:**

```plaintext
xmlIterator := fbXml.AttributeBegin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlAttr := fbXml.AttributeFromIterator(xmlIterator);
  nAttrValue := fbXml.AttributeAsInt(xmlAttr);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
```

### 4.6.30 AttributeName

This method returns the name of a given attribute. The attribute is transferred to the method as input parameter.
### Syntax

METHOD AttributeName : STRING
VAR_INPUT
  a : SXmlAttribute;
END_VAR

Sample call:

sName := fbXml.AttributeName(xmlAttr);

#### 4.6.31 Attributes

This method is used to navigate through the DOM and returns an iterator for all attributes found at an XML node. The iterator can then be used for further navigation through the elements that were found. The node and a reference to the iterator are transferred to the method as input parameters.

Syntax

METHOD Attributes : SXmlAttribute
VAR_INPUT
  n : SXmlNode;
  it : REFERENCE TO SXmlIterator;
END_VAR

Sample call:

xmlRet := fbXml.Attributes(xmlNode, xmlIterator);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlMachineAttrRef := fbXml.Attribute(xmlIterator);
  xmlMachineAttrText := fbXml.AttributeText(xmlMachineAttrRef);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE

#### 4.6.32 AttributeText

This method returns the text of a given attribute. The attribute is transferred to the method as input parameter.

Syntax

METHOD AttributeText : STRING(255)
VAR_INPUT
  a : SXmlAttribute;
END_VAR

Sample call:

sText := fbXml.AttributeText(xmlAttr);

#### 4.6.33 Begin

This method returns an iterator over all child elements of an XML node, always starting from the first child element. The XML node is transferred to the method as input parameter.

Syntax

METHOD Begin : SXmlIterator
VAR_INPUT
  n : SXmlNode;
END_VAR

Sample call:

xmlIterator := fbXml.Begin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlNodeRef := fbXml.Node(xmlIterator);
  xmlNodeValue := fbXml.NodeText(xmlNodeRef);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
### 4.6.34 BeginByName

This method returns an iterator over all child elements of an XML node, starting at a particular element. The XML node is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD BeginByName : SXmlIterator
VAR_INPUT
    n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
    name : STRING;
END_VAR
```

**Sample call:**

```plaintext
xmlNode := fbXml.ChildByName(xmlDoc, 'Machines');
xmlIterator := fbXml.BeginByName(xmlNode, 'NameX');
WHILE NOT fbXml.IsEnd(xmlIterator) DO
    xmlNodeRef := fbXml.Node(xmlIterator);
    xmlNodeValue := fbXml.NodeText(xmlNodeRef);
    xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
```

### 4.6.35 Child

This method is used to navigate through the DOM. It returns a reference to the (first) child element of the current node. The start node is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD ChildByName : SXmlNode
VAR_INPUT
    n : SXmlNode;
END_VAR
```

**Sample call:**

```plaintext
xmlChild := fbXml.Child(xmlNode);
```

### 4.6.36 ChildByAttribute

This method is used to navigate through the DOM. It returns a reference to a child element in the XML document. The start node and the name and value of the attribute are transferred to the method as input parameters.

**Syntax**

```plaintext
METHOD ChildByAttribute : SXmlNode
VAR_INPUT
    n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
    attr : STRING;
    value : STRING;
END_VAR
```

**Sample call:**

```plaintext
xmlMachine1 := fbXml.ChildByAttribute(xmlMachines, 'Type', '1');
```

### 4.6.37 ChildByAttributeAndName

This method is used to navigate through the DOM. It returns a reference to a child element in the XML document. The start node, the name and value of the attribute, and the name of the child element are transferred to the method as input parameters.
### Syntax

**METHOD ChildByAttributeAndName** : SXmlNode  
VAR_INPUT  
  n : SXmlNode;  
END_VAR  
VAR_IN_OUT CONSTANT  
  attr : STRING;  
  value : STRING;  
  child : STRING;  
END_VAR  
Sample call:  
xmlMachine2 := fbXml.ChildByAttributeAndName(xmlMachines, 'Type', '2', 'Machine');

### 4.6.38 ChildByAttributeAndName

This method is used to navigate through the DOM. It returns a reference to a child element in the XML document. The start node and the name of the element to be returned are transferred to the method as input parameters.

### Syntax

**METHOD ChildByAttributeAndName** : SXmlNode  
VAR_INPUT  
  n : SXmlNode;  
END_VAR  
VAR_IN_OUT CONSTANT  
  attr : STRING;  
  value : STRING;  
  child : STRING;  
END_VAR  
Sample call:  
xmlMachines := fbXml.ChildByAttributeAndName(xmlDoc, 'Machines');

### 4.6.39 Children

This method is used to navigate through the DOM. It returns an iterator for several child elements found in the XML document. The iterator can then be used for further navigation through the elements that were found. The start node and a reference to the iterator are transferred to the method as input parameters.

### Syntax

**METHOD Children** : SXmlNode  
VAR_INPUT  
  n : SXmlNode;  
  it : REFERENCE TO SXmlIterator;  
END_VAR

Sample call:
xmlRet := fbXml.Children(xmlNode, xmlIterator);  
WHILE NOT fbXml.IsEnd(xmlIterator) DO  
  xmlMachineNodeRef := fbXml.Node(xmlIterator);  
  xmlMachineNodeText := fbXml.NodeText(xmlMachineNodeRef);  
  xmlIterator := fbXml.Next(xmlIterator);  
END_WHILE

### 4.6.40 ChildrenByName

This method is used to navigate through the DOM. It returns an iterator for several child elements found in the XML document. The iterator can then be used for further navigation through the elements that were found. The start node, the name of the child elements to be found and a reference to the iterator are transferred to the method as input parameters.
Syntax

METHOD ChildrenByName : SXmlNode
VAR_INPUT
   n : SXmlNode;
   it : REFERENCE TO SXmlIterator;
END_VAR
VAR_IN_OUT CONSTANT
   name : STRING;
END_VAR

Sample call:
xmlMachineNode := fbXml.ChildrenByName(xmlMachines, xmlIterator, 'Machine');
WHILE NOT fbXml.IsEnd(xmlIterator) DO
   xmlMachineNodeRef := fbXml.Node(xmlIterator);
   xmlMachineNodeText := fbXml.NodeText(xmlMachineNodeRef);
   xmlIterator := fbXml.Next(xmlIterator);
END_WHILE

4.6.41 Compare

This method checks two iterators for equality.

Syntax

METHOD Compare : BOOL
VAR_INPUT
   it1 : SXmlIterator;
   it2 : SXmlIterator;
END_VAR

Sample call:
bResult := fbXml.Compare(xmlIt1, xmlIt2);

4.6.42 CopyAttributeText

This method reads the value of an XML attribute and writes it to a variable of data type String. The XML attribute, the target variable and the length to be written are transferred to the method as input parameters. The method returns the actual size.

Syntax

METHOD CopyAttributeText : UDINT
VAR_INPUT
   a : SXmlAttribute;
END_VAR
VAR_IN_OUT CONSTANT
   sXml : STRING;
END_VAR
VAR_INPUT
   nXml : UDINT;
END_VAR

Sample call:
nLength := fbXml.CopyAttributeText(xmlAttr, sTarget, SIZEOF(sTarget));

4.6.43 CopyDocument

This method copies the contents of the DOM memory into a variable of the data type String. The length to be written and the variable into which the resulting string is to be written are transferred to the method as input parameters. The method returns the actually written length. Note that the size of the string variable is at least equal to the size of the XML document in the DOM.
Function blocks

Syntax
METHOD CopyDocument : UDINT
VAR_IN_OUT CONSTANT
  sXml : STRING;
END_VAR
VAR_INPUT
  nXml : UDINT;
END_VAR
Sample call:
  nLength := fbXml.CopyDocument(sTarget, SIZEOF(sTarget));

4.6.44 CopyNodeText

This method reads the value of an XML node and writes it to a variable of data type String. The XML node, the target variable and the length to be written are transferred to the method as input parameters. The method returns the actual size.

Syntax
METHOD CopyNodeText : UDINT
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  sXml : STRING;
END_VAR
VAR_INPUT
  nXml : UDINT;
END_VAR
Sample call:
  nLength := fbXml.CopyNodeText(xmlNode, sTarget, SIZEOF(sTarget));

4.6.45 CopyNodeXml

This method reads the XML structure of an XML node and writes it to a variable of data type String. The XML node, the target variable and the length to be written are transferred to the method as input parameters. The method returns the actual size.

Syntax
METHOD CopyNodeXml : UDINT
VAR_INPUT
  a : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  sXml : STRING;
END_VAR
VAR_INPUT
  nXml : UDINT;
END_VAR
Sample call:
  nLength := fbXml.CopyNodeXml(xmlNode, sTarget, SIZEOF(sTarget));

4.6.46 FirstNodeByPath

This method navigates through an XML document using a path that was transferred to the method. The path and the start node are transferred to the method as input parameters. The path is specified with "/" as separator. The method returns a reference to the XML node that was found.
### GetAttributeTextLength

This method returns the length of the value of an XML attribute. The XML attribute is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD GetAttributeTextLength : UDINT
VAR_INPUT
  a : SXmlAttribute;
END_VAR
```

**Sample call:**

```plaintext
nLength := fbXml.GetAttributeTextLength(xmlAttr);
```

### GetDocumentLength

This method returns the length of an XML document in bytes.

**Syntax**

```plaintext
METHOD GetDocumentLength : UDINT
```

**Sample call:**

```plaintext
nLength := fbXml.GetDocumentLength();
```

### GetDocumentNode

This method returns the root node of an XML document. This is not the same as the first XML node in the document (the method GetRootNode() should be used for this). The method can also be used to create an empty XML document in the DOM.

**Syntax**

```plaintext
METHOD GetDocumentNode : SXmlNode
```

**Sample call:**

```plaintext
objRoot := fbXml.GetDocumentNode();
```

### GetNodeTextLength

This method returns the length of the value of an XML node. The XML node is transferred to the method as input parameter.

**Syntax**

```plaintext
METHOD GetNodeTextLength : UDINT
VAR_INPUT
  n : SXmlNode;
END_VAR
```

**Sample call:**

```plaintext
xmlFoundNode := fbXml.FirstNodeByPath(xmlStartNode, 'Level1/Level2/Level3');
```
### 4.6.51 GetNodeXmlLength

This method returns the length of the XML structure of an XML node. The XML node is transferred to the method as input parameter.

**Syntax**

METHOD GetNodeXmlLength : UDINT
VAR_INPUT
n : SXmlNode;
END_VAR

Sample call:

nLength := fbXml.GetNodeXmlLength(xmlNode);

### 4.6.52 GetRootNode

This method returns a reference to the first XML node in the XML document.

**Syntax**

METHOD GetRootNode : SXmlNode

Sample call:

xmlRootNode := fbXml.GetRootNode();

### 4.6.53 InsertAttributeCopy

This method adds an attribute to an XML node. The name and value of an existing attribute are copied. The attribute can be placed at a specific position. The XML node, the position and a reference to the existing attribute object are transferred to the method as input parameters. The method returns a reference to the newly added attribute.

**Syntax**

METHOD InsertAttributeCopy : SXmlAttribute
VAR_INPUT
n : SXmlNode;
before : SXmlAttribute;
copy : SXmlAttribute;
END_VAR

Sample call:

xmlNewAttr := fbXml.InsertAttributeCopy(xmlNode, xmlBeforeAttr, xmlCopyAttr);

### 4.6.54 InsertAttribute

This method adds an attribute to an XML node. The attribute can be placed at a specific position. The XML node and the position and name of the new attribute are transferred to the method as input parameters. The method returns a reference to the newly added attribute. A value for the attribute can then be entered using the SetAttribute() method, for example.

**Syntax**

METHOD InsertAttribute : SXmlAttribute
VAR_INPUT
n : SXmlNode;
before : SXmlAttribute;
END_VAR
4.6.55  **InsertChild**

This method adds a node to an existing XML node. The new node can be placed at a specific location. The existing XML node and the position and name of the new node are transferred to the method as input parameters. The method returns a reference to the newly added node. A value for the node can then be entered using the SetChild() method, for example.

**Syntax**

METHOD InsertChild : SXmlNode

VAR_INPUT
  n : SXmlNode;
  before : SXmlNode;

VAR_IN_OUT CONSTANT
  name : STRING;

END_VAR

**Sample call:**

```plaintext
xmlNewNode := fbXml.InsertChild(xmlNode, xmlBeforeNode, 'SomeName');
```

4.6.56  **InsertCopy**

This method adds a new node to an existing XML node and copies an existing node. The new node can be placed anywhere in the existing node. The XML node, the position and a reference to the existing node object are transferred to the method as input parameters. The method returns a reference to the newly added node.

**Syntax**

METHOD InsertCopy : SXmlNode

VAR_INPUT
  n : SXmlNode;
  before : SXmlNode;
  copy : SXmlNode;

END_VAR

**Sample call:**

```plaintext
xmlNewNode := fbXml.InsertCopy(xmlNode, xmlBeforeNode, xmlCopyNode);
```

4.6.57  **IsEnd**

This method checks whether a given XML iterator is at the end of the iteration that is to be performed.

**Syntax**

METHOD IsEnd : BOOL

VAR_INPUT
  it : SXmlIterator;

END_VAR

**Sample call:**

```plaintext
xmlIterator := fbXml.Begin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlNodeRef := fbXml.Node(xmlIterator);
```
Function blocks

```plaintext
xmlNodeValue := fbXml.NodeText(xmlNodeRef);
xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
```

### 4.6.58 LoadDocumentFromFile

This method loads an XML document from a file. The absolute path to the file is transferred to the method as input parameter.

A rising edge on the input parameter bExec triggers the loading procedure. The asynchronous process is terminated as soon as the reference bExec is set back to FALSE from the method. When the process ends, the return value of the method indicates for one call whether the loading of the file was successful (TRUE) or failed (FALSE).

**Syntax**

METHOD LoadDocumentFromFile : BOOL
VAR_IN_OUT CONSTANT
  sFile : STRING;
END_VAR
VAR_INPUT
  bExec : REFERENCE TO BOOL;
END_VAR
VAR_OUTPUT
  hrErrorCode: HRESULT;
END_VAR

**Sample call:**

```plaintext
IF bLoad THEN
  bLoaded := fbXml.LoadDocumentFromFile('C:\Test.xml', bLoad);
END_IF
```

### 4.6.59 NewDocument

This method creates an empty XML document in the DOM memory.

**Syntax**

METHOD NewDocument : BOOL

**Sample call:**

```plaintext
fbXml.NewDocument();
```

### 4.6.60 Next

This method sets an XML iterator for the next object that is to be processed.

**Syntax**

METHOD Next : SXmlIterator
VAR_INPUT
  it : SXmlIterator;
END_VAR

**Sample call:**

```plaintext
xmlIterator := fbXml.Begin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlNodeRef := fbXml.Node(xmlIterator);
  xmlNodeValue := fbXml.NodeText(xmlNodeRef);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE
```
### 4.6.61 NextAttribute

This method returns the next attribute for a given XML attribute.

**Syntax**

```plaintext
METHOD NextAttribute : SXmlAttribute
VAR_INPUT
    a : SXmlAttribute;
END_VAR

Sample call:
xmlNextAttr := fbXml.NextAttribute(xmlAttr);
```

### 4.6.62 NextByName

This method sets an XML iterator for the next object that is to be processed, which is identified by its name.

**Syntax**

```plaintext
METHOD NextByName : SXmlIterator
VAR_INPUT
    it : SXmlIterator;
END_VAR
VAR_IN_OUT CONSTANT
    name : STRING;
END_VAR

Sample call:
xmlIterator := fbXml.Begin(xmlNode);
WHILE NOT fbXml.IsEnd(xmlIterator) DO
    xmlNodeRef := fbXml.Node(xmlIterator);
    xmlNodeValue := fbXml.NodeText(xmlNodeRef);
    xmlIterator := fbXml.NextByName(xmlIterator, 'SomeName');
END_WHILE
```

### 4.6.63 NextSibling

This method returns the next direct node for a given XML node at the same XML level.

**Syntax**

```plaintext
METHOD NextSibling : SXmlNode
VAR_INPUT
    n : SXmlNode;
END_VAR

Sample call:
xmlSibling := fbXml.NextSibling(xmlNode);
```

### 4.6.64 NextSiblingByName

This method returns the next direct node for a given XML node with a particular name at the same XML level.

**Syntax**

```plaintext
METHOD NextSiblingByName : SXmlNode
VAR_INPUT
    n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
    name : STRING;
END_VAR
```
Sample call:
xmlSibling := fbXml.NextSibling(xmlNode, 'SomeName');

4.6.65 Node

This method is used in conjunction with an iterator to navigate through the DOM. The iterator is transferred to the method as input parameter. The method then returns the current XML node as return value.

Syntax
METHOD Node : SXmlNode
VAR_INPUT
  it : SXmlNode;
END_VAR

Sample call:
xmlMachineNode := fbXml.ChildrenByName(xmlMachines, xmlIterator, 'Machine');
WHILE NOT fbXml.IsEnd(xmlIterator) DO
  xmlMachineNode := fbXml.Node(xmlIterator);
  xmlMachineNodeValue := fbXml.NodeText(xmlMachineNode);
  xmlIterator := fbXml.Next(xmlIterator);
END_WHILE

4.6.66 NodeAsBool

This method returns the text of an XML node as data type Boolean. The XML node is transferred to the method as input parameter.

Syntax
METHOD NodeAsBool : BOOL
VAR_INPUT
  n : SXmlNode;
END_VAR

Sample call:
bXmlNode:= fbXml.NodeAsBool(xmlMachine1);

4.6.67 NodeAsDouble

This method returns the text of an XML node as data type Double. The XML node is transferred to the method as input parameter.

Syntax
METHOD NodeAsDouble : LREAL
VAR_INPUT
  n : SXmlNode;
END_VAR

Sample call:
lXmlNode:= fbXml.NodeAsDouble(xmlMachine1);

4.6.68 NodeAsFloat

This method returns the text of an XML node as data type Float. The XML node is transferred to the method as input parameter.
Syntax
METHOD NodeAsFloat : REAL
VAR_INPUT
   n : SXmlNode;
END_VAR

Sample call:
  rXmlNode:= fbXml.NodeAsFloat(xmlMachine1);

4.6.69 NodeAsInt
This method returns the text of an XML node as a data type Integer. The XML node is transferred to the method as input parameter.

Syntax
METHOD NodeAsInt : DINT
VAR_INPUT
   n : SXmlNode;
END_VAR

Sample call:
  nXmlNode:= fbXml.NodeAsInt(xmlMachine1);

4.6.70 NodeAsLint
This method returns the text of an XML node as a data type Integer64. The XML node is transferred to the method as input parameter.

Syntax
METHOD NodeAsLint : LINT
VAR_INPUT
   n : SXmlNode;
END_VAR

Sample call:
  nXmlNode:= fbXml.NodeAsLint(xmlMachine1);

4.6.71 NodeAsUint
This method returns the text of an XML node as data type Unsigned Integer. The XML node is transferred to the method as input parameter.

Syntax
METHOD NodeAsUint : UDINT
VAR_INPUT
   n : SXmlNode;
END_VAR

Sample call:
  nXmlNode:= fbXml.NodeAsUint(xmlMachine1);

4.6.72 NodeAsUlint
This method returns the text of an XML node as data type Unsigned Integer64. The XML node is transferred to the method as input parameter.
4.6.73 NodeName

This method returns the name of an XML node. A reference to the XML node is transferred to the method as input parameter.

Syntax

METHOD NodeName : STRING
VAR_INPUT
  n : SXmlNode;
END_VAR

Sample call:

sNodeName := fbXml.NodeName(xmlMachine1);

4.6.74 NodeText

This method returns the text of an XML node. The XML node is transferred to the method as input parameter.

Syntax

METHOD NodeText : STRING(255)
VAR_INPUT
  n : SXmlNode;
END_VAR

Sample call:

sMachine1Name := fbXml.NodeText(xmlMachine1);

4.6.75 ParseDocument

This method loads an XML document into the DOM memory for further processing. The XML document exists as a string and is transferred to the method as input parameter. A reference to the XML document in the DOM is returned to the caller.

Syntax

METHOD ParseDocument : SXmlNode
VAR_IN_OUT CONSTANT
  sXml : STRING;
END_VAR

Sample call:

xmlDoc := fbXml.ParseDocument(sXmlToParse);

4.6.76 RemoveChild

This method removes an XML child node from a given XML node. The two XML nodes are transferred to the method as input parameters. The method returns TRUE if the operation was successful and the XML node was removed.
**Syntax**

METHOD RemoveChild : BOOL
VAR_INPUT
  n : SXmlNode;
  child : SXmlNode;
END_VAR

**Sample call:**

bRemoved := fbXml.RemoveChild(xmlParent, xmlChild);

---

### 4.6.77 RemoveChildByName

This method removes an XML child node from a given XML node. The node to be removed is addressed by its name. If there is more than one child node, the last child node is removed. The method returns TRUE if the operation was successful and the XML node was removed.

**Syntax**

METHOD RemoveChildByName : BOOL
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  name : STRING;
END_VAR

**Sample call:**

bRemoved := fbXml.RemoveChildByName(xmlParent, 'SomeName');

---

### 4.6.78 SaveDocumentToFile

This method saves the current XML document in a file. The absolute path to the file is transferred to the method as input parameter.

A rising edge at the input parameter bExec triggers the saving procedure. The asynchronous process is terminated as soon as the reference bExec is set back to FALSE from the method. When the process ends, the return value of the method indicates for one call whether saving of the file was successful (TRUE) or failed (FALSE).

**Syntax**

METHOD SaveDocumentToFile : BOOL
VAR_IN_OUT CONSTANT
  sFile : STRING;
END_VAR
VAR_INPUT
  bExec : REFERENCE TO BOOL;
END_VAR
VAR_OUTPUT
  hrErrorCode: HRESULT;
END_VAR

**Sample call:**

IF bSave THEN
  bSaved = fbXml.SaveDocumentToFile('C:\Test.xml', bSave);
END_IF

---

### 4.6.79 SetAttribute

This method sets the value of an attribute. The value has the data type String.
Syntax
METHOD SetAttribute : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
END_VAR
VAR_IN_OUT CONSTANT
  value : STRING;
END_VAR
Sample call:
xmlAttr := fbXml.SetAttribute(xmlExistingAttr, 'Test');

4.6.80 SetAttributeAsBool
This method sets the value of an attribute. The value has the data type Boolean.

Syntax
METHOD SetAttributeAsBool : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : BOOL;
END_VAR
Sample call:
xmlAttr := fbXml.SetAttributeAsBool(xmlExistingAttr, TRUE);

4.6.81 SetAttributeAsDouble
This method sets the value of an attribute. The value here has the data type Double.

Syntax
METHOD SetAttributeAsDouble : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : LREAL;
END_VAR
Sample call:
xmlAttr := fbXml.SetAttributeAsDouble(xmlExistingAttr, 42.42);

4.6.82 SetAttributeAsFloat
This method sets the value of an attribute. The value has the data type Float.

Syntax
METHOD SetAttributeAsFloat : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : REAL;
END_VAR
Sample call:
xmlAttr := fbXml.SetAttributeAsFloat(xmlExistingAttr, 42.42);

4.6.83 SetAttributeAsInt
This method sets the value of an attribute. The value has the data type Integer.
Syntax

METHOD SetAttributeAsInt : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : DINT;
END_VAR

Sample call:
xmlAttr := fbXml.SetAttributeAsInt(xmlExistingAttr, 42);

4.6.84 SetAttributeAsLint

This method sets the value of an attribute. The value has the data type Integer64.

Syntax

METHOD SetAttributeAsLint : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : LINT;
END_VAR

Sample call:
xmlAttr := fbXml.SetAttributeAsLint(xmlExistingAttr, 42);

4.6.85 SetAttributeAsUint

This method sets the value of an attribute. The value has the data type Unsigned Integer.

Syntax

METHOD SetAttributeAsUint : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : UDINT;
END_VAR

Sample call:
xmlAttr := fbXml.SetAttributeAsUint(xmlExistingAttr, 42);

4.6.86 SetAttributeAsUlint

This method sets the value of an attribute. The value has the data type Unsigned Integer64.

Syntax

METHOD SetAttributeAsUlint : SXmlAttribute
VAR_INPUT
  a : SXmlAttribute;
  value : ULINT;
END_VAR

Sample call:
xmlAttr := fbXml.SetAttributeAsUlint(xmlExistingAttr, 42);

4.6.87 SetChild

This method sets the value of an XML node. The value is transferred to the method as input parameter of data type String. The input parameter cdata indicates whether the value of the node is to be encapsulated in a CDATA function block, so that certain special characters such as "<" and ">") are allowed as values.
Syntax
METHOD SetChild : SXmlNode
VAR_INPUT
  n : SXmlNode;
END_VAR
VAR_IN_OUT CONSTANT
  value : STRING;
END_VAR
VAR_INPUT
  cdata : BOOL;
END_VAR
Sample call:
xmlNode := fbXml.SetChild(xmlExistingNode, 'SomeText', FALSE);

4.6.88 SetChildAsBool
This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Boolean.

Syntax
METHOD SetChildAsBool : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : BOOL;
END_VAR
Sample call:
xmlNode := fbXml.SetChildAsBool(xmlExistingNode, TRUE);

4.6.89 SetChildAsDouble
This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Double.

Syntax
METHOD SetChildAsDouble : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : LREAL;
END_VAR
Sample call:
xmlNode := fbXml.SetChildAsDouble(xmlExistingNode, 42.42);

4.6.90 SetChildAsFloat
This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Float.

Syntax
METHOD SetChildAsFloat : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : REAL;
END_VAR
Sample call:
xmlNode := fbXml.SetChildAsFloat(xmlExistingNode, 42.42);
4.6.91    SetChildAsInt

This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Integer.

Syntax

METHOD SetChildAsInt : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : DINT;
END_VAR

Sample call:
xmlNode := fbXml.SetChildAsInt(xmlExistingNode, 42);

4.6.92    SetChildAsLint

This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Integer64.

Syntax

METHOD SetChildAsLint : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : LINT;
END_VAR

Sample call:
xmlNode := fbXml.SetChildAsLint(xmlExistingNode, 42);

4.6.93    SetChildAsUint

This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Unsigned Integer.

Syntax

METHOD SetChildAsUint : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : UDINT;
END_VAR

Sample call:
xmlNode := fbXml.SetChildAsUint(xmlExistingNode, 42);

4.6.94    SetChildAsUlint

This method sets the value of an XML node. The value is transferred to the method as input parameter of data type Unsigned Integer64.

Syntax

METHOD SetChildAsUlint : SXmlNode
VAR_INPUT
  n : SXmlNode;
  value : ULINT;
END_VAR

Sample call:
xmlNode := fbXml.SetChildAsUlint(xmlExistingNode, 42);
4.7  FB_JwtEncode

The function block enables the creation and signing of a JSON Web Token (JWT).

Syntax

Definition:

FUNCTION_BLOCK FB_JwtEncode
VAR_INPUT
  bExecute       : BOOL;
  sHeaderAlg     : STRING(46);
  sPayload       : STRING(1023);
  sKeyFilePath   : STRING(511);
  tTimeout       : TIME;
  pKey           : PVOID;
  nKeySize       : UDINT;
  nJwtSize       : UDINT;
END_VAR
VAR_IN_OUT CONSTANT
  sJwt           : STRING;
END_VAR
VAR_OUTPUT
  bBusy          : BOOL;
  bError         : BOOL;
  hrErrorCode    : HRESULT;
  initStatus     : HRESULT;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>A rising edge activates processing of the function block.</td>
</tr>
<tr>
<td>sHeaderAlg</td>
<td>STRING(46)</td>
<td>The algorithm to be used for the JWT header, e.g. RS256.</td>
</tr>
<tr>
<td>sPayload</td>
<td>STRING(1023)</td>
<td>The JWT payload to be used.</td>
</tr>
<tr>
<td>sKeyFilePath</td>
<td>STRING(511)</td>
<td>Path to the private key to be used for the signature of the JWT.</td>
</tr>
<tr>
<td>tTimeout</td>
<td>TIME</td>
<td>ADS timeout, which is used internally for file access to the private key.</td>
</tr>
<tr>
<td>pKey</td>
<td>PVOID</td>
<td>Buffer for the private key to be read.</td>
</tr>
<tr>
<td>nKeySize</td>
<td>UDINT</td>
<td>Maximum size of the buffer.</td>
</tr>
<tr>
<td>sJwt</td>
<td>STRING</td>
<td>Contains the fully coded and signed JWT after the function block has been processed.</td>
</tr>
<tr>
<td>nJwtSize</td>
<td>UDINT</td>
<td>Size of the generated JWT including zero termination.</td>
</tr>
</tbody>
</table>

Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>Is TRUE as long as processing of the function block is in progress.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>Becomes TRUE as soon as an error situation occurs.</td>
</tr>
<tr>
<td>hrErrorCode</td>
<td>HRESULT</td>
<td>Returns an error code if the bError output is set. An explanation of the possible error codes can be found in the Appendix.</td>
</tr>
<tr>
<td>initStatus</td>
<td>HRESULT</td>
<td>Returns an error code in case of a failed initialization of the function block.</td>
</tr>
</tbody>
</table>
## Requirements

<table>
<thead>
<tr>
<th>TwinCAT version</th>
<th>Hardware</th>
<th>Libraries to be integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT 3.1, Build 4024.4</td>
<td>x86, x64, ARM</td>
<td>Tc3.JsonXml 3.3.6.0</td>
</tr>
</tbody>
</table>
5 Interfaces

5.1 ITcJsonSaxHandler

5.1.1 OnBool

This callback method is triggered if a value of the data type BOOL was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnBool : HRESULT
VAR_INPUT
    value : BOOL;
END_VAR

5.1.2 OnDint

This callback method is triggered if a value of the data type DINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnDint : HRESULT
VAR_INPUT
    value : DINT;
END_VAR

5.1.3 OnEndArray

This callback method is triggered if a square closing bracket, which corresponds to the JSON synonym for an ending array, was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnEndArray : HRESULT

5.1.4 OnEndObject

This callback method is triggered if a curly closing bracket, which corresponds to the JSON synonym for an ending object, was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnEndObject : HRESULT

5.1.5 OnKey

This callback method is triggered if a property was found at the position of the SAX reader. The property name lies on the input/output parameter key and its length on the input parameter len. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.
Syntax
METHOD OnKey : HRESULT
VAR_IN_OUT CONSTANT
    key : STRING;
END_VAR
VAR_INPUT
    len : UDINT;
END_VAR

5.1.6 OnLint
This callback method is triggered if a value of the data type LINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax
METHOD OnLint : HRESULT
VAR_INPUT
    value : LINT;
END_VAR

5.1.7 OnLreal
This callback method is triggered if a value of the data type LREAL was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax
METHOD OnLreal : HRESULT
VAR_INPUT
    value : LREAL;
END_VAR

5.1.8 OnNull
This callback method is triggered if a NULL value was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax
METHOD OnNull : HRESULT

5.1.9 OnStartArray
This callback method is triggered if a square opening bracket, which corresponds to the JSON synonym for a starting array, was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax
METHOD OnStartArray : HRESULT

5.1.10 OnStartObject
This callback method is triggered if a curly opening bracket, which corresponds to the JSON synonym for a starting object, was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.
### Interfaces

**Syntax**

METHOD OnStartObject : HRESULT

#### 5.1.11 OnString

This callback method is triggered if a value of the data type STRING was found at the position of the SAX reader. The In/Out parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

**Syntax**

METHOD OnString : HRESULT
VAR_IN_OUT CONSTANT
  value : STRING;
END_VAR
VAR_INPUT
  len : UDINT;
END_VAR

#### 5.1.12 OnUdint

This callback method is triggered if a value of the data type UDINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

**Syntax**

METHOD OnUdint : HRESULT
VAR_INPUT
  value : UDINT;
END_VAR

#### 5.1.13 OnUlint

This callback method is triggered if a value of the data type ULIMIT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

**Syntax**

METHOD OnUlint : HRESULT
VAR_INPUT
  value : ULIMIT;
END_VAR

#### 5.2 ITcJsonSaxValues

##### 5.2.1 OnBoolValue

This callback method is triggered if a value of the data type BOOL was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

**Syntax**

METHOD OnBoolValue : HRESULT
VAR_INPUT
  level : UDINT;
  infos : POINTER TO TcJsonLevelInfo;
  value : BOOL;
END_VAR
5.2.2 OnDintValue

This callback method is triggered if a value of the data type DINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnDintValue : HRESULT
VAR_INPUT
    level : UDINT;
    infos : POINTER TO TcJsonLevelInfo;
    value : DINT;
END_VAR

5.2.3 OnLintValue

This callback method is triggered if a value of the data type LINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnLintValue : HRESULT
VAR_INPUT
    level : UDINT;
    infos : POINTER TO TcJsonLevelInfo;
    value : LINT;
END_VAR

5.2.4 OnLrealValue

This callback method is triggered if a value of the data type LREAL was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnLrealValue : HRESULT
VAR_INPUT
    level : UDINT;
    infos : POINTER TO TcJsonLevelInfo;
    value : LREAL;
END_VAR

5.2.5 OnNullValue

This callback method is triggered if a NULL value was found at the position of the SAX reader. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnNull : HRESULT
VAR_INPUT
    level : UDINT;
    infos : POINTER TO TcJsonLevelInfo;
END_VAR

5.2.6 OnStringValue

This callback method is triggered if a value of the data type STRING was found at the position of the SAX reader. The input/output parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.
5.2.7 OnUdintValue

This callback method is triggered if a value of the data type UDINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnUdintValue : HRESULT
VAR_INPUT
   level : UDINT;
   infos : POINTER TO TcJsonLevelInfo;
   value : UDINT;
END_VAR

5.2.8 OnUlintValue

This callback method is triggered if a value of the data type ULINT was found at the position of the SAX reader. The input parameter value contains the value found. The SAX parsing procedure is aborted by setting the return value HRESULT to S_FALSE.

Syntax

METHOD OnUlintValue : HRESULT
VAR_INPUT
   level : UDINT;
   infos : POINTER TO TcJsonLevelInfo;
   value : ULINT;
END_VAR
6 Samples

6.1 Tc3JsonXmlSampleJsonDataType

Sample of the automatic conversion of structures into a JSON message

This sample illustrates how a data structure can be converted into a JSON message (and vice versa). In the conversion the layout of a structure is converted one-to-one into a corresponding JSON equivalent. Additional metadata can be created via PLC attributes on the member variables of the structure.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/3664376331.zip

---

Layout of the data structure to be converted

```plaintext
TYPE ST_Values :
    STRUCT
        {attribute 'Unit' := 'm/s'}
        {attribute 'DisplayName' := 'Speed'}
        Sensor1 : REAL;
    {attribute 'Unit' := 'V'}
        {attribute 'DisplayName' := 'Voltage'}
        Sensor2 : DINT;
    {attribute 'Unit' := 'A'}
        {attribute 'DisplayName' := 'Current'}
        Sensor3 : DINT;
END_STRUCT
END_TYPE
```

---

Declaration range

```plaintext
PROGRAM MAIN
VAR
    dtTimestamp : DATE_AND_TIME := DT#2017-04-04-12:42:42;
    fbJson : FB_JsonSaxWriter;
    fbJsonDataType : FB_JsonReadWriteDataType;
    sJsonDoc : STRING(255);
    sJsonDoc2 : STRING(2000);
    stValues : ST_Values;
END_VAR
```

---

Implementation range

Two ways of generating the JSON message are shown, starting with the instance fbJson of the function block FB_JsonSaxWriter. The GetDocument() method can be used with a JSON message with no more than 255 characters. However, the CopyDocument() method must be used with larger JSON messages.

```plaintext
fbJson.ResetDocument();
fbJson.StartObject();
fbJson.AddKeyDateTime('Timestamp', dtTimestamp);
fbJsonDataType.AddJsonKeyValueFromSymbol(fbJson, 'Values', 'ST_Values', SIZEOF(stValues),ADR(stValues));
fbJsonDataType.AddJsonKeyPropertiesFromSymbol(fbJson, 'MetaData', 'ST_Values', 'Unit|DisplayName');
fbJson.EndObject();
sJsonDoc := fbJson.GetDocument();
fbJson.CopyDocument(sJsonDoc2, SIZEOF(sJsonDoc2));
```

---

Resulting JSON message

```json
{
    "Timestamp": "2017-04-04T12:42:42",
    "Values": {
        "Sensor1": 0.0,
        "Sensor2": 0,
        "Sensor3": 0
    },
    "MetaData": {
        "Sensor1": {
```
As an alternative, the method AddJsonValueFromSymbol() can also be used to generate a JSON format directly from a data structure.

```plaintext
fbJson.ResetDocument();
fbJsonDataType.AddJsonValueFromSymbol(fbJson, 'ST_Values', SIZEOF(stValues), ADR(stValues));
sJsonDoc := fbJson.GetDocument();
fbJson.CopyDocument(sJsonDoc2, SIZEOF(sJsonDoc2));
```

The resulting JSON object looks like this:

```plaintext
{
  "Sensor1": 0.0,
  "Sensor2": 0,
  "Sensor3": 0
}
```

**Conversion of a JSON message back to a data structure**

The above samples show how a JSON object can be generated from a data structure in a simple manner. There is also a corresponding method in the Tc3_JsonXml library for the reverse process, i.e. the extraction of values from a (received) JSON object back into a data structure. This application is made possible by calling the method SetSymbolFromJson().

```plaintext
fbJsonDataType.SetSymbolFromJson(someJson, 'ST_Values', SIZEOF(stValuesReceive), ADR(stValuesReceive));
```

The string variable sJsonDoc2 contains the JSON object, which is transferred into the structure instance stValuesReceive by calling the method.

---

**Target data structure**

The target data structure must match the structure of the JSON document. Otherwise SetSymbolFromJson() returns FALSE.

---

### 6.2 Tc3JsonXmlSampleJsonSaxReader

**Sample of the parsing of JSON documents via SAX Reader**

This sample illustrates how a JSON message can be run through programmatically. The function block FB_JSONSaxReader is used as the basis.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/3664750475.zip

**Declaration range**

```plaintext
PROGRAM MAIN
VAR
  fbJson  : FB_JSONSaxReader;
  pJsonParse : JsonSaxHandler;
  sJsonDoc  : STRING(255) := '"Values":
  ["Timestamp":"2017-04-04T12:42:42","Sensor1":42.42,"Sensor2":42}];
END_VAR
```
Implementation range

Through the calling of the Parse() method, the transfer of the JSON message as a STRING and the interface pointer to a function block instance that implements the interface ItcJsonSaxHandler, the SAX Reader is activated and the corresponding callback methods are run through.

```plaintext
fbJson.Parse(sJson := sJsonDoc, ipHdl := pJsonParse);
```

Callback methods

The callback methods are called on the instance of the function block that implements the interface ItcJsonSaxHandler. Each callback method represents a "found" element in the JSON message. For example, the callback method OnStartObject() is called as soon as an opening curly bracket has been detected. According to the example JSON message mentioned above, therefore, the following callback methods are run through in this order:

1. OnStartObject(), due to the first opening curly bracket
2. OnKey(), due to the property "Values"
3. OnStartObject(), due to the second opening curly bracket
4. OnKey(), due to the property "Timestamp"
5. OnString(), due to the value of the property "Timestamp"
6. OnKey(), due to the property "Sensor1"
7. OnLreal(), due to the value of the property "Sensor1"
8. OnKey(), due to the property "Sensor2"
9. OnUdint(), due to the value of the property "Sensor2"
10. OnEndObject(), due to the first closing curly bracket
11. OnEndObject(), due to the second closing curly bracket

Within the callback methods the current state is defined and saved via an instance of the enum E_JsonStates. This can also be used to determine whether the JSON message is valid. For example, if the callback method OnLreal() is called and the state is not the expected State 70 (JSON_STATE_ONLREAL), the return value S_FALSE can be returned to the method. The SAX Reader then automatically cancels the further processing.

6.3 Tc3JsonXmlSampleJsonSaxWriter

Sample of the creation of JSON documents via SAX Writer

This sample illustrates how a JSON message can be created over the DAX mechanism. The function block FB_JsonSaxWriter is used as the basis.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/3664753419.zip

Declaration range

```plaintext
PROGRAM MAIN
VAR
  dtTimestamp : DATE_AND_TIME := DT#2017-04-04-12:42:42;
  fbJson      : FB_JsonSaxWriter;
  sJsonDoc    : STRING(255);
END_VAR
```

Implementation range

The SAX mechanism runs sequentially through the JSON document to be created, i.e. the corresponding elements are run though and created one after the other.

```plaintext
fbJson.StartObject();
fbJson.AddKey('Timestamp');
fbJson.AddDateTime(dtTimestamp);
fbJson.AddKey('Values');
fbJson.StartObject();
fbJson.AddKey('Sensor1');
fbJson.AddReal(42.42);
```
fbJson.AddKey('Sensor2');
fbJson.AddDInt(42);
fbJson.AddKey('Sensor3');
fbJson.AddBool(TRUE);
fbJson.EndObject();
sJsonDoc := fbJson.GetDocument();
fbJson.ResetDocument();

Resulting JSON message

```json
{
"Timestamp": "2017-04-04T12:42:42",
"Values": {
  "Sensor1": 42.42,
  "Sensor2": 42,
  "Sensor3": true
}
}
```

6.4 Tc3JsonXmlSampleJsonDomReader

This sample illustrates how a JSON message can be run through programmatically on the basis of DOM. The function block FB_JsonDomParser is used as the basis.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/3916597387.zip

Declaration range

```plaintext
PROGRAM MAIN
VAR
  fbJson : FB_JsonDomParser;
  jsonDoc : SJsonValue;
  jsonProp : SJsonValue;
  jsonValue : SJsonValue;
  bHasMember : BOOL;
  sMessage := '{"serialNumber":"G030PT028191AC4R","batteryVoltage":"1547mV","clickType":"SINGLE"}';
  stReceivedData : ST_ReceivedData;
END_VAR
```

Implementation range

The JSON message is loaded into the DOM tree using the ParseDocument() method. You can subsequently check whether it contains a certain property using the HasMember() method. The FindMember() method selects the property. The GetString() method extracts its value.

```plaintext
jsonDoc := fbJson.ParseDocument(sMessage);
```

```plaintext
bHasMember := fbJson.HasMember(jsonDoc, 'serialNumber');
  IF (bHasMember) THEN
    bHasMember := FALSE;
    jsonProp := fbJson.FindMember(jsonDoc, 'serialNumber');
    stReceivedData.serialNumber := fbJson.GetString(jsonProp);
  END_IF
```

```plaintext
bHasMember := fbJson.HasMember(jsonDoc, 'batteryVoltage');
  IF (bHasMember) THEN
    bHasMember := FALSE;
    jsonProp := fbJson.FindMember(jsonDoc, 'batteryVoltage');
    stReceivedData.batteryVoltage := fbJson.GetString(jsonProp);
  END_IF
```

```plaintext
bHasMember := fbJson.HasMember(jsonDoc, 'clickType');
  IF (bHasMember) THEN
    bHasMember := FALSE;
    jsonProp := fbJson.FindMember(jsonDoc, 'clickType');
    stReceivedData.clickType := fbJson.GetString(jsonProp);
  END_IF
```

The use of the method HasMember() is not absolutely necessary, since the method FindMember() already returns 0 if a property was not found. The code shown above can also be implemented as follows:
Nested JSON objects

The approach is similar with nested JSON objects. Since the entire document is located in the DOM, it is simple to navigate. Let's take a JSON object that looks like this:

`sMessage : STRING(255) := '{"Values":{"serial":"G030PT028191AC4R"}}';`

The property we are looking for is located in the sub-object "Values". The following code shows how to extract the property.

```
jsonDoc := fbJson.ParseDocument(sMessage);
bHasMember := fbJson.HasMember(jsonDoc, 'Values');
IF (bHasMember) THEN
    bHasMember := FALSE;
    jsonProp := fbJson.FindMember(jsonDoc, 'Values');
    IF jsonProp <> 0 THEN
        jsonSerial := fbJson.FindMember(jsonProp, 'serial');
        stReceivedData.serialNumber := fbJson.GetString(jsonSerial);
    END_IF
END_IF
```

6.5 Tc3JsonXmlSampleXmlDomReader

This sample illustrates how an XML document can be processed programmatically based on DOM. The function block FB_XmlDomParser is used as a basis.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/5529225227.zip

Declaration range

```
PROGRAM MAIN
VAR
    fbXml : FB_XmlDomParser;
    xmlDoc : SXmlNode;
    xmlMachines : SXmlNode;
    xmlMachine1 : SXmlNode;
    xmlMachine2 : SXmlNode;
    xmlIterator : SXmlIterator;
    xmlMachineNode : SXmlNode;
    xmlMachine1AttributeRef : SXmlAttribute;
    xmlMachine2AttributeRef : SXmlAttribute;
    sMachine1Name : STRING;
    sMachine2Name : STRING;
    nMachineAttribute : DINT;
    nMachine1Attribute : DINT;
    nMachine2Attribute : DINT;
    sMessageToParse : STRING(255) := '<Machines><Machine Type="1" Test="3">Wilde Nelli</Machine><Machine Type="2">Huber8</Machine></Machines>'; END_VAR
```

Implementation range

The implementation section shows various options for parsing an XML document.
6.6  Tc3JsonXmlSampleXmlDomWriter

This sample illustrates how an XML document can be created programmatically based on DOM. The function block FB_XmlDomParser is used as a basis.

Download: https://infosys.beckhoff.com/content/1033/TcPlcLib_Tc3_JsonXml/Resources/zip/5529228299.zip

Declaration range

PROGRAM MAIN
VAR
  fbXml : FB_XmlDomParser;
  objRoot : SXmlNode;
  objMachines : SXmlNode;
  objMachine : SXmlNode;
  objControllers : SXmlNode;
  objController : SXmlNode;
  objAttribute : SXmlAttribute;
  sXmlString : STRING(1000);
  bCreate : BOOL := FALSE;
  bSave : BOOL := TRUE;
  nLength : UDINT;
  newAttr : SXmlAttribute;
END_VAR

Implementation range

The implementation section shows various options for creating an XML document.
IF bCreate THEN
    (* Create an empty XML document *)
    objRoot := fbXml.GetDocumentNode();

    (* Create a new XML node 'Machines' and add to the empty document *)
    objMachines := fbXml.AppendNode(objRoot, 'Machines');

    (* Create a new XML node 'Machine' and add an attribute to this node. Append node to 'Machines' *)
    objMachine := fbXml.AppendNode(objMachines, 'Machine');
    objAttribute := fbXml.AppendAttribute(objMachine, 'Name', 'Wilde Nelli');

    (* Create a new XML node 'Controllers' and add to the 'Machine' node *)
    objControllers := fbXml.AppendNode(objMachine, 'Controllers');

    (* Create a new XML node 'Controller' and add some attributes. Append node to 'Controllers'. *)
    objController := fbXml.AppendChild(objControllers, 'Controller', 'CX5120', FALSE);
    objAttribute := fbXml.AppendAttribute(objController, 'Type', 'EPC');
    objAttribute := fbXml.AppendAttribute(objController, 'OS', 'Windows Embedded Compact 7');

    (* Create a new XML node 'Controller' and add some attributes. Append node to 'Controllers'. *)
    objController := fbXml.AppendChild(objControllers, 'Controller', 'CX2040', FALSE);
    objAttribute := fbXml.AppendAttribute(objController, 'Type', 'EPC');
    objAttribute := fbXml.AppendAttribute(objController, 'OS', 'Windows Embedded Standard 7');

    (* Create a new XML node 'Controller' and add some attributes. Append node to 'Controllers'. *)
    objController := fbXml.AppendChild(objControllers, 'Controller', 'C6015', FALSE);
    objAttribute := fbXml.AppendAttribute(objController, 'Type', 'IPC');
    objAttribute := fbXml.AppendAttribute(objController, 'OS', 'Windows 10 IoT Enterprise');

    (* Create a new XML node 'Machine' and add an attribute to this node. Append node to 'Machines' *)
    objMachine := fbXml.AppendNode(objMachines, 'Machine');
    objAttribute := fbXml.AppendAttribute(objMachine, 'Name', 'Stanze Oscar');

    (* Create a new XML node 'Controllers' and add to the 'Machine' node *)
    objControllers := fbXml.AppendNode(objMachine, 'Controllers');

    (* Create a new XML node 'Controller' and add some attributes. Append node to 'Controllers'. *)
    objController := fbXml.AppendChild(objControllers, 'Controller', 'C6017', FALSE);
    objAttribute := fbXml.AppendAttribute(objController, 'Type', 'IPC');
    objAttribute := fbXml.AppendAttribute(objController, 'OS', 'Windows 10 IoT Enterprise');
    newAttr := fbXml.InsertAttribute(objController, objAttribute, 'AddAttribute');
    fbXml.SetAttribute(newAttr, 'Hola');

    (* Retrieve XML document and store in a variable of data type STRING(1000) *)
    nLength := fbXml.CopyDocument(sXmlString, SIZEOF(sXmlString));
    bCreate := FALSE;
END_IF
7 Error Codes

7.1 ADS Return Codes

Grouping of error codes:
Global error codes: 0x0000 [110]... (0x9811_0000 ...)
Router error codes: 0x0500 [110]... (0x9811_0500 ...)
General ADS errors: 0x0700 [111]... (0x9811_0700 ...)
RTime error codes: 0x1000 [113]... (0x9811_1000 ...)  

Global error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>0</td>
<td>0x9811_0000</td>
<td>ERR_NOERROR</td>
<td>No error.</td>
</tr>
<tr>
<td>0x1</td>
<td>1</td>
<td>0x9811_0001</td>
<td>ERR_INTERNAL</td>
<td>Internal error.</td>
</tr>
<tr>
<td>0x2</td>
<td>2</td>
<td>0x9811_0002</td>
<td>ERR_NORTIME</td>
<td>No real time.</td>
</tr>
<tr>
<td>0x3</td>
<td>3</td>
<td>0x9811_0003</td>
<td>ERR_ALLOCLOCKEDMEM</td>
<td>Allocation locked – memory error.</td>
</tr>
<tr>
<td>0x4</td>
<td>4</td>
<td>0x9811_0004</td>
<td>ERR_INSERTMAILBOX</td>
<td>Mailbox full – the ADS message could not be sent. Reducing the number of ADS messages per cycle will help.</td>
</tr>
<tr>
<td>0x5</td>
<td>5</td>
<td>0x9811_0005</td>
<td>ERR_WRONGRECEIVEHMSG</td>
<td>Wrong HMSG.</td>
</tr>
<tr>
<td>0x6</td>
<td>6</td>
<td>0x9811_0006</td>
<td>ERR_TARGETPORTNOTFOUND</td>
<td>Target port not found – ADS server is not started or is not reachable.</td>
</tr>
<tr>
<td>0x7</td>
<td>7</td>
<td>0x9811_0007</td>
<td>ERR_TARGETMACHINENOTFOUND</td>
<td>Target computer not found – AMS route was not found.</td>
</tr>
<tr>
<td>0x8</td>
<td>8</td>
<td>0x9811_0008</td>
<td>ERR_UNKNOWNCOMMANDID</td>
<td>Unknown command ID.</td>
</tr>
<tr>
<td>0x9</td>
<td>9</td>
<td>0x9811_0009</td>
<td>ERR_BADTASKID</td>
<td>Invalid task ID.</td>
</tr>
<tr>
<td>0xA</td>
<td>10</td>
<td>0x9811_000A</td>
<td>ERR_NOIO</td>
<td>No IO.</td>
</tr>
<tr>
<td>0xB</td>
<td>11</td>
<td>0x9811_000B</td>
<td>ERR_UNKNOWNAMSCMD</td>
<td>Unknown AMS command.</td>
</tr>
<tr>
<td>0xC</td>
<td>12</td>
<td>0x9811_000C</td>
<td>ERR_WIN32ERROR</td>
<td>Win32 error.</td>
</tr>
<tr>
<td>0xD</td>
<td>13</td>
<td>0x9811_000D</td>
<td>ERR_PORTNOTCONNECTED</td>
<td>Port not connected.</td>
</tr>
<tr>
<td>0xE</td>
<td>14</td>
<td>0x9811_000E</td>
<td>ERR_INVALIDAMSLNGTH</td>
<td>Invalid AMS length.</td>
</tr>
<tr>
<td>0xF</td>
<td>15</td>
<td>0x9811_000F</td>
<td>ERR_INVALIDAMSNETID</td>
<td>Invalid AMS Net ID.</td>
</tr>
<tr>
<td>0x10</td>
<td>16</td>
<td>0x9811_0010</td>
<td>ERR_LOWINSTLEVEL</td>
<td>Installation level is too low – TwinCAT 2 license error.</td>
</tr>
<tr>
<td>0x11</td>
<td>17</td>
<td>0x9811_0011</td>
<td>ERR_NODEBUGINTAVAILABLE</td>
<td>No debugging available.</td>
</tr>
<tr>
<td>0x12</td>
<td>18</td>
<td>0x9811_0012</td>
<td>ERR_PORTDISABLED</td>
<td>Port disabled – TwinCAT system service not started.</td>
</tr>
<tr>
<td>0x13</td>
<td>19</td>
<td>0x9811_0013</td>
<td>ERR_PORTALREADYCONNECTED</td>
<td>Port already connected.</td>
</tr>
<tr>
<td>0x14</td>
<td>20</td>
<td>0x9811_0014</td>
<td>ERR_AMSSYNC_W32ERROR</td>
<td>AMS Sync Win32 error.</td>
</tr>
<tr>
<td>0x15</td>
<td>21</td>
<td>0x9811_0015</td>
<td>ERR_AMSSYNC_TIMEOUT</td>
<td>AMS Sync Timeout.</td>
</tr>
<tr>
<td>0x16</td>
<td>22</td>
<td>0x9811_0016</td>
<td>ERR_AMSSYNC_AMSERROR</td>
<td>AMS Sync error.</td>
</tr>
<tr>
<td>0x17</td>
<td>23</td>
<td>0x9811_0017</td>
<td>ERR_AMSSYNC_NOINDEXINMAP</td>
<td>No index map for AMS Sync available.</td>
</tr>
<tr>
<td>0x18</td>
<td>24</td>
<td>0x9811_0018</td>
<td>ERR_INVALIDAMSPORT</td>
<td>Invalid AMS port.</td>
</tr>
<tr>
<td>0x19</td>
<td>25</td>
<td>0x9811_0019</td>
<td>ERR_NOMEMORY</td>
<td>No memory.</td>
</tr>
<tr>
<td>0x1A</td>
<td>26</td>
<td>0x9811_001A</td>
<td>ERR_TCPSEND</td>
<td>TCP send error.</td>
</tr>
<tr>
<td>0x1B</td>
<td>27</td>
<td>0x9811_001B</td>
<td>ERR_HOSTUNREACHABLE</td>
<td>Host unreachable.</td>
</tr>
<tr>
<td>0x1C</td>
<td>28</td>
<td>0x9811_001C</td>
<td>ERR_INVALIDAMSFRAAGMENT</td>
<td>Invalid AMS fragment.</td>
</tr>
<tr>
<td>0x1D</td>
<td>29</td>
<td>0x9811_001D</td>
<td>ERR_TLSSEND</td>
<td>TLS send error – secure ADS connection failed.</td>
</tr>
<tr>
<td>0x1E</td>
<td>30</td>
<td>0x9811_001E</td>
<td>ERR_ACCESSDENIED</td>
<td>Access denied – secure ADS access denied.</td>
</tr>
</tbody>
</table>

Router error codes
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x500</td>
<td>1280</td>
<td>0x9811 0500</td>
<td>ROUTERR_NOLOCKEDMEMORY</td>
<td>Locked memory cannot be allocated.</td>
</tr>
<tr>
<td>0x501</td>
<td>1281</td>
<td>0x9811 0501</td>
<td>ROUTERR_RESIZEMEMORY</td>
<td>The router memory size could not be changed.</td>
</tr>
<tr>
<td>0x502</td>
<td>1282</td>
<td>0x9811 0502</td>
<td>ROUTERR_MAILBOXFULL</td>
<td>The mailbox has reached the maximum number of possible messages.</td>
</tr>
<tr>
<td>0x503</td>
<td>1283</td>
<td>0x9811 0503</td>
<td>ROUTERR_DEBUGBOXFULL</td>
<td>The Debug mailbox has reached the maximum number of possible messages.</td>
</tr>
<tr>
<td>0x504</td>
<td>1284</td>
<td>0x9811 0504</td>
<td>ROUTERR_UNKNOWNPORTTYPE</td>
<td>The port type is unknown.</td>
</tr>
<tr>
<td>0x505</td>
<td>1285</td>
<td>0x9811 0505</td>
<td>ROUTERR_NOTINITIALIZED</td>
<td>The router is not initialized.</td>
</tr>
<tr>
<td>0x506</td>
<td>1286</td>
<td>0x9811 0506</td>
<td>ROUTERR_PORTALREADYINUSE</td>
<td>The port number is already assigned.</td>
</tr>
<tr>
<td>0x507</td>
<td>1287</td>
<td>0x9811 0507</td>
<td>ROUTERR_NOTREGISTERED</td>
<td>The port is not registered.</td>
</tr>
<tr>
<td>0x508</td>
<td>1288</td>
<td>0x9811 0508</td>
<td>ROUTERR_NOREQUEUES</td>
<td>The maximum number of ports has been reached.</td>
</tr>
<tr>
<td>0x509</td>
<td>1289</td>
<td>0x9811 0509</td>
<td>ROUTERR_INVALIDPORT</td>
<td>The port is invalid.</td>
</tr>
<tr>
<td>0x50A</td>
<td>1290</td>
<td>0x9811 050A</td>
<td>ROUTERR_NOTACTIVATED</td>
<td>The router is not active.</td>
</tr>
<tr>
<td>0x50B</td>
<td>1291</td>
<td>0x9811 050B</td>
<td>ROUTERR_FRAGMENTBOXFULL</td>
<td>The mailbox has reached the maximum number for fragmented messages.</td>
</tr>
<tr>
<td>0x50C</td>
<td>1292</td>
<td>0x9811 050C</td>
<td>ROUTERR_FRAGMENTTIMEOUT</td>
<td>A fragment timeout has occurred.</td>
</tr>
<tr>
<td>0x50D</td>
<td>1293</td>
<td>0x9811 050D</td>
<td>ROUTERR_TOBEREMOVED</td>
<td>The port is removed.</td>
</tr>
</tbody>
</table>

General ADS error codes
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x70</td>
<td>1792</td>
<td>0x9811 0700</td>
<td>ADSERRDEVICE_ERROR</td>
<td>General device error.</td>
</tr>
<tr>
<td>0x70</td>
<td>1793</td>
<td>0x9811 0701</td>
<td>ADSERRDEVICE_SRVNOTSUPP</td>
<td>Service is not supported by the server.</td>
</tr>
<tr>
<td>0x70</td>
<td>1794</td>
<td>0x9811 0702</td>
<td>ADSERRDEVICE_INVALIDGRP</td>
<td>Invalid index group.</td>
</tr>
<tr>
<td>0x70</td>
<td>1795</td>
<td>0x9811 0703</td>
<td>ADSERRDEVICE_INVALIDOFFSET</td>
<td>Invalid index offset.</td>
</tr>
<tr>
<td>0x70</td>
<td>1796</td>
<td>0x9811 0704</td>
<td>ADSERRDEVICE_INVALIDACCESS</td>
<td>Reading or writing not permitted.</td>
</tr>
<tr>
<td>0x70</td>
<td>1797</td>
<td>0x9811 0705</td>
<td>ADSERRDEVICE_INVALIDSIZE</td>
<td>Parameter size not correct.</td>
</tr>
<tr>
<td>0x70</td>
<td>1798</td>
<td>0x9811 0706</td>
<td>ADSERRDEVICE_INVALIDDATA</td>
<td>Invalid data values.</td>
</tr>
<tr>
<td>0x70</td>
<td>1799</td>
<td>0x9811 0707</td>
<td>ADSERRDEVICE_NOTREADY</td>
<td>Device is not ready to operate.</td>
</tr>
<tr>
<td>0x70</td>
<td>1800</td>
<td>0x9811 0708</td>
<td>ADSERRDEVICE_BUSY</td>
<td>Device is busy.</td>
</tr>
<tr>
<td>0x70</td>
<td>1801</td>
<td>0x9811 0709</td>
<td>ADSERRDEVICE_INVALIDCONTEXT</td>
<td>Invalid operating system context. This can result from use of ADS blocks in different tasks. It may be possible to resolve this through multitasking synchronization in the PLC.</td>
</tr>
<tr>
<td>0x70A</td>
<td>1802</td>
<td>0x9811 070A</td>
<td>ADSERRDEVICE_NOMEMORY</td>
<td>Insufficient memory.</td>
</tr>
<tr>
<td>0x70B</td>
<td>1803</td>
<td>0x9811 070B</td>
<td>ADSERRDEVICE_INVALIDPARAM</td>
<td>Invalid parameter values.</td>
</tr>
<tr>
<td>0x70C</td>
<td>1804</td>
<td>0x9811 070C</td>
<td>ADSERRDEVICE_NOTFOUND</td>
<td>Not found (files,...).</td>
</tr>
<tr>
<td>0x70D</td>
<td>1805</td>
<td>0x9811 070D</td>
<td>ADSERRDEVICE_SYNTAX</td>
<td>Syntax error in file or command.</td>
</tr>
<tr>
<td>0x70E</td>
<td>1806</td>
<td>0x9811 070E</td>
<td>ADSERRDEVICE_INCOMPATIBLE</td>
<td>Objects do not match.</td>
</tr>
<tr>
<td>0x70F</td>
<td>1807</td>
<td>0x9811 070F</td>
<td>ADSERRDEVICE_EXISTS</td>
<td>Object already exists.</td>
</tr>
<tr>
<td>0x710</td>
<td>1808</td>
<td>0x9811 0710</td>
<td>ADSERRDEVICE_SYMBOLNOTFOUND</td>
<td>Symbol not found.</td>
</tr>
<tr>
<td>0x711</td>
<td>1809</td>
<td>0x9811 0711</td>
<td>ADSERRDEVICE_SYMBOLVERSIONINVALID</td>
<td>Invalid symbol version. This can occur due to an online change. Create a new handle.</td>
</tr>
<tr>
<td>0x712</td>
<td>1810</td>
<td>0x9811 0712</td>
<td>ADSERRDEVICE_INVALIDSTATE</td>
<td>Device (server) is in invalid state.</td>
</tr>
<tr>
<td>0x713</td>
<td>1811</td>
<td>0x9811 0713</td>
<td>ADSERRDEVICE_TRANSMODENOTSUPP</td>
<td>AdsTransMode not supported.</td>
</tr>
<tr>
<td>0x714</td>
<td>1812</td>
<td>0x9811 0714</td>
<td>ADSERRDEVICE_NOTIFYHINDINVALID</td>
<td>Notification handle is invalid.</td>
</tr>
<tr>
<td>0x715</td>
<td>1813</td>
<td>0x9811 0715</td>
<td>ADSERRDEVICE_CLIENTUNKNOWN</td>
<td>Notification client not registered.</td>
</tr>
<tr>
<td>0x716</td>
<td>1814</td>
<td>0x9811 0716</td>
<td>ADSERRDEVICE_NOMOREHDLSD</td>
<td>No further handle available.</td>
</tr>
<tr>
<td>0x717</td>
<td>1815</td>
<td>0x9811 0717</td>
<td>ADSERRDEVICE_INVALIDWATCHSIZE</td>
<td>Notification size too large.</td>
</tr>
<tr>
<td>0x718</td>
<td>1816</td>
<td>0x9811 0718</td>
<td>ADSERRDEVICE_NOTINIT</td>
<td>Device not initialized.</td>
</tr>
<tr>
<td>0x719</td>
<td>1817</td>
<td>0x9811 0719</td>
<td>ADSERRDEVICE_TIMEOUT</td>
<td>Device has a timeout.</td>
</tr>
<tr>
<td>0x71A</td>
<td>1818</td>
<td>0x9811 071A</td>
<td>ADSERRDEVICE_Nointerface</td>
<td>Interface query failed.</td>
</tr>
<tr>
<td>0x71B</td>
<td>1819</td>
<td>0x9811 071B</td>
<td>ADSERRDEVICE_INVALIDINTERFACE</td>
<td>Wrong interface requested.</td>
</tr>
<tr>
<td>0x71C</td>
<td>1820</td>
<td>0x9811 071C</td>
<td>ADSERRDEVICE_INVALIDCLSID</td>
<td>Class ID is invalid.</td>
</tr>
<tr>
<td>0x71D</td>
<td>1821</td>
<td>0x9811 071D</td>
<td>ADSERRDEVICE_INVALIDOBJID</td>
<td>Object ID is invalid.</td>
</tr>
<tr>
<td>0x71E</td>
<td>1822</td>
<td>0x9811 071E</td>
<td>ADSERRDEVICE_PENDING</td>
<td>Request pending.</td>
</tr>
<tr>
<td>0x71F</td>
<td>1823</td>
<td>0x9811 071F</td>
<td>ADSERRDEVICE_ABORTED</td>
<td>Request is aborted.</td>
</tr>
<tr>
<td>0x720</td>
<td>1824</td>
<td>0x9811 0720</td>
<td>ADSERRDEVICE_WARNING</td>
<td>Signal warning.</td>
</tr>
<tr>
<td>0x721</td>
<td>1825</td>
<td>0x9811 0721</td>
<td>ADSERRDEVICE_INVALIDARRAYIDX</td>
<td>Invalid array index.</td>
</tr>
<tr>
<td>0x722</td>
<td>1826</td>
<td>0x9811 0722</td>
<td>ADSERRDEVICE_SYMBOLNOTACTIVE</td>
<td>Symbol not active.</td>
</tr>
<tr>
<td>0x723</td>
<td>1827</td>
<td>0x9811 0723</td>
<td>ADSERRDEVICE_ACCESSDENIED</td>
<td>Access denied.</td>
</tr>
<tr>
<td>0x724</td>
<td>1828</td>
<td>0x9811 0724</td>
<td>ADSERRDEVICE_LICENSENOTFOUND</td>
<td>Missing license.</td>
</tr>
<tr>
<td>0x725</td>
<td>1829</td>
<td>0x9811 0725</td>
<td>ADSERRDEVICE_LICENSEEXPIRED</td>
<td>License expired.</td>
</tr>
<tr>
<td>0x726</td>
<td>1830</td>
<td>0x9811 0726</td>
<td>ADSERRDEVICE_LICENSEEXCEEDED</td>
<td>License exceeded.</td>
</tr>
<tr>
<td>0x727</td>
<td>1831</td>
<td>0x9811 0727</td>
<td>ADSERRDEVICE_LICENSEINVALID</td>
<td>Invalid license.</td>
</tr>
<tr>
<td>0x728</td>
<td>1832</td>
<td>0x9811 0728</td>
<td>ADSERRDEVICE_LICENSESYSTEMID</td>
<td>License problem: System ID is invalid.</td>
</tr>
<tr>
<td>0x729</td>
<td>1833</td>
<td>0x9811 0729</td>
<td>ADSERRDEVICE_LICENSENOMETLIMIT</td>
<td>License not limited in time.</td>
</tr>
<tr>
<td>0x72A</td>
<td>1834</td>
<td>0x9811 072A</td>
<td>ADSERRDEVICE_LICENSEFUTUREISSUE</td>
<td>Licensing problem: time in the future.</td>
</tr>
<tr>
<td>0x72B</td>
<td>1835</td>
<td>0x9811 072B</td>
<td>ADSERRDEVICE_LICENSESETIMETOLONG</td>
<td>License period too long.</td>
</tr>
<tr>
<td>0x72C</td>
<td>1836</td>
<td>0x9811 072C</td>
<td>ADSERRDEVICE_EXCEPTION</td>
<td>Exception at system startup.</td>
</tr>
<tr>
<td>0x72D</td>
<td>1837</td>
<td>0x9811 072D</td>
<td>ADSERRDEVICE_LICENSEDEPRECATED</td>
<td>License file read twice.</td>
</tr>
<tr>
<td>0x72E</td>
<td>1838</td>
<td>0x9811 072E</td>
<td>ADSERRDEVICE_SIGNATUREINVALID</td>
<td>Invalid signature.</td>
</tr>
<tr>
<td>0x72F</td>
<td>1839</td>
<td>0x9811 072F</td>
<td>ADSERRDEVICE_CERTIFICATEINVALID</td>
<td>Invalid certificate.</td>
</tr>
<tr>
<td>0x730</td>
<td>1840</td>
<td>0x9811 0730</td>
<td>ADSERRDEVICE_LICENSESEOMNOTFOUND</td>
<td>Public key not known from OEM.</td>
</tr>
<tr>
<td>0x731</td>
<td>1841</td>
<td>0x9811 0731</td>
<td>ADSERRDEVICE_LICENSERESTRICTED</td>
<td>License not valid for this system ID.</td>
</tr>
<tr>
<td>0x732</td>
<td>1842</td>
<td>0x9811 0732</td>
<td>ADSERRDEVICE_LICENSEDEMOVEDENIED</td>
<td>Demo license prohibited.</td>
</tr>
<tr>
<td>0x733</td>
<td>1843</td>
<td>0x9811 0733</td>
<td>ADSERRDEVICE_LICENSEFNCID</td>
<td>Invalid function ID.</td>
</tr>
<tr>
<td>0x734</td>
<td>1844</td>
<td>0x9811 0734</td>
<td>ADSERRDEVICE_OUTOFRANGE</td>
<td>Outside the valid range.</td>
</tr>
<tr>
<td>0x735</td>
<td>1845</td>
<td>0x9811 0735</td>
<td>ADSERRDEVICE_INVALIDALIGNMENT</td>
<td>Invalid alignment.</td>
</tr>
</tbody>
</table>
### Error Codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x736</td>
<td>1846</td>
<td>0x9811 0736</td>
<td>ADSERR_DEVICE_LICENSEPLATFORM</td>
<td>Invalid platform level.</td>
</tr>
<tr>
<td>0x737</td>
<td>1847</td>
<td>0x9811 0737</td>
<td>ADSERR_DEVICE_FORWARD_PL</td>
<td>Context – forward to passive level.</td>
</tr>
<tr>
<td>0x738</td>
<td>1848</td>
<td>0x9811 0738</td>
<td>ADSERRDEVICE_FORWARD_DL</td>
<td>Context – forward to dispatch level.</td>
</tr>
<tr>
<td>0x739</td>
<td>1849</td>
<td>0x9811 0739</td>
<td>ADSERRDEVICE_FORWARD_RT</td>
<td>Context – forward to real time.</td>
</tr>
<tr>
<td>0x740</td>
<td>1850</td>
<td>0x9811 0740</td>
<td>ADSERRCLIENT_ERROR</td>
<td>Client error.</td>
</tr>
<tr>
<td>0x741</td>
<td>1851</td>
<td>0x9811 0741</td>
<td>ADSERR_CLIENT_INVALIDPARM</td>
<td>Service contains an invalid parameter.</td>
</tr>
<tr>
<td>0x742</td>
<td>1852</td>
<td>0x9811 0742</td>
<td>ADSERR_CLIENT_LISTEMPTY</td>
<td>Polling list is empty.</td>
</tr>
<tr>
<td>0x743</td>
<td>1853</td>
<td>0x9811 0743</td>
<td>ADSERR_CLIENT_VARUSED</td>
<td>Var connection already in use.</td>
</tr>
<tr>
<td>0x744</td>
<td>1854</td>
<td>0x9811 0744</td>
<td>ADSERR_CLIENT_DUPLINVOKED</td>
<td>The called ID is already in use.</td>
</tr>
<tr>
<td>0x745</td>
<td>1855</td>
<td>0x9811 0745</td>
<td>ADSERR_CLIENT_SYNCTIMEOUT</td>
<td>Timeout has occurred – the remote terminal is not responding in the specified ADS timeout. The route setting of the remote terminal may be configured incorrectly.</td>
</tr>
<tr>
<td>0x746</td>
<td>1856</td>
<td>0x9811 0746</td>
<td>ADSERR_CLIENT_W32ERROR</td>
<td>Error in Win32 subsystem.</td>
</tr>
<tr>
<td>0x747</td>
<td>1857</td>
<td>0x9811 0747</td>
<td>ADSERR_CLIENT_TIMEOUTINVALID</td>
<td>Invalid client timeout value.</td>
</tr>
<tr>
<td>0x748</td>
<td>1858</td>
<td>0x9811 0748</td>
<td>ADSERR_CLIENT_PORTNOTOPEN</td>
<td>Port not open.</td>
</tr>
<tr>
<td>0x749</td>
<td>1859</td>
<td>0x9811 0749</td>
<td>ADSERR_CLIENT_NOAMSADDR</td>
<td>No AMS address.</td>
</tr>
<tr>
<td>0x750</td>
<td>1860</td>
<td>0x9811 0750</td>
<td>ADSERR_CLIENT_SYNCINTERNAL</td>
<td>Internal error in Ads sync.</td>
</tr>
<tr>
<td>0x751</td>
<td>1861</td>
<td>0x9811 0751</td>
<td>ADSERR_CLIENT_ADDHASH</td>
<td>Hash table overflow.</td>
</tr>
<tr>
<td>0x752</td>
<td>1862</td>
<td>0x9811 0752</td>
<td>ADSERR_CLIENT_REMOVEHASH</td>
<td>Key not found in the table.</td>
</tr>
<tr>
<td>0x753</td>
<td>1863</td>
<td>0x9811 0753</td>
<td>ADSERR_CLIENT_NOMORESYM</td>
<td>No symbols in the cache.</td>
</tr>
<tr>
<td>0x754</td>
<td>1864</td>
<td>0x9811 0754</td>
<td>ADSERR_CLIENT_SYNCRESINVALID</td>
<td>Invalid response received.</td>
</tr>
<tr>
<td>0x755</td>
<td>1865</td>
<td>0x9811 0755</td>
<td>ADSERR_CLIENT_SYNCPORTLOCKED</td>
<td>Sync Port is locked.</td>
</tr>
</tbody>
</table>

### RTime error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>4096</td>
<td>0x9811 1000</td>
<td>RTERR_INTERNAL</td>
<td>Internal error in the real-time system.</td>
</tr>
<tr>
<td>0x1001</td>
<td>4097</td>
<td>0x9811 1001</td>
<td>RTERR_BADTIMERPERIODS</td>
<td>Timer value is not valid.</td>
</tr>
<tr>
<td>0x1002</td>
<td>4098</td>
<td>0x9811 1002</td>
<td>RTERR_INVALIDASKPTR</td>
<td>Task pointer has the invalid value 0 (zero).</td>
</tr>
<tr>
<td>0x1003</td>
<td>4099</td>
<td>0x9811 1003</td>
<td>RTERR_INVALIDSTACKPTR</td>
<td>Stack pointer has the invalid value 0 (zero).</td>
</tr>
<tr>
<td>0x1004</td>
<td>4100</td>
<td>0x9811 1004</td>
<td>RTERR_PRIEXISTS</td>
<td>The request task priority is already assigned.</td>
</tr>
<tr>
<td>0x1005</td>
<td>4101</td>
<td>0x9811 1005</td>
<td>RTERR_NOMORETCB</td>
<td>No free TCB (Task Control Block) available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The maximum number of TCBs is 64.</td>
</tr>
<tr>
<td>0x1006</td>
<td>4102</td>
<td>0x9811 1006</td>
<td>RTERR_NOMORESEMAS</td>
<td>No free semaphores available. The maximum number of semaphores is 64.</td>
</tr>
<tr>
<td>0x1007</td>
<td>4103</td>
<td>0x9811 1007</td>
<td>RTERR_NOMOREQUEUES</td>
<td>No free space available in the queue. The maximum number of positions in the queue is 64.</td>
</tr>
<tr>
<td>0x100D</td>
<td>4109</td>
<td>0x9811 100D</td>
<td>RTERR_EXTERNQALREADYDEF</td>
<td>An external synchronization interrupt is already applied.</td>
</tr>
<tr>
<td>0x100E</td>
<td>4110</td>
<td>0x9811 100E</td>
<td>RTERR_EXTERNQNOTDEF</td>
<td>No external sync interrupt applied.</td>
</tr>
<tr>
<td>0x100F</td>
<td>4111</td>
<td>0x9811 100F</td>
<td>RTERR_EXTERNQINSTALLFAILED</td>
<td>Application of the external synchronization interrupt has failed.</td>
</tr>
<tr>
<td>0x1010</td>
<td>4112</td>
<td>0x9811 1010</td>
<td>RTERR_IRQLNOTLESSOREQUAL</td>
<td>Call of a service function in the wrong context</td>
</tr>
<tr>
<td>0x1017</td>
<td>4119</td>
<td>0x9811 1017</td>
<td>RTERR_VMXTNOTSUPPORTED</td>
<td>Intel VT-x extension is not supported.</td>
</tr>
<tr>
<td>0x1018</td>
<td>4120</td>
<td>0x9811 1018</td>
<td>RTERR_VMXDISABLED</td>
<td>Intel VT-x extension is not enabled in the BIOS.</td>
</tr>
<tr>
<td>0x1019</td>
<td>4121</td>
<td>0x9811 1019</td>
<td>RTERR_VMXCONTROLSMISSING</td>
<td>Missing function in Intel VT-x extension.</td>
</tr>
<tr>
<td>0x101A</td>
<td>4122</td>
<td>0x9811 101A</td>
<td>RTERR_VMXENABLEFAILS</td>
<td>Activation of Intel VT-x fails.</td>
</tr>
</tbody>
</table>

### Specific positive HRESULT Return Codes:

<table>
<thead>
<tr>
<th>HRESULT</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000_0000</td>
<td>S_OK</td>
<td>No error.</td>
</tr>
<tr>
<td>0x0000_0001</td>
<td>S_FALSE</td>
<td>No error. Example: successful processing, but with a negative or incomplete result.</td>
</tr>
<tr>
<td>0x0000_0203</td>
<td>S_PENDING</td>
<td>No error. Example: successful processing, but no result is available yet.</td>
</tr>
<tr>
<td>0x0000_0256</td>
<td>S_WATCHDOG_TIMEOUT</td>
<td>No error. Example: successful processing, but a timeout occurred.</td>
</tr>
</tbody>
</table>
## TCP Winsock error codes

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x274C</td>
<td>10060</td>
<td>WSAETIMEDOUT</td>
<td>A connection timeout has occurred - error while establishing the connection, because the remote terminal did not respond properly after a certain period of time, or the established connection could not be maintained because the connected host did not respond.</td>
</tr>
<tr>
<td>0x274D</td>
<td>10061</td>
<td>WSAECONNREFUSED</td>
<td>Connection refused - no connection could be established because the target computer has explicitly rejected it. This error usually results from an attempt to connect to a service that is inactive on the external host, that is, a service for which no server application is running.</td>
</tr>
<tr>
<td>0x2751</td>
<td>10065</td>
<td>WSAEHOSTUNREACH</td>
<td>No route to host - a socket operation referred to an unavailable host.</td>
</tr>
</tbody>
</table>

More Winsock error codes: Win32 error codes
8 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Beckhoff’s branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff’s branch offices and representatives round the world can be found on her internet pages: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963 157
Fax: +49 5246 963 9157
e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963 460
Fax: +49 5246 963 479
e-mail: service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone: +49 5246 963 0
Fax: +49 5246 963 198
e-mail: info@beckhoff.com
web: https://www.beckhoff.com