Technical Note

Application Programming Interface

For FaceReader 8

Noldus

Information Technology

Documentation: Paul Ivan, Hans van Kuilenburg, Olga Krips, Leanne Loijens

December 2018

Noldus Information Technology bv.

International headquarters

Wageningen, The Netherlands

Phone +31-317-473300

Fax +31-317-424496

E-mail info@noldus.nl

For addresses of our other offices and support, please see our web site www.noldus.com.

1 What is an API?

"An application programming interface (API) is a particular set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers."
(http://en.wikipedia.org/wiki/Application_programming_interface)

This document describes how the FaceReader API can be used to integrate FaceReader into another application or framework. You do NOT need to read this document if you want to use FaceReader as a standalone tool.

If you want to use the API, please contact our support department for the possibilities. Our support department can also supply you with an example program that can receive FaceReader data. You can contact our support department via the **Help** menu of the FaceReader program, or via our website www.noldus.com/support-center.

2 What can the FaceReader API be used for?

The FaceReader API allows you to integrate FaceReader with third-party software or your own software. It allows you to:

- Start and stop FaceReader analysis.
- Score Stimuli and Event Markers (only with the Project Analysis Module).
- Receive the classification results from FaceReader (live as they become available).

Whenever FaceReader has classified an image, video frame or camera frame, a network package containing the emotional state or detailed values is sent out to connected applications (TCP/IP) that can then use these results. This allows application developers to, for example, adjust the content of a website or application based on the emotions of a user or to log FaceReader values together with input values from other sensors.

In early versions of FaceReader (3 or lower) this was already possible by constantly reading the last line from the “live log” that FaceReader can produce. However, the API offers a number of advantages:

- Lower I/O overhead; no hard disk operations are necessary anymore and results are only sent once they become available.
- Integrating with applications on platforms other than Windows becomes easier as reading large log files from shared network locations is not always feasible.
- Integrations with applications running on PCs in a completely different network, possibly on the other side of the world.
- Packages are sent in XML format which can be parsed consistently, regardless of the settings you choose in FaceReader.

3 How to use the API

To make use of the API there are two options:

- Use the supplied FaceReaderAPI.dll. This is a .Net.dll containing all functionality needed to receive from and send messages to FaceReader. The way to use this dll is explained by the sample source code of the 'FaceReaderExternalControlSample' application.
- Build your own TCP communication application and send the messages defined in the API. This is explained in this document.

RECEIVING FACEREADER DATA IN MICROSOFT .NET APPLICATIONS

For developers who use Microsoft .NET technology a library (FaceReaderAPI.dll) has been included that allows you to communicate with FaceReader with only a few lines of code.

Preparation – Start FaceReader

1. Install and start FaceReader
2. Choose **File** > **Settings** > **Data Export**.
3. Under **External communication (API and Stimulus Presentation Tool)**, select **Enable external control**.
4. Use FaceReader as you normally would. You may close all visualization windows to speed up the analysis

Adding FaceReaderAPI.dll to your project

1. Make sure the .NET 4.5 framework is installed on the computer where this dll is used. This framework can be downloaded from the Microsoft Download website, using Automatic Updates (check the Optional components). Alternatively, install FaceReader and uninstall it immediately afterwards. This will automatically install the .NET framework when it's missing.

2. Add **FaceReaderAPI.dll** as a reference in your project.

BUILD YOUR OWN TCP COMMUNICATION APPLICATION

The FaceReader API uses TCP messages to talk to FaceReader and to receive messages from FaceReader. These messages are XML documents describing the information. There are three types of messages:

- **Action message** – Used to make FaceReader perform an action. Possible actions are:
  - **Start Analyzing** – makes FaceReader start the currently open Analysis.
  - **Stop Analyzing** – makes FaceReader stop the currently open Analysis.
  - **Start Receiving Detailed Logs** – tells FaceReader to start sending detailed logs to this client
  - **Start Receiving State Logs** – tells FaceReader to start sending state logs to this client
  - **Stop Receiving Detailed Logs** – tells FaceReader to stop sending detailed logs to this client
  - **Stop Receiving State Logs**– tells FaceReader to stop sending state logs to this client
  - **Get Stimuli** – makes FaceReader send a *Response Message* containing a list of the available stimuli in the project.
  - **Get Event Markers** – makes FaceReader send a *Response Message* containing a list of the available event markers in the project.
  - **Score Stimulus** – makes FaceReader score a certain stimulus at the current timestamp.
  - **Score event marker** – makes FaceReader score a certain event marker at the current timestamp.
- **Response message** – Used by FaceReader to send a response corresponding to a received *Action Message*. Possible responses are:

  - **Error** – the action defined by the *Action Message* was not possible or another error occurred.
  - **Success** – the *Action Message* was handled correctly.
  - **Sends Stimuli** – FaceReader sends the available stimuli in the project as a response to a *Get Stimuli Action Message*.
  - **Sends Event Markers** – FaceReader sends the available event markers in the project as a response to a *Get Event Markers Action Message*.
- **Classification Message** – Contains the results of a frame/image classified by FaceReader. Can be either:
  - **State Log** – contains the dominant expressions only.
  - **Detailed Log** – contains all the classifications enabled in the Logging Settings.

Action Message

An action message contains:

- **Id** – The Id can be any string. Can be used to identify a **Response Message**. FaceReader will send the **Response Message** corresponding to this action with the same Id.
- Action Type:
  - FaceReader_Start_Analyzing
  - FaceReader_Stop_Analyzing
  - FaceReader_Start_DetailedLogSending
  - FaceReader_Start_StateLogSending
  - FaceReader_Stop_DetailedLogSending
  - FaceReader_Stop_StateLogSending
  - FaceReader_Get_Stimulus
  - FaceReader_Get_EventMarker

  Information – Can be any amount of strings. Currently only the messages for scoring a stimulus or scoring an event marker should contain the name of this marker in the Information.

Example of an action message without extra information

```
<?xml version="1.0" encoding="utf-8"?>

<ActionMessage xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">

<Id>ID01</Id>

<ActionType>FaceReader_Start_Analyzing</ActionType>

</ActionMessage>
```

Example of an action message with extra information

```
<?xml version="1.0" encoding="utf-8"?>

<ActionMessage xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">

<Id>ID02</Id>

<ActionType>FaceReader_Score_Stimulus</ActionType>

<Information>

<string>Stimulus 1</string>

</Information>

</ActionMessage>
```

Response Message

A response message contains:

- **Id** – The Id corresponds to the *Action Message* this message is responding to.
- **ResponseType** –
  - FaceReader_Sends_Error
  - FaceReader_Sends_Success
  - FaceReader_Sends_Stimuli
  - FaceReader_Sends_EventMarkers

- **Information** – Can be any amount of strings. Contains for example an error message or a success message or the Stimuli/ Event Markers that the project contains.

Example of a response message with information

```
<?xml version="1.0" encoding="utf-8"?>

<ResponseMessage xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">

<ResponseType>FaceReader_Sends_Error</ResponseType>

<Information>

<string>Already analyzing</string>

</Information>

</ResponseMessage>
```

Example of a response message with multiple Information

```
<?xml version="1.0" encoding="utf-8"?>

<ResponseMessage xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/
XMLSchema">

<ResponseType>FaceReader_Sends_Stimuli</ResponseType>

<Information>

<string>Stimulus 1</string>

<string>Stimulus 2</string>

<string>Stimulus 3</string>

</Information>

</ResponseMessage>
```

Classification Message

An action message contains:

- **Id** – The Id corresponds to the *Action Message* this message is responding to.

- LogType:
  - DetailedLog
  - StateLog
- AnalysisType:
  - Video
  - Camera
- FrameNumber
- **AnalysisStartTime** – Start time of the Analysis, when the first result was analyzed.
- **FrameTimeTicks** – TimeStamp of the frame in ticks. (1 second = 10.000.000 ticks)
  - In case of video, the FrameTimeTicks is the frame time in the video
  - In case of camera, the FrameTimeTicks is the frame time past the AnalysisStartTime
- **ClassificationValues** – List of ClassificationValue objects
  - Classification Value
    - Type
      - Value (in case of a numeric value)
      - State (in case of a string value)
    - Label
      - Name of the ClassificationValue
    - - Value
      - Only has a value if Type == Value
      - Can be any amount of floats
    - State
      - Only has a value if Type == State
      - Can be any amount of strings

Example of a Value-Typed ClassificationValue

```
<ClassificationValue>
<Type>Value</Type>
<Label>Neutral</Label>
<Value>
<float>0.201352075</float>
</Value>
<State />
</ClassificationValue>
```

Example of a State-Typed ClassificationValue

```
<ClassificationValue>
<Type>State</Type>
<Label>Gender</Label>
<Value />
<State>
<string>Female</string>
</State>
</ClassificationValue>
```

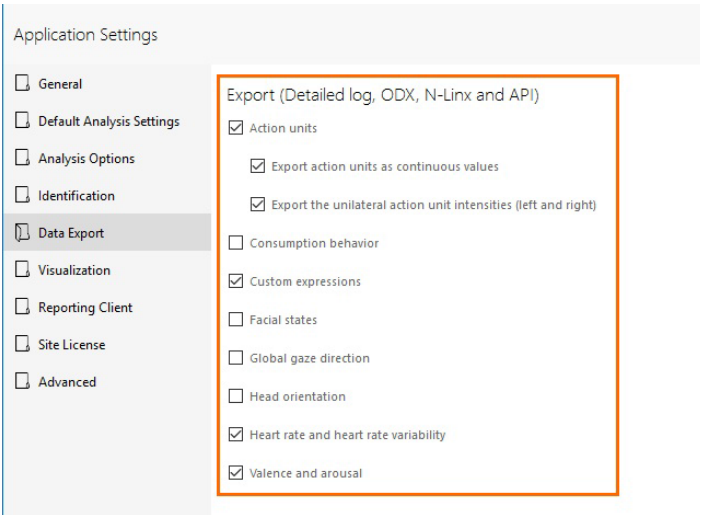
Example of a MultiValued-Typed ClassificationValue

```
<ClassificationValue>
<Type>Value</Type>
<Label>Example</Label>
<Value>
<float>282.03125</float>
<float>301.93515</float>
<float>290.5138</float>
<float>300.786652</float>
```

```
<float>300.093658</float>

<float>302.1726</float>

</Value>

<State />

</ClassificationValue>
```

Example of a small Detailed Log

Note that the ClassificationValues reported by the API are selected in with **File** -> **Settings** -> **Data Export** -> **Export (Detailed Log, ODX, N-Linx and API)**.



Export (Detailed log, ODX, N-Linx and API)

- ✓ Action units
  - ✓ Export action units as continuous values
  - ✓ Export the unilateral action unit intensities (left and right)
- ☐ Consumption behavior
- ✓ Custom expressions
- ☐ Facial states
- ☐ Global gaze direction
- ☐ Head orientation
- ✓ Heart rate and heart rate variability
- ✓ Valence and arousal

For a full example of a Detailed Log, see Appendix A.

```
<?xml version="1.0" encoding="utf-8"?>

<Classification xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" mlns:xsd="http://www.w3.org/2001/XMLSchema">

<LogType>DetailedLog</LogType>
```

```
<AnalysisType>Video</AnalysisType>
<FrameNumber>9244</FrameNumber>
<AnalysisStartTime>2012-08-01T06:54:24.4580478+02:00</
AnalysisStartTime>
<FrameTimeTicks>3697600000</FrameTimeTicks>
<ClassificationValues>
<ClassificationValue>
<Type>Value</Type>
<Label>Neutral</Label>
<Value>
<float>0.201352075</float>
</Value>
<State />
</ClassificationValue>
<ClassificationValue>
<Type>Value</Type>
<Label>Happy</Label>
<Value>
<float>0.00125725183</float>
</Value>
<State />
</ClassificationValue>
<ClassificationValue>
<Type>Value</Type>
<Label>Sad</Label>
<Value>
<float>0.151250109</float>
```

```
</Value>

<State />

</ClassificationValue>

<ClassificationValue>

<Type>Value</Type>

<Label>Angry</Label>

<Value>

<float>0.2761046</float>

</Value>

<State />

</ClassificationValue>

</ClassificationValues>

</Classification>
```

Example of a StateLog

```
<?xml version="1.0" encoding="utf-8"?>

<Classification xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<LogType>StateLog</LogType>

<AnalysisType>Video</AnalysisType>

<FrameNumber>9601</FrameNumber>

<AnalysisStartTime>2012-08-01T06:54:24.4580478+02:00</
AnalysisStartTime>

<FrameTimeTicks>3840400000</FrameTimeTicks>

<ClassificationValues>

<ClassificationValue>

<Type>State</Type>

<Label>Dominant Expression</Label>
```

```
<Value />
<State>
<string>Happy</string>
</State>
</ClassificationValue>
</ClassificationValues>
</Classification>
```

Message Packages

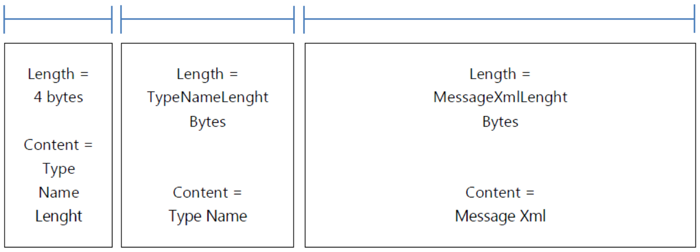
The described messages are sent as bytes over TCP. Here we describe how byte packages are configured. A message is converted to bytes in the following manner:

- First 4 bytes contain the length in bytes (TypeNameLenght) of the typename of the message.
  - Typenames can be:
    - FaceReaderAPI.Messages.ActionMessage
    - FaceReaderAPI.Messages.ResponseMessage
    - FaceReaderAPI.Data.Classification
- The next TypeNameLenght bytes contain the typename as UTF8Encoded bytes
- The rest of the bytes are the Message Xml as UTF8Encoded bytes

Message converted to an array of bytes (with length MessageBytesLength)

MessageBytes = array of bytes of length MessageBytesLength

Containing:



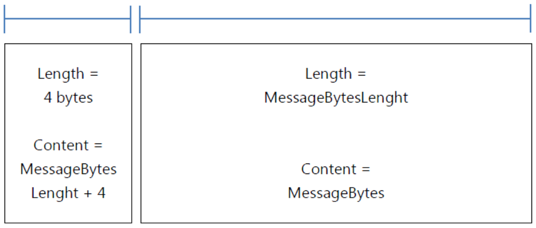
thus:

MessageBytesLength = 4 + TypeNameLength + MessageXmlLength

The bytes of a message as packages sent using TCP. The messages are first prefixed with the length of the message to make receiving easier. So the actual package sent of TCP consist of the following bytes:

- First 4 bytes contain the length of the message as defined above + 4 (thus the total length of the package).
  - Thus MessageBytesLenght + 4
- The rest of the bytes contain the MessageBytes.

PackageBytes = array of bytes of length MessageBytesLength + 4.



4 Technical Support

Technical Support on using the Application Programming Interface is available for people with NoldusCare. Please look on our web site (http://www.noldus.com/support-center/nolduscare) for more information about NoldusCare.

The use of the Application Programming Interface itself is supported for customers with NoldusCare. However, due to the large number of possible programming languages and operating systems that allow implementation of the API into custom made applications, we are unable to provide active support for the applications you make yourself.

You can contact our Help Desk via our web site (www.noldus.com/support-center) and fill out a Support Request Form (preferred), or phone during working hours in three time zones.

Note that if you send us videos showing people's faces, you should have permission from those people that you can use the video for that purpose and you may need to sign a form granting consent for us to use those videos.

Before you contact Technical Support, please have the version number and license number of your copy of FaceReader available. To find these numbers, from the **Help** menu select **About**.

Please refer to the About Noldus - Contact section on our web site (www.noldus.com) for other contact information.

Appendix A

EXAMPLE OF A FULL DETAILED LOG

Note: The landmarks, are logged as a float array containing (x_1, y_1, x_2, y_2, …, x_n, y_n)

```
<?xml version="1.0" encoding="utf-16"?>
<Classification xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <LogType>DetailedLog</LogType>
  <AnalysisType>Video</AnalysisType>
  <FrameNumber>205</FrameNumber>
  <AnalysisStartTime>2017-09-11T12:21:13.7739241+02:00</
AnalysisStartTime>
  <FrameTimeTicks>136666666</FrameTimeTicks>
  <ClassificationValues>
    <ClassificationValue>
      <Type>Value</Type>
      <Label>Neutral</Label>
      <Value>
        <float>0.7553205</float>
      </Value>
      <State />
    </ClassificationValue>
    <ClassificationValue>
      <Type>Value</Type>
      <Label>Happy</Label>
      <Value>
  <float>0.0445850864</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Sad</Label>
 <Value>
  <float>0.205180719</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Angry</Label>
 <Value>
  <float>0.0232287664</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Surprised</Label>
 <Value>
  <float>0.0205680113</float>
 </Value>
 <State />
```

```
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Scared</Label>
 <Value>
  <float>0.04258409</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Disgusted</Label>
 <Value>
  <float>0.02142761</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Contempt</Label>
 <Value>
  <float>0.101011038</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
```

```
  <Label>Valence</Label>
  <Value>
    <float>-0.160595626</float>
  </Value>
  <State />
</ClassificationValue>
<ClassificationValue>
  <Type>Value</Type>
  <Label>Arousal</Label>
  <Value>
    <float>0.12003836</float>
  </Value>
  <State />
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Gender</Label>
  <Value />
  <State>
    <string>Female</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Age</Label>
  <Value />
  <State>
```

```
    <string>25 - 35</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Beard</Label>
  <Value />
  <State>
    <string>None</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Moustache</Label>
  <Value />
  <State>
    <string>None</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Glasses</Label>
  <Value />
  <State>
    <string>No</string>
  </State>
</ClassificationValue>
```

```
<ClassificationValue>
 <Type>State</Type>
 <Label>Ethnicity</Label>
 <Value />
 <State>
  <string>EasternAsian</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Y - Head Orientation</Label>
 <Value>
  <float>-15.7117033</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>X - Head Orientation</Label>
 <Value>
  <float>1.48214245</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Z - Head Orientation</Label>
 <Value>
  <float>0.565930247</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Quality</Label>
 <Value>
  <float>0.9733813</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Mouth</Label>
 <Value />
 <State>
  <string>Closed</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Left Eye</Label>
 <Value />
 <State>
  <string>Open</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Right Eye</Label>
 <Value />
 <State>
  <string>Open</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Left Eyebrow</Label>
 <Value />
 <State>
  <string>Raised</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Right Eyebrow</Label>
 <Value />
 <State>
  <string>Neutral</string>
 </State>
</ClassificationValue>
<ClassificationValue>
```

```
  <Type>State</Type>
  <Label>Gaze Direction</Label>
  <Value />
  <State>
    <string>Forward</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Identity</Label>
  <Value />
  <State>
    <string>Hansius</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Action Unit 01 - Inner Brow Raiser</Label>
  <Value />
  <State>
    <string>NotActive</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Action Unit 02 - Outer Brow Raiser</Label>
  <Value />
```

```
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 04 - Brow Lowerer</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 05 - Upper Lid Raiser</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 06 - Cheek Raiser</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
```

```
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 07 - Lid Tightener</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 09 - Nose Wrinkler</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 10 - Upper Lip Raiser</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 12 - Lip Corner Puller</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 14 - Dimpler</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 15 - Lip Corner Depressor</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 17 - Chin Raiser</Label>
 <Value />
 <State>
```

```
    <string>NotActive</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Action Unit 18 - Lip Puckerer</Label>
  <Value />
  <State>
    <string>NotActive</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Action Unit 20 - Lip Stretcher</Label>
  <Value />
  <State>
    <string>NotActive</string>
  </State>
</ClassificationValue>
<ClassificationValue>
  <Type>State</Type>
  <Label>Action Unit 23 - Lip Tightener</Label>
  <Value />
  <State>
    <string>NotActive</string>
  </State>
</ClassificationValue>
```

```
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 24 - Lip Pressor</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 25 - Lips Part</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 26 - Jaw Drop</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 27 - Mouth Stretch</Label>
 <Value />
 <State>
  <string>NotActive</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>State</Type>
 <Label>Action Unit 43 - Eyes Closed</Label>
 <Value />
 <State>
  <string>C</string>
 </State>
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Heart Rate</Label>
 <Value>
  <float>64</float>
 </Value>
 <State />
</ClassificationValue>
<ClassificationValue>
 <Type>Value</Type>
 <Label>Interest</Label>
 <Value>
  <float>0</float>
 </Value>
```

```
      <State />
    </ClassificationValue>
    <ClassificationValue>
      <Type>Value</Type>
      <Label>Boredom</Label>
      <Value>
        <float>0.6230996</float>
      </Value>
      <State />
    </ClassificationValue>
    <ClassificationValue>
      <Type>Value</Type>
      <Label>Confusion</Label>
      <Value>
        <float>0</float>
      </Value>
      <State />
    </ClassificationValue>
  </ClassificationValues>
</Classification>
```