ISO 8583 Extensions 1.0

In this section:

About the Extensions

The ISO 8583 Extensions are custom extensions for Parasoft's continuous testing solutions. On the client side, they can be used to send and receive custom ISO 8583 messages. On the server side, they can be used to create and send custom virtualized responses.

The following custom extensions are provided for ISO 8583 integration:

- **ISO 8583 Message Format:** A message format used to convert from native ISO to XML ISO and vice versa.
- **ISO 8583 Transport:** A transport protocol for sending and receiving custom ISO 8583 messages.
- **ISO 8583 Message Listener:** A message listener used to listen for custom ISO 8583 messages and generate virtual responses.

Implementation

This extension suite is implemented as com.parasoft.soavirt.iso8583.main-<version>.jar, which depends upon the following jars provided with the distribution:

jPOS

- jpos-1.9.2.jar

JDOM

- jdom-1.1.3.jar

Apache Commons CLI

- commons-cli-1.2.jar

ISO-8583 Extensions

- com.parasoft.soavirt.messages.iso8583-<version>.jar
- com.parasoft.soavirt.transport.iso8583-<version>.jar
- com.parasoft.virtualize.listener.iso8583-<version>.jar

Requirements

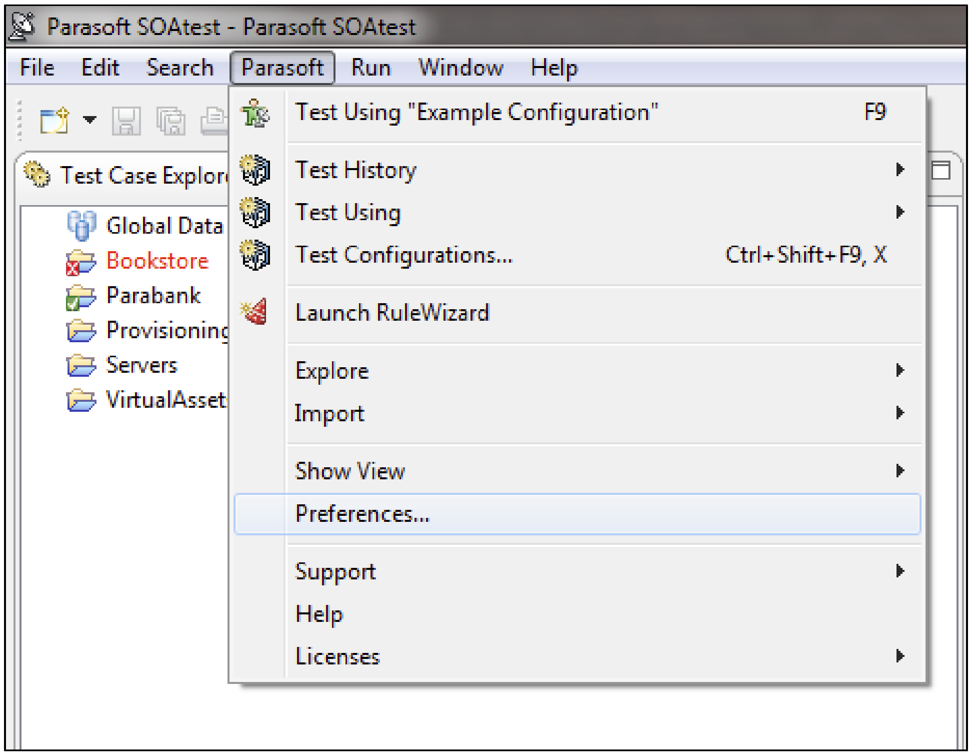
- SOAtest and/or Virtualize 9.5.x or higher.

Installation

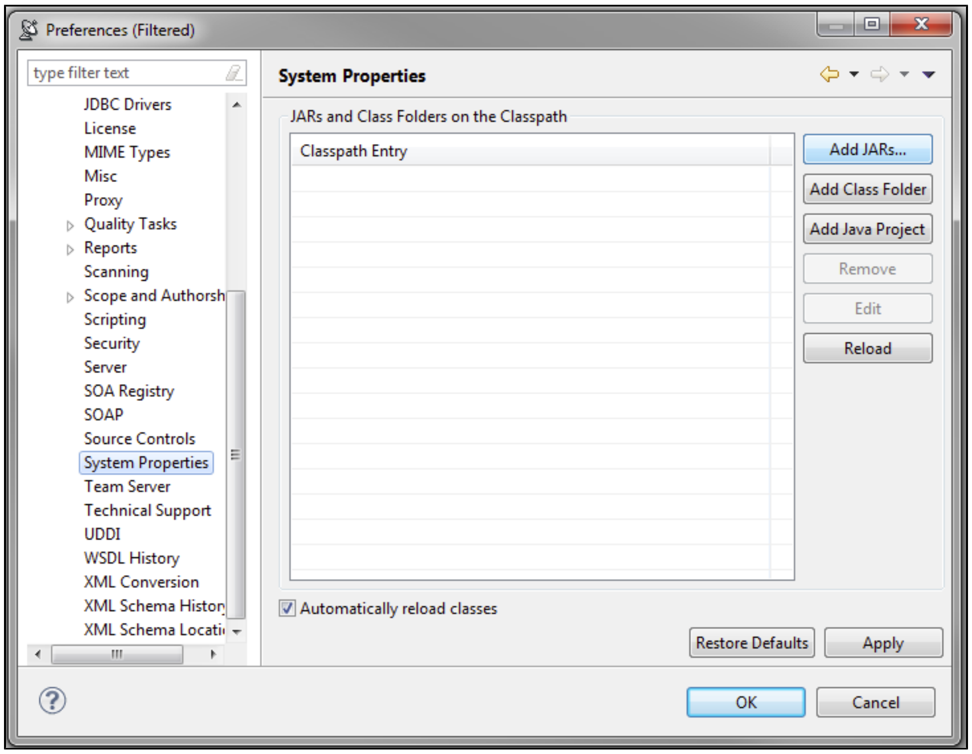
The extensions can be installed from the UI or command line.

UI Installation

1. Choose **Parasoft > Preferences**.



2. In the System Properties preferences page, click **Add JARs**.



3. In the file chooser that opens, select com.parasoft.soavirt.iso8583.main-<version>.jar. All required dependencies will be loaded.
4. Restart SOAtest/Virtualize.

Command Line Installation

Add the com.parasoft.soavirt.iso8583.main-<version>.jar file to the `system.properties.classpath` property in your localsettings properties file.

For example:

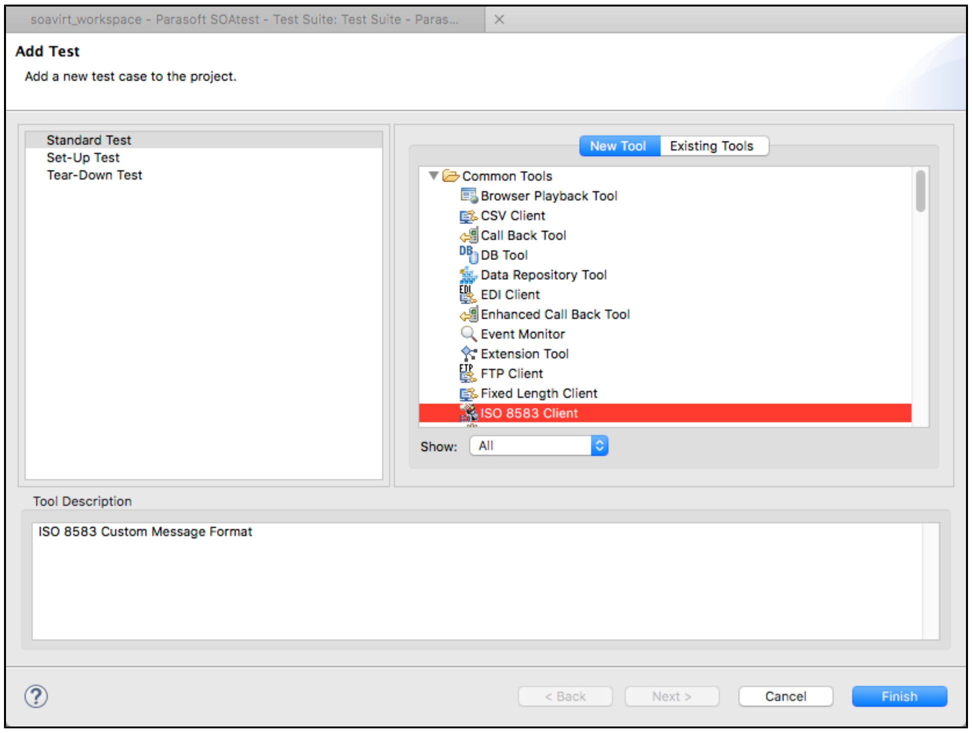
```
system.properties.classpath=<path to jar>/com.parasoft.soavirt.iso8583.main-1.0.0.jar
```

Once the classpath is modified, all of the required dependencies will be loaded.

Usage

SOAtest

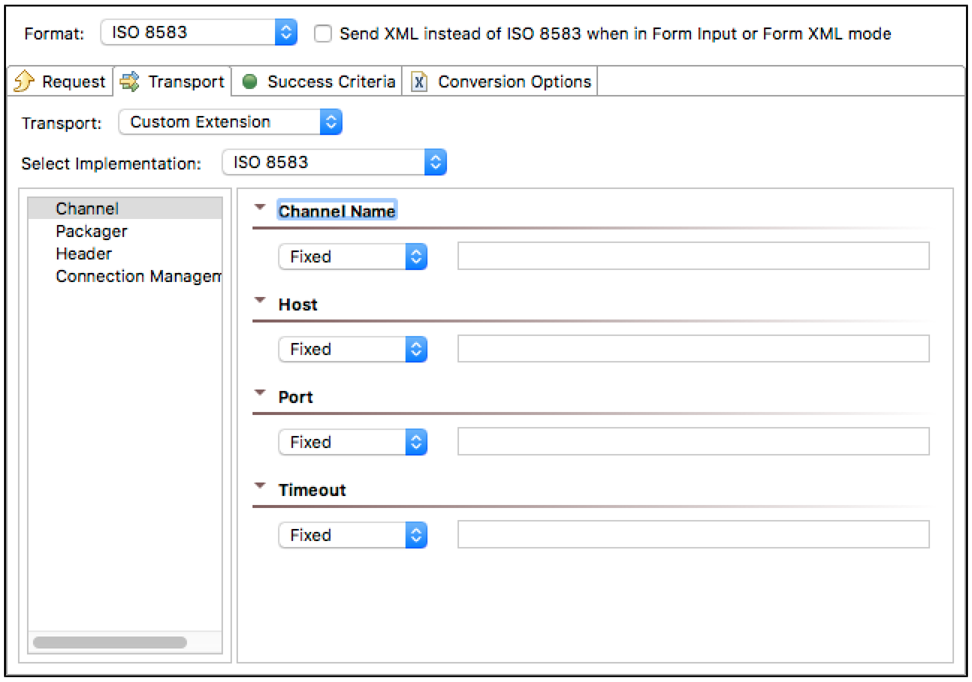
1. Right-click on a test suite and choose **Add New> Test.**
2. Two ISO 8583 clients with the same name will be available in the Add Test wizard. Choose **ISO 8583 Client** in the Common Tools category and click **Finish**. The other ISO Client is the built-in tool shipped with SOAtest that the ISO 8583 extensions replace.



ISO 8583 Custom Message Format

3. Click the **Transport** tab and choose **Custom Extension** from the Transport drop-down menu.

4. Choose **ISO 8583** from the Select Implementation drop-down menu to configure the tool. Packager options are configured on both the Conversion Options tab and in the transport settings under Packager Settings.



Format: ISO 8583

Send XML instead of ISO 8583 when in Form Input or Form XML mode

Transport: Custom Extension

Select Implementation: ISO 8583

Channel Name: Fixed

Host: Fixed

Port: Fixed

Timeout: Fixed

ISO 8583 Message Listener Settings

You can configure the following settings for the ISO 8583 Message Listener

Channel Settings

Channel Name	Defines the ISO 8583 channel to use when sending/receiving messages. All the default channel implementations (e.g., channel implementations that need only a port, host, and packager) included in the jPOS library are available for use. For details, see Channels.
Host	Defines the host to use when making connections.
Port	Defines the port to use when making connections.
Timeout	Defines how many seconds to wait before timing a connection out.

Packager Settings

Packager Name	Defines the packager that will be used to pack and unpack the ISO 8583 messages. Generally, GenericPackager will be used here along with a generic packager XML description. However, all packagers mentioned in the Packagers section are available for use if a predefined ISO packager is more suitable. If a custom ISO packager is implemented (e.g., ISOPackager interface), it must be included on the SOAtest/Virtualize classpath and the fully-qualified class name must be provided (for example org.jpos.iso.packager. GenericPackager). For details, see Packagers.
Packager Path	Defines the path to a generic packager description XML file. This field is only used for generic packagers (e.g., GenericPackager and X92GenericPackager).

Header Settings

Header Length	Specifies the header length for the outgoing response.

Connection Management Settings

Keep connection alive	Enable this option to keep the client connection alive and reused for subsequent publishing.

Close connection after test execution	Enable this option to close the client connection directly after publishing.

ISO 8583 Message Format Settings

You can configure the following settings for the ISO 8583 Message Format.

Packager Name	Defines the packager used to pack and unpack the ISO 8583 messages. Generally, GenericPackager will be used here along with a generic packager XML description. However, all the packagers mentioned in the Packagers section are available for use if a predefined ISO packager is more suitable. If a custom ISO packager is implemented (e.g., ISOPackager interface), it must be included on the SOAtest/Virtualize classpath and the fully-qualified class name must be provided (for example org.jpos.iso.packager.GenericPackager). For details, see Packagers.
Packager Path	Defines the path to a generic packager description XML file. This field is only used for generic packagers (e.g., GenericPackager and X92GenericPackager)

ISO 8583 Transport Settings

You can configure the following settings for the ISO 8583 Transport.

Channel Settings

Channel Name	Defines the ISO 8583 channel to use when sending/receiving messages. All the default channel implementations (e.g., channel implementations that need only a port, host, and packager) included in the jPOS library are available for use. For details, see Channels.
Host	Defines the host to use when making connections.
Port	Defines the port to use when making connections.
Timeout	Defines how many seconds to wait before timing a connection out.

Packager Settings

Packager Name	Defines the packager that will be used when creating the channel. This field works the same as its equivalent in the ISO 8583 Message Format Settings.
Packager Path	Defines the path to a generic packager description XML file. This field works the same as its equivalent in the ISO 8583 Message Format Settings.

Header Settings

Request Header	Defines the custom header to send with requests. This field also doubles as the response header size template—enabling the header in the response to be read properly.

Packagers

Packagers define how ISO 8583 messages are structured, including the number of fields in the message and the field data types. They allow the binary data of an ISO 8583 message to be consumed into a generic ISO message that can be easily manipulated and formatted by different packagers. A number of packers are provided by default, and customer packagers can be used to support special cases.

They also provide extensibility to the jPOS API in that custom packagers can be defined to describe custom ISO 8583 messages making it possible to support virtually any type of ISO 8583 message.

Default Packagers

Packager Name	Packager Description

Base1Packager	VISA Base1 binary packager.
Base1SubFieldPackager	VISA Base1 binary subfield packager.
BASE24Packager	BASE24 ASCII packager.
CTCSubElementPackager	Validating packager for subelements in field 48.
CTCSubFieldPackager	Validating packager for subfields in field 48.
DummyPackager	Dummy packager. Throws exceptions if the message is packed/unpacked.
EuroPackager	EuroPay packager.
EuroSubFieldPackager	EuroPay subfield packager.
FSDPackager	FSD ISO message packager.
GenericPackager	Uses an XML description to describe the ISO message.
GenericSubFieldPackager	Uses an XML description to describe the ISO subfields.
GenericTaggedFieldsPackager	Packager for fields containing TLV subfields without a bitmap.
GenericValidatingPackager	Uses an XML description to validate the ISO message.
ISO87APackager	ISO 8583 v1987 ASCII packager.
ISO87APackagerBBitmap	ISO 8583 v1987 ASCII packager using binary bitmap.
ISO87BPackager	ISO 8583 v1987 BINARY packager.
ISO93APackager	ISO 8583 v1993 ASCII packager.
ISO93BPackager	ISO 8583 v1993 BINARY packager.
LogPackager	Packs and unpacks ISO messages from jPOS logs.
MasterCardEBCDICSubFieldPackager	MasterCard EBCDIC subfield packager.
PackagerWrapper	Wraps another ISO packager.
PostPackager	ISO 8583 v1987 packager for Postilion
VAPSMSPackager	ISO 8583 v1987 BINARY packager for VISA’s VAP Single Message (Deprecated).
VAPVIPPackager	ISO 8583 v1987 BINARY packager for VISA’s VAP Single Message (Deprecated).
X92GenericPackager	Uses an XML description to describe ANSI X9.2 ISO messages.
X92Packager	ANSI X9.2 packager.
XML2003Packager	Packs and unpacks ISO 8583 v2003 messages into XML representation.
XMLPackager	Packs and unpacks ISO 8583 messages into XML representation.

Custom Packagers

Custom packagers can be implemented in two ways to provide support for custom ISO 8583 messages.

- Generic Packager XML Description
- ISOPackager Interface

Generic Packager XML Description

In this implementation, the jPOS library provides a generic packager that can handle most ISO 8583 messages. The generic packager uses an XML description of the ISO 8583 message to properly pack and unpack custom ISO 8583 messages.

To define a generic packager description, start by declaring the XML doctype:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE isopackager PUBLIC
        "-//jPOS/jPOS Generic Packager DTD 1.0//EN"
        "http://jpos.org/dtd/generic-packager-1.0.dtd">
```

The ISO packger definition follows the doctype:

```
<isopackager>
        <!-- ISO Field Definitions -->
</isopackager>
```

The ISO field definitions are placed inside the ISO packager definition. ISO field definitions can be primitive ISO 8583 types or complex types.

Primitive ISO Field Definitions

Use the following format to define primitive ISO field definitions:

```
<isofield
        id="0"
        length="4"
        name="MESSAGE TYPE INDICATOR"
        pad="true"
        class="org.jpos.iso.IFB_NUMERIC"/>
```

The **id** attribute defines the field number, the **length** attribute defines the length of data, the **name** attribute defines the field name, the **pad** attribute (optional) specifies whether the field should be padded with characters, and the **class** attribute defines the field packager which correlates to the data type of the field. Custom field packagers can also be created if necessary by implementing an **ISOFieldPackager** and including the implementation on the SOAtest/Virtualize classpath.

The following ISO field packagers are provided by default (all class names should be prefixed with "org.jpos.iso" in the XML definition).

Class Name	Class Description	Class Name	Class Description
IF_CHAR	Fixed length alphanumeric (ASCII)	**IFB_LLCHAR**	Variable length alphanumeric (BINARY, maxlength=99)
IF_ECHAR	Fixed length alphanumeric (EBCDIC)	**IFB_LLHBINARY**	Variable length binary hex (BINARY, maxlength=99)
IF_NOP	Fixed length empty (dummy) field	**IFB_LLHCHAR**	Variable length alphanumeric hex (BINARY, maxlength=99)
IF_TBASE	Fixed length token field	**IFB_LLHECHAR**	Variable length alphanumeric hex (EBCDIC, maxlength=99)
IF_TCHAR	Fixed length alphanumeric with token	**IFB_LLHFBINARY**	Fixed length binary hex (BINARY, length=99)
IF_UNUSED	Fixed length empty (dummy) field, throws an exception if the field is being used.	**IFB_LLHNUM**	Variable length numeric hex (BINARY, maxlength=99)
IFA_AMOUNT	Fixed length amount padded with zeros (ASCII)	**IFB_LLLBINARY**	Variable length binary (BINARY, maxlength=999)
IFA_AMOUNT2003	Fixed length amount padded with zeros (ASCII) for ISO 8583-2003	**IFB_LLLCHAR**	Variable length alphanumeric (BINARY, maxlength=999)
IFA_BINARY	Fixed length binary (ASCII)	**IFB_LLLHBINARY**	Variable length binary hex (BINARY, maxlength=999)
IFA_BITMAP	Fixed length bitmap (ASCII)	**IFB_LLLHCHAR**	Variable length alphanumeric hex (BINARY, maxlength=999)
IFA_FLLCHAR	Fixed length alphanumeric (ASCII, length=99)	**IFB_LLLHECHAR**	Variable length alphanumeric hex (EBCDIC, maxlength=999)
IFA_FLLNUM	Fixed length numeric (ASCII, length=99)	**IFB_LLLHNUM**	Variable length numeric hex (BINARY, maxlength=999)
IFA_LBINARY	Variable length binary (ASCII, maxlength=9)	**IFB_LLLLBINARY**	Variable length binary (BINARY, maxlength=9999)
IFA_LCHAR	Variable length alphanumeric (ASCII, maxlength=9)	**IFB_LLLNUM**	Variable length numeric (BINARY, maxlength=999)

IFA_LLABINARY	Variable length binary (ASCII, maxlength=99)	**IFB_LLNUM**	Variable length numeric (BINARY, maxlength=99)
IFA_LLBINARY	Variable length binary (ASCII, maxlength=99)	**IFB_NUMERIC**	Fixed length numeric (BINARY)
IFA_LLBNUM	Variable length numeric (ASCII BCD, maxlength=99)	**IFE_AMOUNT**	Fixed length amount (EBCDIC)
IFA_LLCHAR	Variable length alphanumeric (ASCII, maxlength=99)	**IFE_BINARY**	Fixed length binary (EBCDIC)
IFA_LLLABINARY	Variable length binary (ASCII, maxlength=999)	**IFE_BITMAP**	Fixed length bitmap (EBCDIC)
IFA_LLLCHAR	Variable length alphanumeric (ASCII, maxlength=999)	**IFE_CHAR**	Fixed length alphanumeric (EBCDIC)
IFA_LLLLBINARY	Variable length binary (ASCII, maxlength=9999)	**IFE_LLBINARY**	Variable length binary (EBCDIC, maxlength=99)
IFA_LLLLCHAR	Variable length alphanumeric (ASCII, maxlength=9999)	**IFE_LLCHAR**	Variable length alphanumeric (EBCDIC, maxlength=99)
IFA_LLLLLBINARY	Variable length binary (ASCII, maxlength=99999)	**IFE_LLLBINARY**	Variable length binary (EBCDIC, maxlength=999)
IFA_LLLLLCHAR	Variable length alphanumeric (ASCII, maxlength=99999)	**IFE_LLLCHAR**	Variable length alphanumeric (EBCDIC, maxlength=999)
IFA_LLLNUM	Variable length numeric (ASCII, maxlength=999)	**IFE_LLLEBINARY**	Variable length binary EBCDIC (EBCDIC, maxlength=999)
IFA_NUMERIC	Fixed length numeric (ASCII)	**IFE_LLLLBINARY**	Variable length binary (EBCDIC, maxlength=9999)
IFA_TTLBINARY	Variable length binary with token (ASCII, maxlength=9)	**IFE_LLLLCHAR**	Variable length alphanumeric (EBCDIC, maxlength=9999)
IFA_TTLCHAR	Variable length alphanumeric with token (ASCII, maxlength=9)	**IFE_LLNUM**	Variable length numeric (EBCDIC, maxlength=99)
IFA_TTLLBINARY	Variable length binary with token (ASCII, maxlength=99)	**IFE_NUMERIC**	Fixed length numeric (EBCDIC)
IFA_TTLLCHAR	Variable length alphanumeric with token (ASCII, maxlength=99)	**IFE_SIGNED_ NUMERIC**	Fixed length signed numeric (EBCDIC)
IFA_TTLLLBINARY	Variable length binary with token (ASCII, maxlength=999)	**IFEA_LLCHAR**	Esoteric variable length alphanumeric (ASCII EBCDIC, maxlength=99)
IFA_TTLLLCHAR	Variable length alphanumeric with token (ASCII, maxlength=999)	**IFEB_LLLNUM**	Esoteric variable length numeric (EBCDIC, maxlength=999)
IFA_TTLLLLBINARY	Variable length binary with token (ASCII, maxlength=9999)	**IFEB_LLNUM**	Esoteric variable length numeric (EBCDIC, maxlength=99)
IFA_TTLLLLCHAR	Variable length alphanumeric with token (ASCII, maxlength=9999)	**IFEMC_LLCHAR**	Esoteric variable length alphanumeric (EBCDIC, maxlength=99)
IFB_AMOUNT	Fixed length amount (BINARY)	**IFEP_LLCHAR**	Europay variable length alphanumeric (EBCIDC, maxlength=99)
IFB_AMOUNT2003	Fixed length amount for ISO 8583-2003 (BINARY)	**IFEPE_LLCHAR**	Mastercard variable length alphanumeric (EBCDIC, maxlength=99)
IFB_BINARY	Fixed length binary (BINARY)	**IFIPM_LLLCHAR**	Variable length alphanumeric with token (maxlength=999)
IFB_BITMAP	Fixed length bitmap (BINARY)	**IFMC_LLBINARY**	Variable length binary with token (maxlength=99)
IFB_FLLLNUM	Fixed length numeric (BINARY, length=999)	**IFMC_LLCHAR**	Variable length alphanumeric with token (maxlength=99)
IFB_FLLNUM	Fixed length numeric (BINARY, length=99)	**IFMC_LLLBINARY**	Variable length binary with token (maxlength=999)
IFB_FNUMERIC	Fixed length numeric (BINARY)	**IFMC_LLLCHAR**	Variable length alphanumeric with token (maxlength=999)
IFB_LLBINARY	Variable length binary (BINARY, maxlength=99)		

Complex ISO Field Definitions

Use the following format to define complex ISO field definitions:

```
<isofieldpackager
        id="127"
        length="255"
        name="FILE RECORS(S) ACTION/DATA"
        class="org.jpos.iso.IFB_LLHBINARY"
        packager="org.jpos.iso.packager.GenericSubFieldPackager"/>


<!-- ISO Field Definitions -->
</isofieldpackager>
```

The **id, length, name,** and **class** attributes have the same meaning as they do for primitive ISO field definitions. An additional **packager** attribute must be defined to describe how this complex ISO field should be packaged with the rest of the ISO message, followed by the sub-field descriptions, which are defined like primitive ISO fields.

An example custom packager for VISA's Base1 ISO 8583 messages provided with the jPOS examples is included for reference; see base1.xml, which is included in the plugin's zip file.

ISOPackager Interface

In this implementation, the jPOS library also provides a Java interface that can be used to define custom packagers. If the generic packager, or one of the default packagers, cannot support a certain ISO 8583 message, then a custom ISO packager can be implemented.

Channels

Channels define how the ISO 8583 client and server communicate with each other. They ensure that the generic ISO messages are properly formatted before they are sent over the wire and are properly reconstructed at the other end. The channel handles the connections and the protocols used to transfer the ISO 8583 messages between client and server. As with the packagers, the jPOS library provides many channels that should handle most cases. However, if the proper channel does not exist, custom ISO 8583 channels can be implemented by extending the **BaseChannel** class.

The following channels are provided by default:

Channel Name	Channel Description
AmexChannel	American Express channel.
ASCIIChannel	ISO base channel extension with four ASCII character message length header.
BCDChannel	ISO base channel extension with the following message format *[LEN][TPDU][ISOMSG]* Where LEN is 2 hex bytes.
CSChannel	CS Standard Channel.
FSDChannel	ISO base channel extension with the following message format *[LEN][TPDU][ISOMSG]* Where LEN is 2 bytes in network byte order.
GZIPChannel	ISO base channel extension that GZIP compresses data that is sent over the channel.
HEXChannel	ISO base channel extension with a four ASCII hex character message length header.
LogChannel	ISO base channel extension that extracts ISOMSG blocks from the ISO logger.
NACChannel	ISO base channel extension with the following message format *[LEN][TPDU][ISOMSG]* Where LEN is 2 bytes in network byte order.
NCCChannel	ISO base channel extension with the following message format *[LEN][TPDU][ISOMSG]* Where LEN is 2 bytes in binary-coded decimal (BCD) format.
PADChannel	ISO Channel suitable to be used to connect to an X.25 PAD.
PostChannel	ISO base channel extension with the following message format *[LEN][ISOMSG]* Where LEN is 2 bytes in network byte order (NBO).

RawChannel	ISO base channel extension with the following message format *[LEN][ISOMSG]* Where LEN is 4 bytes in network byte order (NBO).
RBPChannel	Record Boundary Preservation channel.
TelnetXMLChannel	Exchanges XML based ISO-8583 messages through a telnet session the telnet commands are ignored.
VAPChannel	VISA’s VAP framing (deprecated).
X25Channel	ISO Channel suitable to be used to connect to an X.25 PAD.
XMLChannel	ISO base channel extension that exchanges XML based ISO-8583 messages.

Third-party Content

This tool set includes items that have been sourced from third parties as outlined below.

- jPOS (GNU AGPL v3.0)
- JDOM library (JDOM license)

Parasoft's ISO extensions are being released under the GNU AGPL v3.0 license.

Additional license details are available in this plugin's **licenses** folder.