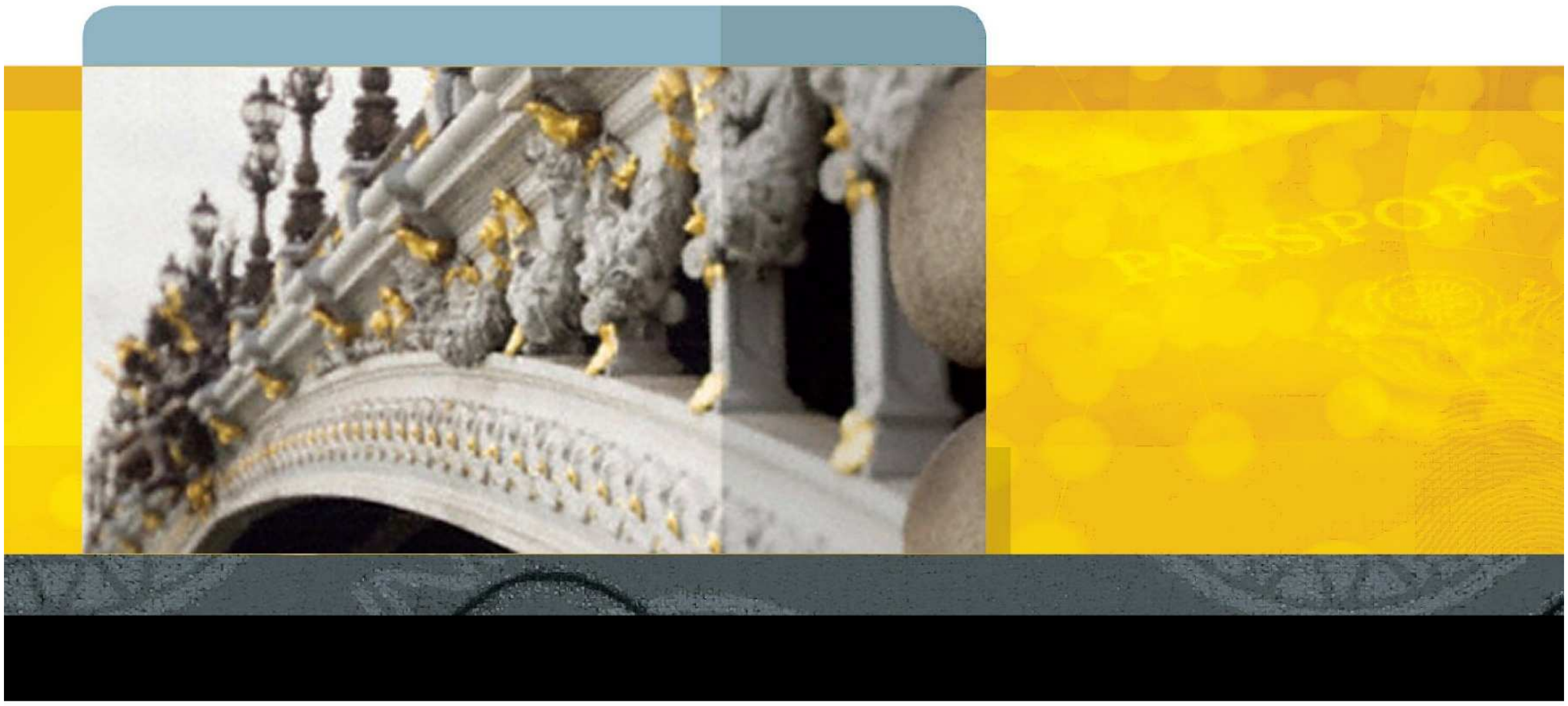

3M™ Output Wedge Reference Manual

Manual No:
Version 1.0
Date: 18 September 2015

Contents **Page**

1 Introduction

1.1 Warnings, Cautions and Notes

This manual contains important information regarding the operation of the 3M Page Reader family. For safe and reliable operation of the readers all users must ensure that they are familiar with and fully understand all instructions contained herein.

⚠ DANGER	**Danger** indicates a hazardous situation which, if not avoided, ***will*** result in death or serious injury.
⚠ WARNING	**Warning** indicates a hazardous situation which, if not avoided, ***could*** result in death or serious injury.
⚠ CAUTION	**Caution** indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
NOTICE	**Notice** indicates a situation which, if not avoided, could result in property damage only. This includes situations which require you to re-install your software or return your equipment to the manufacturer for recalibration.
INFORMATION	**Information** indicates important information that helps get the optimum performance from your scanner and will save you time during evaluation and deployment.

1.2 Proprietary Statement

By using the 3M™ Page Reader SDK (the "Product"), you (the "User"), agree to be bound by the following terms and conditions.

Because use of the Product varies widely and is beyond the control of 3M, the User must evaluate and determine whether a Product is fit for a particular purpose and suitable for User's application prior to use. THE FOLLOWING IS MADE IN LIEU OF ALL EXPRESS AND IMPLIED WARRANTIES OR CONDITIONS (INCLUDING WARRANTIES OR CONDITIONS OF SUITABILITY AND FITNESS FOR A PARTICULAR PURPOSE). If a Product is proved to be defective (the "Defective Product"), the exclusive remedy, at 3M's option, shall be to either repair or replace the Defective Product or refund the purchase price of the Defective Product.

LIMITATION OF LIABILITY: OTHER THAN IN RELATION TO DEATH OR PERSONAL INJURY CAUSED BY ITS NEGLIGENCE 3M AND SELLER, IF ANY, SHALL NOT BE LIABLE FOR ANY INJURY, LOSS OR DAMAGE, WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL HOWSOEVER CAUSED, (INCLUDING DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, INCREASED COSTS OF OPERATION, LITIGATION COSTS AND THE LIKE), WHETHER BASED UPON A CLAIM OR ACTION IN CONTRACT (INCLUDING BREACH OF WARRANTY), TORT, (INCLUDING NEGLIGENCE) OR OTHERWISE, IN CONNECTION WITH THE USE OR PERFORMANCE OF A PRODUCT.

1.3 Notices

3M reserves the right to make changes to its Products at any time and without notice. The information furnished by 3M in this manual is believed to be accurate and reliable. The material contained herein is supplied without any representation or warranty of any kind. 3M therefore assumes no responsibility, consequential or otherwise, of any kind arising from the use of the Product.

1.4 Trademarks & Acknowledgements

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.
All trademarks are acknowledged. All 3M trademarks are trademarks of 3M Company.

1.5 Office Locations

The Americas
3M Center Building 225-4N-14
St. Paul, MN
USA 55144-1000
Telephone: +1 800 581 2631
Web: http://www.3M.com/IdentityManagement

Europe, Middle East and Africa
3M Centre
3M Traffic Safety and Security Division
Cain Road, Bracknell
Berkshire
RG12 8HT
United Kingdom
Telephone: +44 (0) 8705 360036
Fax: +44 (0)1344 858278

Asia, Pacific and Australia
3M Asia Pacific Pte Ltd
3M Traffic Safety and Security Division
1 Yishun Avenue 7
Singapore 768923
Telephone: +65 6450 8888
Fax: +65 6458 5432

1.6 Global Technical Services

The Americas

direct line:	+1 613-722-3629
main number:	+1 613-722-2070
fax:	+1 613-722-2063
email:	3M-AiT-gcs@mmm.com

United Kingdom

direct line:	+44 (0) 1344 858 024
main number:	+44 (0) 1344 858 000
email:	gcs-uk@mmm.com

Asia, Pacific and Australasia

telephone:	+65 6450 8888
fax:	+65 6458 5432

1.7 Revision History

Version	Date	Description
V1.00	Sept 2015	Initial conversion from doxygen to word/pdf.

2 Overview

This reference manual explains how to write script files for the 3M Output Wedge.

Script files are written in XML syntax, so should be quite easy to read. All script files have the following parent element:

```
<?xml version="1.0"?>
<OutputWedge>
      ...
</OutputWedge>
```

The following sections will explain the further sub-components of each script file:

- Scripts and Handlers
- Data Merging
- Input Sources
- Output Sources

3 Scripts and Handlers

3.1 Scripts

A script file can actually have more than one script. A script is defined using the following syntax:

```
<Script name="..." startup="..." sync="...">

      ...(Handlers go here)...

</Script>
```

Attributes:

Startup: This attribute set to "True" will be the script activated on startup. If this attribute is not supplied anywhere, then the first defined script is the startup script. This can be useful as the user can switch between scripts at runtime using the application.

Sync: This attribute set to "True" will wait to execute it's Event and Data Handlers until an end of data event has been received. This is neccessary when data merging from multiply sources codeline/rf/ammva is required.

UnknownDoc: This attribute set to "False" will filter out any Unknown documents from the sdk CodelineData structure, preventing them from being sent to the client application keyboard queue.

BadCharDoc: This attribute set to "False" will filter out any documents where the sdk CodelineData structure contains one of the BadChar list from the Input Source parameter, preventing them from being sent to the client application keyboard queue.

3.2 Handlers

A Handler is defined as a set of output tasks to perform when something is triggered in the input source. There are two kinds:

3.2.1 Event Handlers

```
<EventHandler name="...">
      ...(Tasks go here)...
</EventHandler>
```

Event Handlers are used when the input source raises an event with no data associated with it.

Note:

See each input source to know which events are returned.

3.2.2 Data Handlers

```
<DataHandler name="..." storeAs="...">
      ...(Tasks go here)...
</DataHandler>
```

Data Handlers are used when the input source returns some data. That data will be stored in the variable name given by the `storeAs` attribute which any of the tasks can then access. See Variables for more details on variables.

Note:

See each input source to know which data is returned.

3.2.3 Dependencies

Sometimes you will want an Event/Data Handler to be executed before another, especially if the order of the events/data returned is not guaranteed. To solve this, you can add the following to any Event/Data Handler:

```
<DataHandler name="..." storeAs="...">
      <DependsOn handler="..." />
      ...(Tasks go here)...
</DataHandler>
```

When these elements are added, this Data Handler will not be executed until all other dependencies have been executed first.

4 Data Merging

Merges data from different data items into a single output item name.

Syntax

```
<DataMerging>
      <DestDataItem  name="...">
            <SrcDataItem name="..." priority="..." />
            ...
      </DestDataItem>
</DataMerging>
```

DestDataItem

The `DestDataItem` defines the output data item name to contain the text that will be sent to the output module. The item name can be any string of characters with the exception of the pre defined data types assigned by the input modules.

This can be useful as the user can switch between scripts at runtime using the application.

SrcDataItem

The `SrcDataItem` defines the input data item name to merge into the `DestDataItem`. The item name must be one of the pre defined data types assigned by the input modules. When merging from multiply sources, the code will assign the src data item with the highest priority number. If src data items have the same priority, first from the sdk will be used.

5 Input Sources

An input source is a module component which exclusively waits for input to arrive from some defined source and then passes it on to the Output Wedge manager.

Input source modules/DLLs always use the naming convention **WedgeInputModule_[Name].dll**, where **[Name]** is the name of the input source. Only use the **[Name]** part of an input source when defining it in a script, the full module/DLL name is not necessary as the Output Wedge will know what to look for.

The following input source modules are available:

- Input Source: FullPage
- Input Source: Swipe

Note:

Only **one** input source can be defined for a script.

Syntax

The following example shows how to define an input source in a script:

```
<InputModule name="FullPage">
      <Parameters>
            <Parameter name="LogLevel" value="1" />
            <Parameter name="LogFileName" value="FullPageOutputWedge.log" />
            <Parameter name="DateOfBirthFormat" value="%d-%b-%Y" />
            <Parameter name="ExpiryDateFormat" value="%d-%b-%Y" />
      </Parameters>
</InputModule>
```

From the above example we can see that we want to use the `FullPage` input module as our input source. It also happens that this input source can accept extra parameters, so we list what parameters we want to set also.

However, if we don't want to set any parameters explicitly and just use default values, we could also have written it like this:

```
<InputModule name="FullPage" />
```

Parameters

Name	Optional?	Description
LogLevel	Yes	Sets the log level of the 3M Reader SDK log file. Defaults to 0 (errors only). Must be between 0 and 5.
LogFile	Yes	Sets the file name of the 3M Reader SDK log file. Defaults to "SwipeOutputWedge.log" for the Swipe Reader. Defaults to "FullPageOutputWedge.log" for the Full Page Reader
DateOfBirthFormat	Yes	Sets the formatting rule to apply to CODELINE_DATE_OF_BIRTH_STR_FORMATTED. Defaults to "dd-mm-yyyy".
ExpiryDateFormat	Yes	Sets the formatting rule to apply to CODELINE_EXPIRY_DATE_STR_FORMATTED. Defaults to "dd-mm-yyyy".
BadChars	Yes	Sets the list of bad chars to search for in a codeline. Defaults to "*_".

Date Conversion Rules

Note:

The following rules apply to invalid date conditions:
Unknown (<< or misread) day is always assumed to be 01
Unknown (<< or misread) month is always assumed to be January
Unknown (<< or misread) year is always assumed to be 2000
Invalid dates (eg, 31 of the 13th month) may transform into other "valid" dates
Expiry dates are assumed to be current century
Date of Birth is current century for 2 digit years < this year, all other years are assumed as previous century

Date Formatting

Format	Description	Example
%a	Short weekday name	Mon
%A	Full weekday name	Monday
%d	Day of Month as a decimal number (01-31)	01
%b	Abreviated Month name	Jan
%B	Month name	January
%m	Month as a decimal number	01
%y	Year without the Century	12
%Y	Year with the century	2012
%x	Date formatted according to current system locale	

The following examples shows how to define date formats:

```
<Parameter name="DateOfBirthFormat" value="%d-%b-%Y" />

            01-Jan-2012

<Parameter name="ExpiryDateFormat" value="%d/%m/%y" />

            01/01/12
```

5.1 Input Source: FullPage

Module/DLL Filename: WedgeInputSource_FullPage.dll

Uses the High Level API of the 3M Page Reader SDK to receive data from a full page reader.

Syntax

```
<InputModule name="FullPage">
      <Parameters>
            <Parameter name="..." value="..." />
            ...
      </Parameters>
</InputModule>
```

Events

The following events can be used in script Event Handlers. Please refer to the SDK documentation/source code to understand when each one is invoked.

- DOC_ON_WINDOW
- DOC_REMOVED
- START_OF_DOCUMENT_DATA
- END_OF_DOCUMENT_DATA
- AUTOMATIC_STATE_CHANGE
- RF_CHIP_OPENED_SUCCESSFULLY
- RF_CHIP_OPEN_FAILED
- READER_ERROR_RESOLVED
- SETTINGS_INITIALISED
- PLUGINS_INITIALISED
- START_OF_PLUGINS_DECODE
- RF_CHIP_OPEN_TIMEOUT
- RF_CHIP_REMOVAL_SUCCESS
- RF_CHIP_REMOVAL_TIMEOUT

- READY_FOR_SMARTCARD
- BEGIN_RESOLVING_ERROR
- COM_PORT_OPEN
- COM_PORT_CLOSED
- READING_DATA
- DATA_READ
- START_OF_SWIPE_DATA
- END_OF_SWIPE_DATA
- DEVICE_CONNECTED
- DEVICE_DISCONNECTED
- SWIPE_READER_CONNECTED
- SWIPE_READER_DISCONNECTED

Example:

```
<EventHandler name="START_OF_DOCUMENT_DATA">
      ... (Tasks go here)...
</EventHandler>
```

Data

The following data types can be used in script Data Handlers. Please refer to the SDK documentation/source code to understand what each one does.

- CODELINE_DATA
    - CODELINE_LINE_1
    - CODELINE_LINE_2
    - CODELINE_LINE_3
    - CODELINE_DATE_OF_BIRTH_STR
    - CODELINE_DATE_OF_BIRTH_STR_FORMATTED
    - CODELINE_DOC_ID
    - CODELINE_DOC_NUMBER
    - CODELINE_DOC_TYPE
    - CODELINE_OPTIONAL_DATA_1
    - CODELINE_OPTIONAL_DATA_2
    - CODELINE_EXPIRY_DATE_STR
    - CODELINE_EXPIRY_DATE_STR_FORMATTED
    - CODELINE_FORENAME
    - CODELINE_FORENAMES
    - CODELINE_ISSUING_STATE
    - CODELINE_NATIONALITY
    - CODELINE_SECOND_NAME
    - CODELINE_SEX

  - CODELINE_SHORT_SEX
  - CODELINE_SURNAME
- IMAGEIR
- IMAGEVIS
- IMAGEPHOTO
- IMAGEUV
- IMAGEBARCODE
- SCDG1_CODELINE_DATA
  - SCDG1_CODELINE_LINE_1
  - SCDG1_CODELINE_LINE_2
  - SCDG1_CODELINE_LINE_3
  - SCDG1_CODELINE_DATE_OF_BIRTH_STR
  - SCDG1_CODELINE_DOC_ID
  - SCDG1_CODELINE_DOC_NUMBER
  - SCDG1_CODELINE_DOC_TYPE
  - SCDG1_CODELINE_OPTIONAL_DATA_1
  - SCDG1_CODELINE_OPTIONAL_DATA_2
  - SCDG1_CODELINE_EXPIRY_DATE_STR
  - SCDG1_CODELINE_FORENAME
  - SCDG1_CODELINE_FORENAMES
  - SCDG1_CODELINE_ISSUING_STATE
  - SCDG1_CODELINE_NATIONALITY
  - SCDG1_CODELINE_SECOND_NAME
  - SCDG1_CODELINE_SEX
  - SCDG1_CODELINE_SHORT_SEX
  - SCDG1_CODELINE_SURNAME
- SCDG2_PHOTO
- MSR
  - SWIPE_MSR_TRACK_1
  - SWIPE_MSR_TRACK_2
  - SWIPE_MSR_TRACK_3
- BARCODE_1D_INDUSTRIAL_2_OF_5
- BARCODE_1D_INTERLEAVED_2_OF_5
- BARCODE_1D_IATA_2_OF_5
- BARCODE_1D_3_OF_9
- BARCODE_1D_128
- BARCODE_PDF417
- BARCODE_AZTECCODE
- BARCODE_QRCODE
- BARCODE_DATAMATRIX
- AAMVA_DATA
  - AAMVA_PARSED_LICENCENUMBER
  - AAMVA_PARSED_FULLNAME

- AAMVA_PARSED_SURNAME
- AAMVA_PARSED_FORENAME
- AAMVA_PARSED_MIDDLENAME
- AAMVA_PARSED_MIDDLENAME2
- AAMVA_PARSED_MIDDLENAME3
- AAMVA_PARSED_NAMESUFFIX
- AAMVA_PARSED_GIVENNAMES
- AAMVA_PARSED_SEX
- AAMVA_PARSED_ADDRESS_STREET
- AAMVA_PARSED_ADDRESS_CITY
- AAMVA_PARSED_ADDRESS_STATE
- AAMVA_PARSED_ADDRESS_POSTALCODE
- AAMVA_PARSED_ADDRESS_COUNTRY
- AAMVA_PARSED_SHORTSEX
- AAMVA_PARSED_DATEOFBIRTH
- AAMVA_PARSED_ISSUEDATE
- AAMVA_PARSED_EXPIRYDATE
- AAMVA_PARSED_DATEOFBIRTH_FORMATTED
- AAMVA_PARSED_ISSUEDATE_FORMATTED
- AAMVA_PARSED_EXPIRYDATE_FORMATTED
- AAMVA_PARSED_HEIGHT
- AAMVA_PARSED_WEIGHT
- AAMVA_PARSED_HAIR_COLOR
- AAMVA_PARSED_EYE_COLOR

Example:

```
<DataHandler name="CODELINE_SURNAME" storeAs="$data">
      ...(Tasks go here)...
</DataHandler>
```

5.2 Input Source: Swipe

Module/DLL Filename: WedgeInputSource_Swipe.dll

Uses the Low Level API of the 3M Swipe Reader SDK to receive data from a swipe reader.

Syntax

```
<InputModule name="Swipe">
      <Parameters>
```

```
            <Parameter name="..." value="..." />
            ...
      </Parameters>
</InputModule>
```

Events

The following events can be used in script Event Handlers. Please refer to the SDK documentation/source code to understand when each one is invoked.

- READER_ERROR_RESOLVED
- BEGIN_RESOLVING_ERROR
- COM_PORT_OPEN
- COM_PORT_CLOSED
- READING_DATA
- DATA_READ
- START_OF_SWIPE_DATA
- END_OF_SWIPE_DATA
- DEVICE_CONNECTED
- DEVICE_DISCONNECTED
- SWIPE_READER_CONNECTED
- SWIPE_READER_DISCONNECTED

Example:

```
<EventHandler name="START_OF_SWIPE_DATA">
      ... (Tasks go here)...
</EventHandler>
```

Data

The following data types can be used in script Data Handlers. Please refer to the SDK documentation/source code to understand what each one does.

- WHOLE_DATA
- MESSAGE_CONTENT
- CODELINE_DATA
  - CODELINE_LINE_1
  - CODELINE_LINE_2
  - CODELINE_LINE_3
  - CODELINE_DATE_OF_BIRTH_STR
  - CODELINE_DATE_OF_BIRTH_STR_FORMATTED
  - CODELINE_DOC_ID
  - CODELINE_DOC_NUMBER

    - CODELINE_DOC_TYPE
    - CODELINE_OPTIONAL_DATA_1
    - CODELINE_OPTIONAL_DATA_2
    - CODELINE_EXPIRY_DATE_STR
    - CODELINE_EXPIRY_DATE_STR_FORMATTED
    - CODELINE_FORENAME
    - CODELINE_FORENAMES
    - CODELINE_ISSUING_STATE
    - CODELINE_NATIONALITY
    - CODELINE_SECOND_NAME
    - CODELINE_SEX
    - CODELINE_SHORT_SEX
    - CODELINE_SURNAME
- MSR
    - SWIPE_MSR_TRACK_1
    - SWIPE_MSR_TRACK_2
    - SWIPE_MSR_TRACK_3
- ATB
    - SWIPE_ATB_TRACK_1_BLOCK_1
    - SWIPE_ATB_TRACK_1_BLOCK_2
    - SWIPE_ATB_TRACK_1_BLOCK_3
    - SWIPE_ATB_TRACK_2_BLOCK_1
    - SWIPE_ATB_TRACK_2_BLOCK_2
    - SWIPE_ATB_TRACK_2_BLOCK_3
    - SWIPE_ATB_TRACK_3_BLOCK_1
    - SWIPE_ATB_TRACK_3_BLOCK_2
    - SWIPE_ATB_TRACK_3_BLOCK_3
    - SWIPE_ATB_TRACK_4_BLOCK_1
    - SWIPE_ATB_TRACK_4_BLOCK_2
    - SWIPE_ATB_TRACK_4_BLOCK_3

Example:

```
<DataHandler name="CODELINE_SURNAME" storeAs="$data">
      ...(Tasks go here)...
</DataHandler>
```

6 Output Sources

An output source is a module component which exclusively forwards on any data received from an input source to some external output source.

Output source modules/DLLs always use the naming convention **WedgeOuptutModule_[Name].dll**, where **[Name]** is the name of the output source. Only use the **[Name]** part of an output source when defining it in a script, the full module/DLL name is not necessary as the Output Wedge will know what to look for.

The following output source modules are available:

- Runtime Output Modules
- Output Source: Keyboard
- Output Source: Internet Explorer Web Page
- Output Source: Swipe Protocols
- Output Source: Virtual Serial Port

Note:

One or more output sources can be defined in a script, allowing you to use more than one output source at once.

Syntax

The following example shows how to define one or more output sources in a script:

```
<OutputModules>
      <OutputModule name="Keyboard" alias="keyb">
            <Parameters>
                  <Parameter name="DelayBetweenKeys" value="10" />
            </Parameters>
      </OutputModule>
      ...
</OutputModules>
```

From the above example we can see that we want to use the `Keyboard` output module as our output source. It also happens that this output source can accept extra parameters, so we list what parameters we want to set also.

However, if we don't want to set any parameters explicitly and just use default values, we could also have written it like this:

```
<OutputModules>
      <OutputModule name="Keyboard" alias="keyb" />
```

```
</OutputModules>
```

If more than one output module needs to be used, then the following can be written:

```
<OutputModules>
      <OutputModule name="VirtualSerialPort" alias="vcom" />
      <OutputModule name="SwipeProtocols" alias="proto" />
</OutputModules>
```

You can also carry out some one-time initialisation tasks for an output source. For example, if using the `VirtualSerialPort` output source:

```
<OutputModules>
      <OutputModule name="VirtualSerialPort" alias="vcom">
            <Init>
                  <Task name="vcom:OpenPort" port="6" setCTS="True" setDSR="True"
/>
            </Init>
      </OutputModule>
<OutputModules>
```

Aliases

In the examples above, you will have noticed that there is an `alias` attribute for each `OutputModule`. The alias is used by the Output Wedge to look up output tasks in Data/Event Handlers. It is similar in concept to a namespace in conventional programming languages; it groups related output tasks together. It is also of use when multiple output modules are used to differentiate between which output task is being used.

As an example, look at the following script extract:

```
<OutputModules>
      <OutputModule name="Keyboard" alias="keyb" />
</OutputModules>

<Script name="Main" startup="True">

      <EventHandler name="START_OF_SWIPE_DATA">
            <Task name="keyb:TypeString" source="START" />
            <Task name="keyb:PressSpecialKey" key="RETURN" ctrl="False" alt="False"
shift="False" />
      </EventHandler>

</Script>
```

You will notice that in the EventHandler for `START_OF_SWIPE_DATA` there are two output tasks to be carried out. These tasks can be found under the `Keyboard` output module because they use the `keyb` alias defined.

Therefore, all output task names should have the format **[Alias]:[Name]**, where **[Name]** is the actual name of the task. Task names are defined in each Output Module; refer to the pages for each output module to find out what tasks they expose.

Note:

The alias can be named anything you want as long as it is used consistently throughout the script.

Variables

In the case of Data Handlers, the data these return must be stored somewhere for each output task to access it. Therefore, scripts can use variables to store data returned by Data Handlers and also during the execution of a Data/Event Handler.

Attention:

Variables follow the convention **$[Name]**. Any name that uses the $ symbol is assumed to be a variable name and not a hardcoded data item.

Tasks

Each output module exposes a set of tasks that it can perform. To use a task in a script, use the following syntax:

```
<Task name="..." [param="..." ...] />
```

Each task must have a name so that the Output Wedge can identify it. The `name` attribute value should use the naming convention defined in the Aliases section.

If the task requires more input/parameters, these are written as XML attributes after the `name` attribute. Each defined task will list what parameters it accepts. Please refer to each output module for further details on what each task does and how it is defined.

6.1 Runtime Output Modules

The runtime output modules are actually built into the Output Wedge (the source code can be found in the **OutputWedgeDll** project). These modules are provided since they provide quite common functionality that could be used in any script.

Note:

Runtime modules all have a fixed alias.

The following runtime modules are provided:

- Output Source: File
- Output Source: Math
- Output Source: String
- Output Source: Variables
- Output Source: Utility

6.1.1 Output Source: File

Fixed Alias: file

Used to perform saving of image and text data to disk file. This was previous categorized as "image". The application will still accept "image:SaveImage" tasks.

6.1.1.1 SaveImage

```
<Task name="file:SaveImage" source="..." dest="..." format="..." />
```

Saves an image to file, or can convert an image in memory to another format.

Parameters:

source	Specify what input data to use. Must be a variable reference, e.g. **$ref**. The source image must be in a suitable format, i.e. byte array.
dest	Specify where to save the image. Can optionally be a variable reference. If not a variable, *dest* is assumed to be a filename.
format	Specify what format to save the image as. Can be one of the following:

- BMP
- JPEG
- GIF
- TIFF
- PNG
- JPEG2000

- WSQ

6.1.1.2 SetFilename

```
<Task name="file:SetFilename" dest="..." />
```

Set the default filename for subsequent disk writes.

Parameters:

dest	Specify where to save subsequent text string tasks. Dest is assumed to be a filename.

6.1.1.3 TextString

```
<Task name="file:TextString" source="..." tab="..." linefeed="..." dest="..." />
```

Saves the text string source to the dest file or default file perviously assigned by SetFilename.

Parameters:

source	Specify what to type. Can be a variable reference. If it is not a variable, then it assumes that *source* is a string literal.
tab	Specify whether to send a tab character after the source string. Optional paramater. Must be either "True" or "False".
linefeed	Specify whether to send a linefeed character after the source string. Optional paramater. Must be either "True" or "False".
dest	Specify where to save the text/character data. Dest is assumed to be a filename. Optional paramater.

6.1.2 Output Source: Math

Fixed Alias: math

Used to perform mathematical calculations on values.

6.1.2.1 Add

```
<Task name="math:Add" x="..." y="..." storeIn="..." />
```

Adds two values/variables together and stores the result in another variable.

Parameters:

x	The first value to add. Can be a literal value or a variable reference.

y	The second value to add. Can be a literal value or a variable reference.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.2.2 Subtract

```
<Task name="math:Subtract" x="..." y="..." storeIn="..." />
```

Subtracts one value/variable from another value/variable and stores the result in another variable.

Parameters:

x	The first value to subtract from. Can be a literal value or a variable reference.
y	The second value to subract. Can be a literal value or a variable reference.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.2.3 Multiply

```
<Task name="math:Multiply" x="..." y="..." storeIn="..." />
```

Multiplies two values/variables together and stores the result in another variable.

Parameters:

x	The first value to multiply. Can be a literal value or a variable reference.
y	The second value to multiply. Can be a literal value or a variable reference.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.2.4 Divide

```
<Task name="math:Divide" x="..." y="..." storeIn="..." />
```

Divides one value/variable from another value/variable together and stores the result in another variable.

Parameters:

x	The first value to divide from. Can be a literal value or a variable reference.
y	The second value to divide. Can be a literal value or a variable reference.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The

variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3 Output Source: String

Fixed Alias: str

Used to perform manipulation on string data.

6.1.3.1 Append

```
<Task name="str:Append" source="..." value="..." storeIn="..." />
```

Appends a string to another string and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
value	Specify what to append to *source*. Can optionally be a variable reference. If not a variable, *value* is assumed to be a literal string.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3.2 Pad

```
<Task name="str:Pad" source="..." direction="..." length="..." char="..."
storeIn="..." />
```

Pads a string either to the left or right with extra characters and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
direction	Specify which direction to pad the string to. Can either be "Left", "Right" or "Both".
length	Specify the maximum length of the resulting string.
char	Specify which character to use as the padding character.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3.3 Replace

```
<Task name="str:Replace" source="..." what="..." with="..." storeIn="..." />
```

Replaces all occurances of one substring with another and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
what	Specify what substring should be replaced. Can optionally be a variable reference. If not a variable, *what* is assumed to be a literal string.
with	Specify what substring should be the replacement. Can optionally be a variable reference. If not a variable, *with* is assumed to be a literal string.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3.4 Length

```
<Task name="str:Length" source="..." storeIn="..." />
```

Gets the length of a string and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3.5 Strip

```
<Task name="str:Strip" source="..." remove="..." storeIn="..." />
```

Removes all occurances of the characters listed from a string and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
remove	Specify a character or string of characters to strip out of *source*.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does

not exist a new variable will be created.

6.1.3.6 SubString

```
<Task name="str:SubString" source="..." start="..." length="..." storeIn="..." />
```

Returns a substring from a larger string and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
start	Specify what index to start the substring from within *source*. Indexes start at zero.
length	Specify the length of the substring.
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.3.7 Trim

```
<Task name="str:Trim" source="..." direction="..." storeIn="..." />
```

Trims whitespace from a string and stores the result in a variable.

Parameters:

source	Specify what input data to use. Can optionally be a variable reference. If not a variable, *source* is assumed to be a literal string.
direction	Specify which direction to trim the string. Can either be "Left", "Right" or "Both".
storeIn	Specify what variable name to use to store the result. Must supply a variable reference. The variable can be an existing variable (in which case the original value is overwritten), or if it does not exist a new variable will be created.

6.1.4 Output Source: Variables

Fixed Alias: var

Used to perform manipulation on variables.

6.1.4.1 Copy

```
<Task name="var:Copy" source="..." to="..." />
```

Copies the data from one variable into a new variable.

Parameters:

source	Specify what input data to use. Must be a variable reference, e.g. **$ref**.
to	Specify where to copy the data to. Must be a variable reference, e.g. **$ref**.

6.1.4.2 Set

```
<Task name="var:Set" value="..." to="..." />
```

Sets the value of a variable or creates a new variable if one does not already exist.

Parameters:

value	The value to set.
to	Specify where to set the data to. Must be a variable reference, e.g. **$ref**.

6.1.5 Output Source: Utility

Fixed Alias: util

Used to perform utility operations.

6.1.5.1 SelectApplication

```
<Task name="util:SelectApplication" applicationName="..." />
```

Finds an application which matches the applicationName parameter. The application will be maximized and brought to the foreground.

Parameters:

applicationName	Name to find amongst the currently active/visible applications.

6.2 Output Source: Keyboard

Module/DLL Filename: WedgeOutputSource_Keyboard.dll

Outputs strings to the keyboard by virtually pressing keystrokes.

Attention:

The tasks in this module will type to whatever window has current focus. Be aware of this when testing as you do not want to overwrite important data! The util:SelectApplication task can be used to bring an application to the foreground within a START_OF_DATA event.

Note:

The Alt Gr key on a keyboard can be simulated by pressing both the Ctrl and Alt keys at the same time.

Syntax

```
<OutputModules>
      <OutputModule name="Keyboard" alias="...">
            <Parameters>
                  <Parameter name="..." value="..." />
                  ...
            </Parameters>
      </OutputModule>
</OutputModules>
```

Parameters

Name	Optional?	Description
DelayBetweenKeys	Yes	Sets the delay to wait between key presses, measured in milliseconds. Default value is 10 ms.

6.2.1 PressCharKey

```
<Task name="keyb:PressCharKey" key="..." ctrl="..." alt="..." />
```

Presses a character key.

Parameters:

key	Specify what character key to press. Cannot be an ASCII control character. The need for the shift key will be determined by which character is entered, e.g. if "A" is used, then the shift key will be pressed along with the "A" key to get an uppercase letter.
ctrl	Specify whether to press the Control key at the same time. Must be either "True" or "False".
alt	Specify whether to press the Alt key at the same time. Must be either "True" or "False".

6.2.2 PressSpecialKey

```
<Task name="keyb:PressSpecialKey" key="..." shift="..." ctrl="..." alt="..."
repeat="..." />
```

Presses a special key that does not have an ASCII character associated with it.

Parameters:

key Specify what character key to press. Can be one of the following values:

- BACKSPACE
- TAB
- RETURN
- CAPS_LOCK
- ESC
- SPACE
- PAGE_UP
- PAGE_DOWN
- END
- HOME
- LEFT
- UP
- RIGHT
- DOWN
- PRINT_SCRN
- INSERT
- DELETE
- NUM_PAD_0
- NUM_PAD_1
- NUM_PAD_2
- NUM_PAD_3
- NUM_PAD_4
- NUM_PAD_5
- NUM_PAD_6
- NUM_PAD_7
- NUM_PAD_8
- NUM_PAD_9
- MULTIPLY
- ADD
- SEPARATOR
- SUBTRACT
- DECIMAL
- DIVIDE
- F1
- F2
- F3
- F4
- F5
- F6
- F7

- F8
- F9
- F10
- F11
- F12
- NUM_LOCK
- SCROLL_LOCK

shift	Specify whether to press the Shift key at the same time. Must be either "True" or "False".
ctrl	Specify whether to press the Control key at the same time. Must be either "True" or "False".
alt	Specify whether to press the Alt key at the same time. Must be either "True" or "False".
repeat	Specify number of times to repeat this key. Optional parameter. Must be a number within "".

6.2.3 TypeString

```
<Task name="keyb:TypeString" source="..." clr="..." />
```

Types a string value/variable in full.

Parameters:

source	Specify what to type. Can be a variable reference. If it is not a variable, then it assumes that *source* is a string literal.
clr	Specify whether to send a Backspace key to first clear the preselected text. Optional paramater. Must be either "True" or "False".

6.3 Output Source: Internet Explorer Web Page

Module/DLL Filename: WedgeOutputSource_IEPage.dll

Outputs strings and images to a web page that is displayed in Internet Explorer. This is acheived through COM interfaces to connect to an open IE window/page.

Attention:

The tasks in this module will only work in Internet Explorer, not any other browser.

Syntax

```
<OutputModules>
      <OutputModule name="IEPage" alias="..." />
</OutputModules>
```

6.3.1 Attach

```
<Task name="page:Attach" mode="..." param="..." />
```

Attaches the Output Wedge to an open web page that matches the given criteria.

Attention:

This is required before carrying out any other IEPage task.

Parameters:

mode Specify how to attach to the web page. The following modes can be used:

- "First" - connect to the first IE window found running on the system. Does not require a parameter.
- "Title" - connect to an IE window that is currently displaying a page with a given title. Requires a parameter,
- "URL" - connect to an IE window that is currently displaying a page with the given URL. Requires a parameter.

param If *mode* is "Title" or "URL", specify the extra parameter required (can be a variable reference):

- For "Title", *param* must be the full title of the web page.
- For "URL", *param* must be the full URL of the web page.

6.3.2 Detach

```
<Task name="page:Detach" />
```

Detaches the Output Wedge from a connected web page.

6.3.3 InvokeJScriptFunc

```
<Task name="page:InvokeJScriptFunc" funcName="..." params="..." />
```

Invokes a JScript function that the web page defines.

Parameters:

funcName	Specify the JScript function name to invoke. Can be a variable reference. If not a variable, it assumes that *funcName* is the literal function name to use.
params	Specify any parameters that are required by the function. This is a list of values separated by commas. Can be a variable reference. If not a variable, it assumes that *params* is the literal comma-text list to use.

6.3.4 SetImage

```
<Task name="page:SetImage" lookupAttr="..." lookupValue="..." source="..."
data="..." default="..."/>
```

Finds an image (IMG) tag in the current page and sets the tags `src` attribute to modify the image displayed in it.

Parameters:

lookupAttr	Specify what kind of attribute to use when searching for the right tag to update. Typically you should use attributes like `id` or `name`. Can be a variable reference. If not a variable, it assumes that *lookupAttr* is a literal string.
lookupValue	Specify what value to match against the *lookupAttr*. Can be a variable reference. If not a variable, it assumes that *lookupValue* is a literal string.
source	Specify what to update in the IMG tag. This should be a local file name relative to the page (or a full file path). Can be a variable reference. If not a variable, it assumes that *source* is a literal string.
data	Name of the data source to check to decide whether to display default image. If the data is empty the *default* image will be displayed, otherwise the *source* image will be displayed.
default	Name of the image to display if data source specified in *data* is empty

6.3.5 SetJScriptVar

```
<Task name="page:SetJScriptVar" varName="..." value="..." index="..." />
```

Sets a value to a JScript variable/array index.

Note:

In JavaScript, objects with properties can be accessed using array notation, so this is one way of setting object properties.

Parameters:

varName	Specify the name of the JScript variable to set. Can be a variable reference. If not a variable, it assumes that *varName* is a literal string.
value	Specify what value to set to the JScript variable. Can be a variable reference. If not a variable, it assumes that *value* is a literal string.
index	If *varName* represents an array/object, specify the index of the array/object to access. The index can be an integer or a string. Can be a variable reference. If not a variable, it assumes that *index* is a literal string. If *varName* is a scalar variable, then do not include this parameter.

6.3.6 SetTagAttribute

```
<Task name="page:SetTagAttribute" lookupAttr="..." lookupValue="..." attr="..."
value="..." />
```

Sets a HTML tag's attribute to a specified value.

Parameters:

lookupAttr	Specify what kind of attribute to use when searching for the right tag to update. Typically you should use attributes like `id` or `name`. Can be a variable reference. If not a variable, it assumes that *lookupAttr* is a literal string.
lookupValue	Specify what value to match against the *lookupAttr*. Can be a variable reference. If not a variable, it assumes that *lookupValue* is a literal string.
attr	Specify which attribute in the tag to actually modify. Can be a variable reference. If not a variable, it assumes that *attr* is a literal string.
value	Specify the value to set to *attr*. Can be a variable reference. If not a variable, it assumes that *attr* is a literal string.

6.3.7 SetTagContent

```
<Task name="page:SetTagContent" lookupAttr="..." lookupValue="..." value="..." />
```

Sets a HTML tag's content (the value between <Tag>...</Tag>) to a specified value.

Parameters:

lookupAttr	Specify what kind of attribute to use when searching for the right tag to update. Typically you should use attributes like `id` or `name`. Can be a variable reference. If not a variable, it assumes that *lookupAttr* is a literal string.
lookupValue	Specify what value to match against the *lookupAttr*. Can be a variable reference. If not a variable, it assumes that *lookupValue* is a literal string.
value	Specify the value to set to the tag's content. Can be a variable reference. If not a variable, it assumes that *attr* is a literal string.

6.4 Output Source: Swipe Protocols

Module/DLL Filename: WedgeOutputSource_SwipeProtocols.dll

Uses the 3M Page Reader SDK DLL **SwipeProtocolsDll.dll** to create messages that are formatted to a particular swipe protocol (e.g. RTE Native, ARINC MUSE, SITA CUTE, etc).

Syntax

```
<OutputModules>
      <OutputModule name="SwipeProtocols" alias="..." />
```

```
</OutputModules>
```

6.4.1 CreateProtocol

```
<Task name="proto:AddToMessage" id="..." protocol="..." rteUseBCC="..." />
```

Creates a new protocol manager that subsequent tasks will use.

Note:

You need to use this task before using any of the other tasks; all the other tasks relate back to the protocol builder that this task creates.

Parameters:

id	Specify what identifier to use when using different protocols. This value is then used in other tasks to access the protocol builder that is created by this task.
protocol	Specify what swipe protocol to use. Can be one of the following: • RTE • MUSE • SITA_DIRECT • SITA_INDIRECT • SITE_BGR
rteUseBCC	If *protocol* is set to "RTE", specify whether the protocol builder should manage the BCC checksum for messages. Should be either "True" or "False. If not required, do not include this parameter,

6.4.2 AddToMessage

```
<Task name="proto:AddToMessage" id="..." group="..." source="..." line="..."
track="..." block="..." barcodeType="..." />
```

Adds some data to a protocol message.

Parameters:

id	Specify which protocol builder to add the data to. This should be the same ID used when the CreateProtocol task was used.
group	Specify what group this data is for. Can be one of the following: • OCR • MSR • ATB • Barcode

source Specify what data to add to the message. Can be a variable reference. If not a variable, it assumes that *source* is the literal function name to use.
line (OCR only) Specify what OCR line *source* should be added to.
track (MSR and ATB only) Specify what track *source* should be added to.
block (ATB only) Specify what track block *source* should be added to.
barcodeType (Barcodes only) Specify what kind of barcode data *source* is. Can be one of the following:

- Industrial2of5
- Interleaved2of5
- IATA2of5
- Code39
- Code128
- PDF417
- Aztec
- QRCode
- DataMatrix

6.4.3 GetMessage

```
<Task name="proto:GetMessage" id="..." group="..." storeIn="..." />
```

Get the complete data message formatted to a protocol that can be sent to something else.

Note:
You need to add data to the message first using the AddToMessage task. The cumulative data will then be inserted into a correctly formatted message which this task will return.

Parameters:
id Specify which protocol builder to get the message from. This should be the same ID used when the CreateProtocol task was used.
group Specify what group this data is for. Can be one of the following:

- OCR
- MSR
- ATB
- Barcode

storeIn Specify where to store the result. Must be a variable reference.

6.4.4 ResetMessage

```
<Task name="proto:ResetMessage" id="..." group="..." />
```

Resets all data currently stored within a protocol message.

Parameters:

id Specify which protocol builder to reset. This should be the same ID used when the CreateProtocol task was used.

group Specify what group this data is for. Can be one of the following:

- OCR
- MSR
- ATB
- Barcode

6.5 Output Source: Virtual Serial Port

Module/DLL Filename: WedgeOutputSource_VirtualSerialPort.dll

Uses the 3rd Party ActiveX control from Eltima to set up virtual COM ports to transmit data to any listening device.

Attention:

You will need to install the ActiveX control and drivers to be able to use this output source and to compile the source code. The installer can be obtained from 3M. It is found in source repository under **$/3rd Party/Installers/Eltima/VSPSetup/VSPSetup.exe**

Syntax

```
<OutputModules>
      <OutputModule name="VirtualSerialPort" alias="..." />
</OutputModules>
```

6.5.1 OpenPort

```
<Task name="vcom:OpenPort" port="..." setCTS="..." setDCD="..." setDSR="..."
setRING="..." mustHaveDTR=".." mustHaveRTS="..." />
```

Opens a new or existing virtual COM port.

Attention:

This task must be run before using any other task.

Parameters:

port	Specify the port number to use.
setCTS	Specify whether to set the CTS signal on the port. Must be either "True" or "False". Defaults to "False".
setDCD	Specify whether to set the DCD signal on the port. Must be either "True" or "False". Defaults to "False".
setDSR	Specify whether to set the DSR signal on the port. Must be either "True" or "False". Defaults to "False".
setRING	Specify whether to set the RING signal on the port. Must be either "True" or "False". Defaults to "False".
mustHaveDTR	Specify whether to only transmit if the DTR signal at the other end is set. Must be either "True" or "False". Defaults to "False".
mustHaveRTS	Specify whether to only transmit if the RTS signal at the other end is set. Must be either "True" or "False". Defaults to "False".

6.5.2 ClosePort

```
<Task name="vcom:ClosePort" port="..." />
```

Close an open virtual COM port.

Parameters:

port	Specify the port number to close.

6.5.3 WriteToPort

```
<Task name="vcom:WriteToPort" port="..." source="..." />
```

Writes some data to an open virtual COM port.

Parameters:

port	Specify the port number to write to.
source	Specify what data to write. Can be a variable reference. If not a variable, it assumes that *source* is a literal string.