

Unified-E Standard WebHttp Adapter

Version: 1.5.0.2 und höher
Juli 2017

Inhalt

1	Allgemeines.....	2
2	Adapter-Parameter in Unified-E.....	2
3	Symbolische Adressierung	3
3.1	IsAlive Methode.....	4
3.2	ReadValues-Methode.....	4
3.3	WriteValues Methode	6
3.4	ReadDataTables-Methode	7
3.5	Beispiel	9

1 Allgemeines

Dieser Adapter kommuniziert mit einem Endpunkt mit Hilfe des HTTP-Protokolls.

Der Endpunkt muss folgende Voraussetzungen erfüllen:

- Ein Web-Server muss installiert sein (bzw. http-Aufrufe müssen behandelt werden)
- GET-Methode (IsAlive) für Prüfung der Erreichbarkeit des Endpunktes
- Folgende PUT-Methoden werden wie folgt verwendet
 - Datenpunkte auslesen: ReadValues
 - Datenpunkte schreiben: WriteValues
 - Datenpunkt-Tabelle auslesen: ReadDataTables

Ein HTTP-Endpunkt, kann zum Beispiel

- Werte aus einem ERP auslesen
- Werte aus einer Datenbank auslesen
- Werte von der SPS via WebHttp auslesen, falls auf der SPS ein Web-Server vorhanden ist (anstatt OPC)

Eine Beschreibung, wie man die PUT-Methoden beim WebHttp-Endpunkt implementiert, finden Sie in Kapitel „WebHttp Endpunkt implementieren“.

2 Adapter-Parameter in Unified-E

Die vollständige Basis-Adresse eines Endpunktes ist typischerweise wie folgt:

`http://<Adresse>:<Port>/<Namespace>`, zum Beispiel <http://192.168.1.4:8085/Myplant>

Unter dieser Basis-Adresse müssen dann die oben genannten PUT-Methoden zu finden sein.

Adresse:

Die Adresse des Web-Server (Endpunktes), z. B. die IP-Adresse, URL oder localhost.

Port:

Der Port an dem der WebHttp Endpunkt lauscht.

Namespace:

Der Adresszusatz.

Timeout:

Der Timeout-Wert beim Aufruf der PUT-Methoden.

Protocol:

Wählen Sie das Protokoll, http oder https.

Datapoint context:

Folgende Datenpunkt-Kontexte werden unterstützt.

Global:

Alle Benutzer bekommen beim Anfragen eines bestimmten Datenpunktes denselben Wert geliefert. Dies ist üblich bei Datenpunkten, die Maschinenzustände repräsentieren.

User:

Unterschiedliche Benutzer können beim Anfragen eines bestimmten Datenpunktes unterschiedliche Werte erhalten. Beispiele:

- Diagramm- oder Listenanzeige: Benutzer wollen nur ihren eingestellten Bereich sehen, der ebenfalls über benutzerspezifische Datenpunkte konfiguriert ist.
- Aktuelle Produktionsaufträge des App-Benutzers.
- Leistungskennzahlen des App-Benutzers.

Der Ap-Benutzer einer Anfrage wird in den JSON-Anfragen (siehe unten) im Feld "User" mitgeführt.

Language:

Alle Benutzer einer Sprache erhalten denselben Datenpunkt-Wert für einen bestimmten Datenpunkt. Dieser Kontext macht dann Sinn, wenn Datenpunkte vom Typ „Text“ mehrsprachig sein sollen.

Authentication:

None: Der Web-Endpunkt erfordert keine Authentifizierung.

Custom authentication: Die Authentifizierung erfolgt bei jedem http-Aufruf und ist Unified-E-spezifisch (siehe unten).

Basic authentication: Das Standard-Protokoll "Http Basic Auth" wird benutzt.

User name:

Der Benutzername ist einzutragen. Dieses Feld ist nur sichtbar, wenn eine Authentifizierung gewählt wurde.

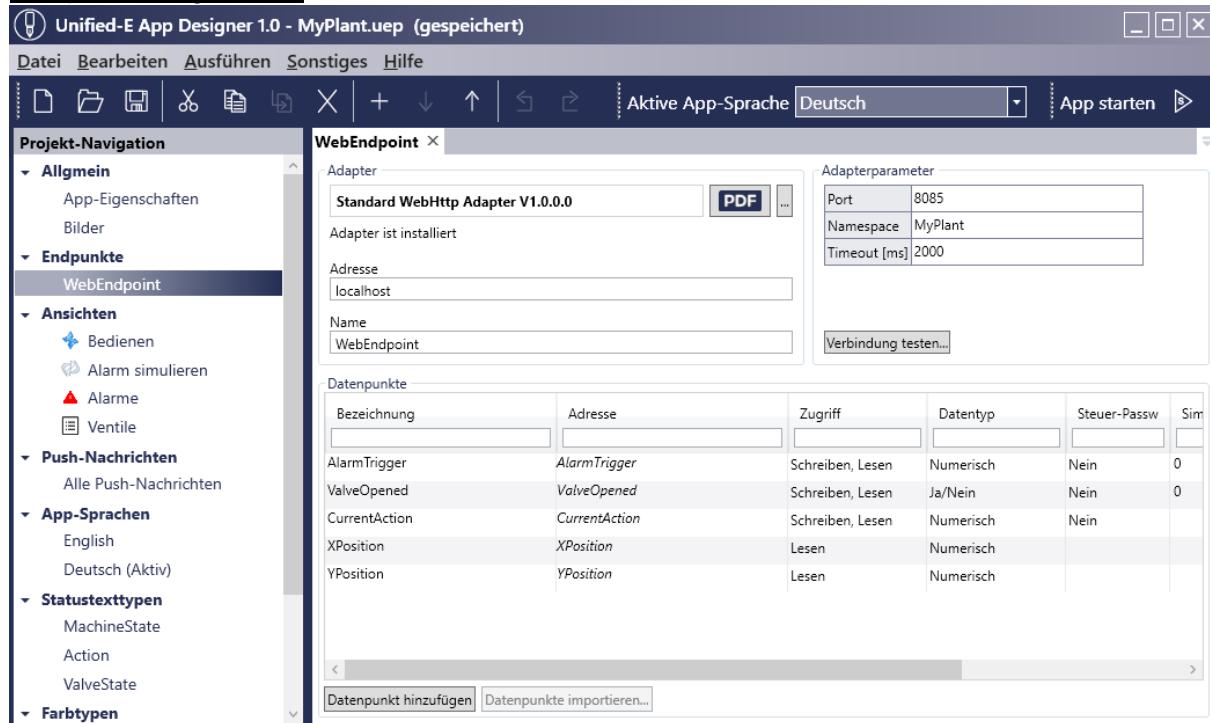
Password:

Das Passwort für die Authentifizierung. Dieses Feld ist nur sichtbar, wenn eine Authentifizierung gewählt wurde.

3 Symbolische Adressierung

Grundsätzlich erfolgt der Zugriff auf Variablen symbolisch. Wird keine Adresse beim Datenpunkt gesetzt, dann wird für die Adressierung die Datenpunkt-Bezeichnung verwendet.

Beispiel-Konfiguration:



WebHttp-Endpunkt implementieren

Es sind PUT-Methoden zu implementieren, die Kommunikation erfolgt mit JSON-Paketen. Die Syntax der Pakete wird im Folgenden mit C#-Syntax beschrieben.

Anmeldung mit "Custom Authentication":

Bei dieser Authentifizierung werden bei jedem Request die Felder "EndPointUser" und "EndPointPassword" so gesetzt wie bei den Endpunkt-Parametern "User name" und "Password" angegeben.

3.1 IsAlive-Methode

Der GET-Aufruf wird ohne Parameter aufgerufen und wird für die Überprüfung der grundsätzlichen Erreichbarkeit eines Endpunktes verwendet. Als Rückgabewert wird "True" erwartet.

3.2 ReadValues-Methode

Allgemein:

```
DataPointValueResult[] ReadValues(EndPointReadValuesRequest request);
```

Request:

```
/// <summary>
/// ReadValues: Request
/// </summary>
public class EndPointReadValuesRequest
{
    /// <summary>
    /// The user (only provided on custom authentication).
    /// </summary>
    public string EndPointUser { get; set; }
```

```
/// <summary>
/// The password (only provided on custom authentication).
/// </summary>
public string EndPointPassword { get; set; }

/// <summary>
/// The app user.
/// </summary>
public string User { get; set; }

/// <summary>
/// The language code (e.g. en, de) in 2-letter format.
/// </summary>
public string LanguageId { get; set; }

/// <summary>
/// Array of EndPointDataPoint objects.
/// </summary>
public EndPointDataPoint[] DataPoints { get; set; }
}

public class EndPointDataPoint
{
    /// <summary>
    /// Name of datapoint
    /// </summary>
    public string DataPointAddress { get; set; }
}
```

Response:

Hinweis: Die ReadValues-Methode gibt ein Array von DataPointValueResult-Objekten zurück.

```
/// <summary>
/// Returns the current value of a datapoint.
/// </summary>
public class DataPointValueResult
{
    /// <summary>
    /// The current value. Numeric values must always be returned in EN culture.
    /// For yes/no types return "1" or "0".
    /// </summary>
    public string Value { get; set; }

    /// <summary>
    /// The access state.
    /// </summary>
    public DataPointAccessState AccessState { get; set; }
}

public enum DataPointAccessState
{
    /// <summary>
    /// The value is valid.
    /// </summary>
    Valid = 0,

    /// <summary>
    /// Datapoint unknown.
    /// </summary>
    DataPointNotFound = 1,
```

```
/// <summary>
/// Other error.
/// </summary>
Error = 2
}
```

3.3 WriteValues Methode

Allgemein:

```
bool WriteValues(EndPointWriteValuesRequest request);
```

Request:

```
/// <summary>
/// Request to write multiple datapoints.
/// </summary>
public class EndPointWriteValuesRequest
{
    /// <summary>
    /// The user (only provided on custom authentication).
    /// </summary>
    public string EndPointUser { get; set; }

    /// <summary>
    /// The password (only provided on custom authentication).
    /// </summary>
    public string EndPointPassword { get; set; }

    /// <summary>
    /// The app user.
    /// </summary>
    public string User { get; set; }

    /// <summary>
    /// The language id.
    /// </summary>
    public string LanguageId { get; set; }

    /// <summary>
    /// The datapoint values to be written.
    /// </summary>
    public NamedDataPointValue[] DataPointValues { get; set; }
}

/// <summary>
/// Holds the new value for a datapoint.
/// </summary>
public class NamedDataPointValue
{
    /// <summary>
    /// Describes the datapoint.
    /// </summary>
    public EndPointDataPoint DataPoint { get; set; }

    /// <summary>
    /// The value to be written.
    /// </summary>
    public string Value { get; set; }
}

public class EndPointDataPoint
```

```
/// <summary>
/// Name of datapoint
/// </summary>
public string DataPointAddress { get; set; }
}
```

Response:

„True“, wenn erfolgreich, sonst „False“.

3.4 ReadDataTables-Methode

Allgemein:

```
EndPointDataTableValueResult[] ReadDataTables(EndPointReadDataTablesRequest request);
```

Request:

```
public class EndPointReadDataTablesRequest
{
    /// <summary>
    /// The user (only provided on custom authentication).
    /// </summary>
    public string EndPointUser { get; set; }

    /// <summary>
    /// The password (only provided on custom authentication).
    /// </summary>
    public string Password { get; set; }

    /// <summary>
    /// The app user.
    /// </summary>
    public string User { get; set; }

    /// <summary>
    /// The language code (e.g. en, de) in 2-letter format.
    /// </summary>
    public string LanguageId { get; set; }

    /// <summary>
    /// All datapoints of the requested datatables.
    /// </summary>
    public EndPointDataTableDataPoint[] DataPoints { get; set; }
}

public class EndPointDataTableDataPoint
{
    /// <summary>
    /// Symbolic name or address, depending on DataPointIdentificationType.
    /// </summary>
    public string DataPointAddress { get; set; }

    /// <summary>
    /// If set, the request will be optimized. If the current modification Id
    /// equals to the id set in LastModificationId, no rows will be returned.
    /// If not set, all rows will be returned.
    /// </summary>
    public string LastModificationId { get; set; }
}
```

Response:

```
public class EndPointDataTableValueResult
{
```

```
    /// <summary>
    /// The datatable containing the result.
    /// </summary>
    public EndPointDataTable DataTable { get; set; }

    /// <summary>
    /// Describes what is contained in the datatable.
    /// </summary>
    public EndPointDataTableModificationType ModificationType { get; set; }

    /// <summary>
    /// The access state.
    /// </summary>
    public DataPointAccessState AccessState { get; set; }

    /// <summary>
    /// The current modification id.
    /// </summary>
    public string ModificationId { get; set; }
}

public class EndPointDataTable
{
    /// <summary>
    /// The datatable rows.
    /// </summary>
    public EndPointDataRow[] Rows { get; set; }
}

public class EndPointDataRow
{
    /// <summary>
    /// The column values of a row.
    /// </summary>
    public string[] Items { get; set; }
}

public enum DataPointAccessState
{
    /// <summary>
    /// The value is valid.
    /// </summary>
    Valid = 0,

    /// <summary>
    /// Datapoint unknown.
    /// </summary>
    DataPointNotFound = 1,

    /// <summary>
    /// Other error.
    /// </summary>
    Error = 2
}

public enum EndPointDataTableModificationType
{
    /// <summary>
    /// All rows are returned because of an unspecified modification or if the
    /// property ModificationId has not been set in the request call.
    /// </summary>
    Reset = 0,
```

```
    /// <summary>
    /// No modifications have been made since the last call.
    /// </summary>
    NoModifications = 1,
    /// <summary>
    /// New rows have been added since the last call.
    /// </summary>
    NewRowsOnTop = 2,
    /// <summary>
    /// New rows have been added since the last call.
    /// </summary>
    NewRowsOnBottom = 3
}
}
```

3.5 Beispiel

Es sollen die Werte der Datenpunkte „Temperatur1“ und „Temperatur2“ ausgelesen werden.

Request (JSON):

```
{
    "DataPoints": [
        {
            "DataPointAddress": "Temperatur1"
        },
        {
            "DataPointAddress": "Temperatur2"
        }
    ],
    "LanguageId": "EN"
}
```

Response (JSON):

```
[
    {
        "AccessState": 0,
        "Value": "22.1"
    },
    {
        "AccessState": 0,
        "Value": "24.0"
    }
]
```