FICO Application Studio/Optimization Executor Integration Package

8.3

REFERENCE MANUAL

FICO® Xpress Optimization

FICO

©2014–2026 Fair Isaac Corporation. All rights reserved. This documentation is the property of Fair Isaac Corporation ("FICO"). Receipt or possession of this documentation does not convey rights to disclose, reproduce, make derivative works, use, or allow others to use it except solely for internal evaluation purposes to determine whether to purchase a license to the software described in this documentation, or as otherwise set forth in a written software license agreement between you and FICO (or a FICO affiliate). Use of this documentation and the software described in it must conform strictly to the foregoing permitted uses, and no other use is permitted.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, please report them to us in writing. Neither FICO nor its affiliates warrant that this documentation is error-free, nor are there any other warranties with respect to the documentation except as may be provided in the license agreement. FICO and its affiliates specifically disclaim any warranties, express or implied, including, but not limited to, non-infringement, merchantability and fitness for a particular purpose. Portions of this documentation and the software described in it may contain copyright of various authors and may be licensed under certain third-party licenses identified in the software, documentation, or both.

In no event shall FICO or its affiliates be liable to any person for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this documentation or the software described in it, even if FICO or its affiliates have been advised of the possibility of such damage. FICO and its affiliates have no obligation to provide maintenance, support, updates, enhancements, or modifications except as required to licensed users under a license agreement.

FICO is a registered trademark of Fair Isaac Corporation in the United States and may be a registered trademark of Fair Isaac Corporation in other countries. Other product and company names herein may be trademarks of their respective owners.

Patent(s): www.fico.com/en/patents

FICO® Xpress Optimization 9.9

Deliverable Version: A

Last Revised: 14 April, 2016

Contents

1 Introduction 1
1.1 XML record format . . . 1

2 Package functionality 3
2.1 Constants . . . 3
2.2 Subroutines . . . 3
appsxmlexport . . . 4
appsxmlfnin . . . 5
appsxmlfnout . . . 6
appsxmlimport . . . 7
appsxmlin . . . 8
appsxmlout . . . 9

Appendix 10

A Contacting FICO 10
FICO Customer Support . . . 10
Documentation . . . 10
FICO Learning . . . 11
Sales and maintenance . . . 11
About FICO . . . 11

Index 12

CHAPTER 1

Introduction

The helper package *fssappstudio* provides a set of routines to transform input and result data of a Mosel model into the XML record format of FICO Application Studio. This package is imported by adding the following line to the top of your model:

```
uses "fssappstudio"
```

1.1 XML record format

The XML record format used by FICO Application Studio has the following structure: The top-level element is `InputRecordList` (for inputs) or `ResultRecordList` (for results), which always contains a single `InputRecord` or `ResultRecord`. Data comes in the form of scalars or arrays.

- Scalars are represented as values enclosed in a named element, as shown in the example below.
- Arrays are represented as a 'table' comprised of a named element which contains multiple 'rec' (`_rec`) elements that represent rows of the table. Each table row contains multiple elements representing individual cells. The following example has a table named 'Channels' with three columns: 'Channel', 'Cost', and 'Capacity'.

```
<?xml version="1.0"?>
<InputRecordList xmlns="http://www.fico.com/input"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <InputRecord>
    <MyScalarValue>7.5</MyScalarValue>
    <AnotherScalar>a string value</AnotherScalar>
    <Channels>
      <Channels_rec>
        <Channel>1</Channel>
        <Cost>0.5</Cost>
        <Capacity>100</Capacity>
      </Channels_rec>
      <Channels_rec>
        <Channel>2</Channel>
        <Cost>1.2</Cost>
        <Capacity>50</Capacity>
      </Channels_rec>
      <Channels_rec>
        <Channel>3</Channel>
        <Cost>3</Cost>
        <Capacity>40</Capacity>
      </Channels_rec>
    </Channels>
  </InputRecord>
</InputRecordList>
```

In a Mosel model, data input from such XML files needs to be preceeded by a call to `appsxmlimport` to register the inputs:

```
declarations
  myScalar: real
  ChannelCost,ChannelCapacity: array(CHANNELS: range) of real
end-declarations

appsxmlimport
initialisations from appsxmlin("MyScalarValue")
  myScalar as "[](MyScalarValue)"
end-initialisations
initialisations from appsxmlin("Channels")
  [ChannelCost,ChannelCapacity] as "[](Channel,Cost,Capacity)
end-initialisations
```

Inversely, data output from a Mosel model needs to be terminated with a call to `appsxmlexport` to generate the XML format data and the corresponding XSD schema definition from the output data:

```
declarations
  ResultCost,ResultCapacity: array(CHANNELS: range) of real
end-declarations

initialisations appsxmlout("ResultChannels")
  [ChannelCost,ChannelCapacity] as "[](ResultChannel,ResultCost,ResultCapacity)"
end-initialisations
appsxmlexport
```

Please note that array data needs to be specified in *sparse format*, that is, all indices are specified along with the value columns.

CHAPTER 2

Package functionality

2.1 Constants

The following publicly declared objects can be employed by Mosel models that load the package *fssappstudio*:

APPSXML_FMT_TYPED_RECORDLIST = 1

Normal XML format, use `InputRecordList/InputRecord` or `ResultRecordList/ResultRecord` elements

APPSXML_FMT_UNTYPED_RECORDLIST = 2

Untyped XML format, use `RecordList/Record` elements

2.2 Subroutines

`appsxmlexport`	Build XML from CVS files.	p. 4
`appsxmlfnin`	Generate a file name for an input file.	p. 5
`appsxmlfnout`	Generate a file name for an output file.	p. 6
`appsxmlimport`	Extract all data files included in provided XML.	p. 7
`appsxmlin`	Generate the extended filename for 'initialisations from' from a label.	p. 8
`appsxmlout`	Generate the extended filename for 'initialisations to' from a label.	p. 9

appsxmlexport

Purpose

Build XML from CVS files.

Synopsis

```
procedure appsxmlexport(fname:string, type:integer)
procedure appsxmlexport(fname:string)
procedure appsxmlexport
```

Arguments

`fname` name of the XML output file, defaults to ”result”

`type` entity label

`APPSXML_FMT_TYPED_RECORDLIST` use `ResultRecordList`/ `ResultRecord` elements (default)

`APPSXML_FMT_UNTYPED_RECORDLIST` use `RecordList`/ `Record` elements

Example

See function `appsxmlout`.

Further information

1. This function transforms result data that has been generated by a Mosel model using `initializations to` with `appsxmlout` to the XML result data format of *fssappstudio*.
2. `appsxmlexport` also generates the XSD schema that corresponds to the resulting XML file.
3. FICO Application studio expects the fixed filename `result` for XML output data, using a single file per model.
4. The reverse operation, transforming XML format input data to CSV files is performed by `appsxmlimport`.

Related topics

`appsxmlout`, `appsxmlimport`

appsxmlfnin

Purpose

Generate a file name for an input file.

Synopsis

```
function appsxmlfnin(label:string):string
```

Argument

`label` entity label

Return value

Filename or `null`: if the file has not been generated.

Example

The following line of Mosel code outputs the full path to a file called `myScalar` in a temporary directory created by this package.

```
writeln(appsxmlfnin("myScalar"))
```

Further information

This function returns the name of a temporary input file as is used by `appsxmlin`. It needs to be preceeded by a call to `appsxmlimport`.

Related topics

`appsxmlin`, `appsxmlimport`

appsxmlfnout

Purpose

Generate a file name for an output file.

Synopsis

```
function appsxmlfnout(label:string):string
```

Argument

`label` entity label

Return value

Filename.

Example

The following line of Mosel code outputs the full path to a file called `myScalar` in a temporary directory created by this package.

```
writeln(appsxmlfnout("myScalar"))
```

Further information

This function generates and records the name of a temporary output file as is used by `appsxmlout`. If the file already exists it is deleted.

Related topics

`appsxmlout`

appsxmlimport

Purpose

Extract all data files included in provided XML.

Synopsis

```
procedure appsxmlimport(fname:string, type:integer)
procedure appsxmlimport(fname:string)
procedure appsxmlimport
```

Arguments

`fname` name of an XML input file, defaults to ”input”

`type` data structure type

`APPSXML_FMT_TYPED_RECORDLIST` use `InputRecordList`/ `InputRecord` elements (default)

`APPSXML_FMT_UNTYPED_RECORDLIST` use `RecordList`/ `Record` elements

Example

See function `appsxmlin`.

Further information

1. This function prepares input data provided in *fssappstudio* XML format for reading them via `initializations from` using `appsxmlin` by transforming the XML data into a set of temporary CSV format files (one per entity).
2. `appsxmlimport` also generates the XSD schema that corresponds to the provided XML file.
3. FICO Application studio expects the fixed filename `input` for XML input data, using a single file per model.
4. The reverse operation, transforming CSV format data into the XML result data format expected by *fssappstudio* is performed by `appsxmlexport`.

Related topics

`appsxmlin`, `appsxmlexport`

appsxmlin

Purpose

Generate the extended filename for 'initialisations from' from a label.

Synopsis

```
function appsxmlin(label:string):string
```

Argument

`label` entity label

Return value

An extended CSV filename

Example

The following lines of Mosel code

```
declarations
  myStr: string
end-declarations

appsxmlimport("input")
initialisations from appsxmlin("AnotherScalar")
  myStr as "[](AnotherScalar)"
end-initialisations
```

read the value of the scalar from an input file of this form:

```
<InputRecordList>
  <InputRecord>
    <MyScalarValue>7.5</MyScalarValue>
    <AnotherScalar>a string value</AnotherScalar>
  </InputRecord>
</InputRecordList>
```

Further information

1. This function generates the extended file name to be used by `initializations from` for the specified label.
2. Data input from the resulting file needs to be preceeded by a call to `appsxmlimport`.

Related topics

`appsxmlimport`

appsxmlout

Purpose

Generate the extended filename for 'initialisations to' from a label.

Synopsis

```
function appsxmlout(label:string):string
```

Argument

`label` entity label

Return value

An extended CSV filename

Example

The following lines of Mosel code

```
declarations
  myArr: array(range) of real
end-declarations

myArr::[1..3][1.2, 3.5, 6.7]
initialisations to appsxmlout("AnArray")
  myArr as "[](ArIndex,ArValue)"
end-initialisations
appsxmlexport("result")
```

output the array `myArr` to a result file of this form:

```
<ResultRecordList>
  <ResultRecord>
    <AnArray>
      <AnArray_rec>
        <ArIndex>1</ArIndex>
        <ArValue>1.2</ArValue>
      </AnArray_rec>
      <AnArray_rec>
        <ArIndex>2</ArIndex>
        <ArValue>3.5</ArValue>
      </AnArray_rec>
      <AnArray_rec>
        <ArIndex>3</ArIndex>
        <ArValue>6.7</ArValue>
      </AnArray_rec>
    </AnArray>
  </ResultRecord>
</ResultRecordList>
```

Further information

1. This function generates the extended file name to be used by `initializations to` for the specified label.
2. Data output to the resulting file needs to be terminated by a call to `appsxmlexport` in order to generate the XML file.

Related topics

`appsxmlexport`

Appendix A

Contacting FICO

FICO provides clients with support and services for all our products.

FICO Customer Support

FICO Customer Support offers technical support and services ranging from self-help tools to direct assistance with a FICO technical support engineer. Support is available to all clients who have an active maintenance contract.

The FICO Customer Self-Service Portal (support.fico.com) is a secure web portal that allows users to open, review, and update their support cases; manage their organization's portal users; find solutions to common problems in the FICO Knowledge Base; and view the availability of their cloud applications 24 hours a day, 7 days a week.

You can find support contact information and a link to the FICO Customer Self-Service Portal (online support) on the Product Support home page (www.fico.com/en/product-support).

Please include *'Xpress'* in the subject line of your support queries.

Documentation

FICO continually looks for new ways to improve and enhance the value of the products and services we provide.

If you have comments or suggestions regarding how we can improve this documentation, let us know by sending your suggestions to techpubs@fico.com. Please include your contact information (name, company, email address, and optionally, your phone number) so we may reach you if we have questions.

FICO Learning

FICO Learning is the principal provider of product training for our clients and partners. FICO Learning offers instructor-led classroom courses, web-based training, seminars, and training tools for both new user enablement and ongoing performance support.

For additional information, visit the FICO Learning home page at www.fico.com/en/product-training or email producteducation@fico.com.

Sales and maintenance

If you need information on other Xpress Optimization products, or you need to discuss maintenance contracts or other sales-related items, contact FICO by:

- Phone: +1 (408) 535-1500 or +44 207 940 8718
- Web: www.fico.com/optimization and use the available contact forms

About FICO

FICO (NYSE:FICO) is a leading analytics software company, helping businesses in 90+ countries make better decisions that drive higher levels of growth, profitability, and customer satisfaction. Learn more at www.fico.com or contact us at www.fico.com/en/contact-us.

Index

A
`APPSXML_FMT_TYPED_RECORDLIST`, 3
`APPSXML_FMT_UNTYPED_RECORDLIST`, 3
`appsxmlexport`, 4
`appsxmlfnin`, 5
`appsxmlfnout`, 6
`appsxmlimport`, 7
`appsxmlin`, 8
`appsxmlout`, 9

C
CSV format data
    generate, 7

E
export data, 4

F
file name
    input file, 5
    result file, 6

I
import data, 7
input file
    extended file name, 8
    file name, 5
`InputRecord`, 1
`InputRecordList`, 1

O
output file, *see* result file
    extended file name, 9

R
result file
    file name, 6
`ResultRecord`, 1
`ResultRecordList`, 1

S
sparse format, 2

X
XML format data
    generate, 4