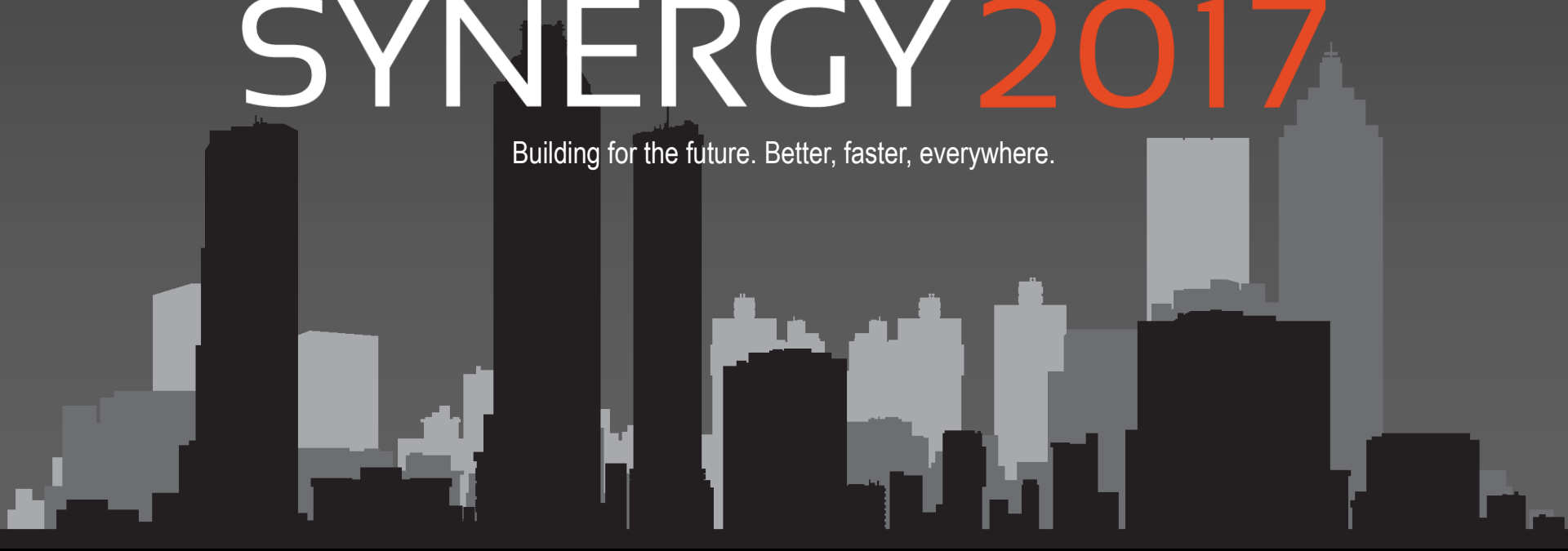
6 – 7 . APRIL ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.



Putting the DataFlex 19.0 JSON Parser to Work

Harm Wibier
Data Access Europe

Java**S**cript **O**bject **N**otation

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language,Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

The JSON Format

- Objects
- Arrays

- Strings
- Booleans
- Numbers

```
{
   "name" : "John",
   "details" : {
      "age" : 31,
      "male" : true
   },
   "ratings" : [ 8, 7.5, 8, 5.5 ]
}
```

JSON

```
{
   "name" : "John",
   "details" : {
      "age" : 31,
      "male" : true,
   },
   "ratings" : [
      8,
      7.5,
      8,
      5.5
   ]
}
```

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<student>
   <name>John</name>
   <details>
      <age>31</age>
      <male>true</male>
   </details>
   <ratings>
      <rate>8</rate>
      <rate>7.5</rate>
      <rate>8</rate>
      <rate>5.5</rate>
   </ratings>
</student>
```

JSON

- Human readable
- Hierachical
- Quicker
  - Shorter
    - No end tags
  - Easier to parse
    - Can be evaluated in some languges
- Has arrays

- Lighter and native to JavaScript

XML

- Human readable
- Hierachical
- Better standardized
- More extensive
  - Attributes
  - Namespaces
  - XML Schema
  - XSL
  - XPath

- Heavier but wider supported

Usage

- Interchange data
  - Restfull JSON API’s
- Store data
  - Serialize data from memory

JSON in DataFlex

http://localhost/WebOrder_19/CustomerAndOrderInfo.wso

- Web Services
  - Built in JSON support
  - Every DataFlex Web-Service can be called using JSON
  - JSON parsing & generation happens in ISAPI handler
- cJsonObject
  - Manually parse & generate JSON
  - Serialize / deserialize structs and arrays
- cJsonHttpTransfer
  - Communicate with JSON Services

cJsonObject

- Parse JSON
- Generate JSON
- Manipulate JSON like a DOM structure

- Convert DataFlex structs and arrays into JSON
- Convert JSON into DataFlex structs and arrays

Parsing Example

```
Get Create (RefClass(cJsonObject)) to hoJson

Get ParseString of hoJson '{"name" : "John", "details" : {"age" : 31, "male" : true}}' to bSuccess
If (bSuccess) Begin
   Get Member of hoJson "details" to hoDetails

   Get MemberValue of hoDetails "age" to iAge

   Send Destroy of hoDetails
End
Else Begin
   Send ReportParseError of hoJson
End

Send Destroy of hoJson
```

JSON <> Struct

```
{
 "name": "Troy Umstead",
 "details": {
  "age": 20,
  "male": true
 },
 "ratings": [
  2.3,
  5.2,
  4.0,
  9.4
 ]
}
```

```
Struct tStudentDetail
   Integer age
   Boolean male
End_Struct

Struct tStudent
   String name
   tStudentDetail details
   Number[] ratings
End_Struct
```

JSON > Struct

```
tStudent student
Get Create (RefClass(cJsonObject)) to hoJson

Get ParseString of hoJson '{"name" : "John", "details" : {"age"
   : 31, "male" : true}, "ratings" : [ 8, 7.5, 8, 5.5 ]}' to bSuccess

If (bSuccess) Begin
   //  Convert JSON structure into struct
   Get JsonToDataType of hoJson to student
End
Else Begin
   Send ReportParseError of hoJson
End

Send Destroy of hoJson
```

```
Struct tStudentDetail
   Integer age
   Boolean male
End_Struct

Struct tStudent
   String name
   tStudentDetail details
   Number[] ratings
End_Struct
```

Struct > JSON

```
tStudent student

Move "John" to student.name
Move 31     to student.details.age
Move True   to student.details.male
Move 8      to student.ratings[0]
Move 7.5    to student.ratings[1]

Get Create (RefClass(cJsonObject)) to hoJson

//  Convert struct to JSON structure
Send DataTypeToJson of hoJson student

Set peWhiteSpace of hoJson to jpWhitespace_Spaced
Get Stringify of hoJson to sJson
```

```
Struct tStudentDetail
   Integer age
   Boolean male
End_Struct

Struct tStudent
   String name
   tStudentDetail details
   Number[] ratings
End_Struct
```

ParseUtf8 and StringifyUtf8

- UChar array as parameter / argument
  - No argument-size limitations
  - Supported by Read_Block, Write_Block, Set_Field_Value, Get_Field_Value, Field_Current_UCAValue
- Expected encoding is UTF-8
  - Default format when transmitting JSON object the web
- Convert manually if needed
  - ConvertUCharArray of cChartTranslate

```
UChar[] uData

Direct_Input "FileWithJsonData.json"
Read_Block uData -1
Close_Input

Get Create (RefClass(cJsonObject)) to hoJsonDom
Get ParseUtf8 of hoJsonDom uData to bParsed
```

cJsonHttpTransfer

- Easily communicate with Restfull JSON Services
- HTTP Verbs:
  - POST, GET, DELETE, PUT, PATCH
- Directly parses into cJsonObject

cJsonHttpTransfer

```
Object oJsonHttp is a cJsonHttpTransfer
End_Object

Get HttpGetJson of oJsonHttp "api.example.com" "customers" (&bOk) to hoJsonOut

Get HttpGetJson of oJsonHttp "api.example.com" "customers/1" (&bOk) to hoJsonOut

Get HttpGetJson of oJsonHttp "api.example.com" "customers/1/Orders" (&bOk) to hoJsonOut

Get HttpPostJson of oJsonHttp "api.example.com" "customers" hoJSONIn (&bOk) to hoJsonOut

Get HttpPatchJson of oJsonHttp "api.example.com" "customers/1" hoJSONIn (&bOk) to hoJsonOut

Get HttpDeleteJson of oJsonHttp "api.example.com" "customers/1" 0 (&bOk) to hoJsonOut
```

Example

http://jsonplaceholder.typicode.com/

Sample REST JSON API

Thank you for your time!

I'll be arround for any questions you might have…

6 – 7 . APRIL ATLANTA, GA, USA

SYNERGY 2017

Building for the future. Better, faster, everywhere.