# Integration New Media's Moka Xtra

**For Macromedia® Director®**

Version 1.1

User Manual

integration new media

 | Manual Version 1.2 | 2003-03-26

# Contents


**Contents** 2
**License Agreement** 4
**Welcome to Moka Xtra** 6
System Requirements 6
Installation 6
Accessing custom Java™ class libraries 7
How to register your Moka Xtra license 7
Technical support 7
About INM 8
Contacting us 8
**Java™ types and signatures** 9
The JClassBrowser tool 9
Data types 10
Object types 10
java.lang.String type 10
Signature types 10
**Using Moka Xtra** 12
Checking for a valid JRE™ 12
Creating an instance of Moka Xtra 12
Checking if New was successful 13
Creating an instance of a Java™ class 13
Calling Java™ methods 13
Simple Java™ method call 13
Passing arrays to Java™ classes 14
Starting a Java™ class main() function 14
Passing an absolute path to a Java™ object 14
Destroying the instance of a class 15
Closing the Xtra instance 15
Debugging your code 15
Problems on Mac OS X 15
Basic documentation 16
**Delivering to the end user** 17
Standalone projectors 17
Mac projectors 17
Distributing the Java™ Runtime Engine 17
Delivering Shockwave projectors and movies 18
**Where to learn more** 19
FAQ's, Demos and Tech Notes 19
MokaXtra-L discussion list 19

Java™ web sites 19
**Methods Reference 20**
New 20
moka_CreateJavaObject 21
moka_DestroyJavaObject 22
moka_CallJavaMethod 22
moka_CallStaticJavaMethod 23
moka_GetClassInfo 23
moka_GetCurrentJVMInstalled 24
moka_GetJavaMember 25
moka_GetLastJavaException 25
moka_GetStaticJavaMember 26
moka_Path 26
moka_Register 27
moka_SetJavaMember 27
moka_SetStaticJavaMember 28
moka_TranlsateToSignature 28
**Error Codes 29**
**Index 30**

# License Agreement

PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE USING Integration New Media's MOKA XTRA. BY USING Integration New Media's MOKA XTRA, YOU AGREE TO BECOME BOUND BY THE TERMS OF THIS LICENSE AGREEMENT.

The enclosed computer program(s), license file and data (collectively, "Software") are licensed, not sold, to you by Integration New Media, Inc. ("INM") for the purpose of using it for the development of your own products ("Products") only under the terms of this Agreement. INM and its licensors reserve any rights not expressly granted to you. INM and its licensors grant you no right, title or interest in or to the Software. The Software is owned by INM and its licensors and is protected by International copyright laws and international treaties.

1. **License**.

(a) You may install one copy of the Software on a single Windows-compatible computer and one copy on a single Macintosh-compatible computer. To "install" the Software means that the Software is either loaded or installed on the permanent memory of a computer (i.e., hard disk). This installed copy of the Software may be accessible by multiple computers, however, the Software cannot be installed on more than one computer at any time. You may only install the Software on another computer if you first remove the Software from the computer on which it was previously installed. You may not sublease, rent, loan or lease the Software.

(b) You may make one copy of the Software in machine-readable form solely for backup purposes. As an express condition of this Agreement, you must reproduce on each copy any copyright notice or other proprietary notice that is on the original copy supplied by INM.

(c) Your license is limited to the particular version (collectively "Version") of the Software you have purchased. Therefore, use of a Version other than the one encompassed by this License Agreement requires a separate license.

(d) The Software contains a license file (.LIC), which is subject to the restrictions set forth above and may not be distributed by you in any way. However, INM and its licensors grant you a royalty-free right to reproduce and distribute the file named "Moka Xtra.X32" on Windows, “Moka Xtra.XTR” on Mac (collectively, "Runtime Kit") provided that (i) you distribute the Runtime Kit only in conjunction with and as part of your own Products; (ii) own a license for the specific Version of the Software that contains the Runtime Kit; (iii) agree to indemnify, hold harmless and defend INM and its licensors from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your Products with the Runtime Kit.

(e) Any third party who may distribute or otherwise make available a product containing the Runtime Kit must purchase its own license of the Software.

(f) Any third party who will use the Runtime Kit in an authoring environment must purchase its own license of the Software.

(g) Notwithstanding any other terms in this License, if the Software is licensed as an upgrade or update, then you may only use the Software to replace previously validly licensed versions of the same software. You agree that the upgrade or update does not constitute the granting of a second license to the Software (i.e., you may not use the upgrade or update in addition to the software it is replacing, nor may you transfer the software which is being replaced to a third party).

2. **Restrictions**.

(a) The Software contains trade secrets in its human perceivable form and, to protect them, you may not MODIFY, TRANSLATE, REVERSE ENGINEER, REVERSE ASSEMBLE, DECOMPILE, DISASSEMBLE OR OTHERWISE REDUCE THE SOFTWARE TO ANY HUMAN PERCEIVABLE FORM. YOU MAY NOT MODIFY, ADAPT, TRANSLATE, RENT, LEASE, LOAN OR CREATE DERIVATIVE WORKS BASED UPON THE SOFTWARE OR ANY PART THEREOF.

(b) THE SOFTWARE IS NOT INTENDED FOR USE IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, AIR TRAFFIC CONTROL, OR OTHER ENVIRONMENTS IN WHICH THE FAILURE OF THE SOFTWARE COULD LEAD TO DEATH, PERSONAL INJURY, OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE.

(c) You may not transfer or assign your rights under this License to another party without INM's prior written consent. Assignment application forms can be obtained from INM's sales department.

3. **Copyright notices**. You may not alter or change INM's and its licensors' copyright notices as contained in the Software. You must include a copyright notice, in direct proximity to your own copyright notice, in substantially the following form "Portions of code are Copyright ©2001-2003 used under license by Integration New Media, Inc.".

4. **Acceptance**. The Software shall be deemed accepted by you upon delivery unless you provide INM, within two (2) weeks therein, with a written description of any bona fide defects in material or workmanship.

5. **Termination**. This Agreement is effective until terminated. This Agreement will terminate immediately without notice from INM or judicial resolution if you fail to comply with any provision of this Agreement. Upon such

termination you must destroy the Software, all accompanying written materials and all copies thereof, and Sections 6 and 7 will survive any termination.

6. **Limited Warranty**. INM warrants for a period of ninety (90) days from your date of purchase (as evidenced by a copy of your receipt) that the media on which the Software is recorded will be free from defects in materials and workmanship under normal use and the Software will perform substantially in accordance with the manual. INM's entire liability and your sole and exclusive remedy for any breach of the foregoing limited warranty will be, at INM's option, replacement of the disk, refund of the purchase price or repair or replacement of the Software.

7. **Limitation of Remedies and Damages**. In no event will INM, its parent or subsidiaries or any of the licensers, directors, officers, employees or affiliates of any of the foregoing be liable to you for any consequential, incidental, indirect or special damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of business information and the like), whether foreseeable or not, arising out of the use of or inability to use the Software or accompanying written materials, regardless of the basis of the claim and even if INM or an INM representative has been advised of the possibility of such damage. INM's liability to you for direct damages for any cause whatsoever, and regardless of the form of the action, will be limited to the greater of US $199.00 or the money paid for the Software that caused the damages.

THIS LIMITATION WILL NOT APPLY IN CASE OF PERSONAL INJURY ONLY WHERE AND TO THE EXTENT THAT APPLICABLE LAW REQUIRES SUCH LIABILITY. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. **General**. This Agreement will be construed under the laws of the Province of Quebec, except for that body of law dealing with conflicts of law. If any provision of this Agreement shall be held by a court of competent jurisdiction to be contrary to law, that provision will be enforced to the maximum extent permissible, and the remaining provisions of this Agreement will remain in full force and effect.

9. The parties acknowledge having requested and being satisfied that this Agreement and its accessories be drawn in English. Les parties reconnaissent avoir demandé que cette entente et ses documents connexes soient rédigés en anglais et s'en déclarent satisfaits.

# Welcome to Moka Xtra

Integration New Media's Moka Xtra blends the functionality of Java™ with Director's rich-media interface. Through the Lingo API, Director applications can call Java™ classes, and get and set the values of Java™ class members. With this Xtra, multi-media developers will now be able to accomplish complex tasks in Director that could not be achieved through Lingo alone. Java™ programmers will love the ease of creating engaging user interfaces in Director.

To keep up with the latest Moka Xtra developments, bookmark the URL and visit the product site regularly:

http://www.MokaXtra.com

## System Requirements

- Director 7 and higher (including Director MX on both Mac OS X and Windows)

### Macintosh

- All Macintosh computers equipped with Mac OS 8.1 and higher, including Mac OS X.
- Mac OS Runtime for Java™ (MRJ) version 2.1 or later installed on the end-user's system. The MRJ is included with the Mac OS. It is recommended to use MRJ 2.2 because of bugs in version 2.1. For versions of Mac OS prior to 8.6, you will probably have to upgrade the MRJ, because MRJ 2.1 began shipping with Mac OS 8.6.

### Windows

- All Windows computers equipped with Windows 9x/NT/2000/XP
- Sun JRE™ (Java™ Runtime Environment) 1.3.x or 1.4.x installed on the end-user's system, or distributed with the projector (see Delivering to the end user)

To download Sun's Java Platform (Java™ VM and Java™ API) go to:

http://java.sun.com/getjava/index.html

## Installation

To install Moka Xtra for authoring (i.e., in the Director application):

1 Make sure Director is not running,

2. Place the Xtra (On the PC: a file named "**MokaXtra.x32**", on Mac: a file named "**MokaXtra.XTR**") into Director's Xtras folder. This folder is located in the same place as the Director application (a typical folder path for Windows is: C:\Program Files\Macromedia\Director 8.5\Xtras).
3. Create a subfolder within Director's Xtras folder, and name it "Moka Xtra".
4. Move the JClassBrowser tool, the file named "**JClassBrowser.dcr**", into this new Moka Xtra folder. This will ensure that the tool is available by clicking **Xtras > Moka Xtra > JClassBrowser** from Director's menu.
5. Launch Director.

### Accessing custom Java™ class libraries

Java™ class libraries that you create may be located in any folder accessible to Director. You specify the path(s) for Moka Xtra to locate them when you instantiate Moka Xtra, using the `New` method. The classes you create may be uncompressed, as .class files, or compressed in .jar or .zip files. See New in the Methods Reference section.

## How to register your Moka Xtra license

**Note:** The free evaluation version of Moka Xtra contains all the functionality of the licensed version, except that a Demo Version alert appears when Moka Xtra is instantiated. Click the alert box to close it.

In order to ship a product that uses Moka Xtra you must purchase a license for the software. Please make sure to read and agree to the License Agreement (see License Agreement) before purchasing your Moka Xtra license. To purchase a license, visit our online store at: http://www.IntegrationNewMedia.com/store/,

or contact us by phone at:
+1 800 400 1772 or +1 514 871 1333, Option 5.

**Note:** Never distribute the license number. If your movie must be sent to other developers, first remove the license number from your script.

To register your Xtra, and remove the "Unregistered" alert screen, you need to call the Moka Xtra method `moka_Register(`*`thekey`*`)` at the beginning of each Director movie that uses the Xtra. The string parameter *`theKey`* is the license number that was given to you by INM upon purchase of the license. It must be entered in quotes. To avoid making mistakes we recommend you copy/paste this number from the email that Integration New Media sends you.

Example: `moka_Register("J1R9-T6K8-2HC2-J391")`

## Technical support

The MokaXtra-L discussion list is the best place to get support and benefit from the experience of the community of Moka Xtra users.

If you need further assistance or personal attention, our Technical Support team is pleased to help you. Contact them by E-mail

(preferred) or by phone. See Contacting us below. Licensed users are given priority status when contacting Technical Support.

## About INM

Integration New Media (INM) specializes in the development of cross-platform software components for multimedia applications and on-line communication networks. INM offers a wide range of products, solutions and services related to multimedia applications.

Our Home Page is located at:
http://www.IntegrationNewMedia.com/

We develop high-quality products, such as V12 Database Engine, GoldenGate Database Connector, SecureNet Xtra and PDF Xtra, and also offer customized services at various levels, including:

- **Programming Services:** We offer programming expertise in Director, Authorware, C++, Java, Flash programming, and Visual Basic. Our expert programmers can provide you with the programming support you need, while remaining completely behind the scenes.
- **Turnkey Solutions:** We have the experience and expertise to manage and develop entire new media projects from conception to completion.
- **Custom Xtras**: With our years of experience and proven track record, we are sure to develop the right custom Xtra for your Director or Authorware project.
- **Emergency Assistance:** We can provide emergency assistance for questions or issues that are specific to your project.

For further information about our services, please contact our Sales team (see below).

## Contacting us

We encourage you to contact us with questions, comments or feedback. We can be reached at:


Integration New Media, Inc.
1425 Rene-Levesque West, Suite 906
Montreal, Quebec, Canada H3G 1T7

| | |
|---|---|
| Phone: | +1 514 871 1333, Option 5 |
| | +1 800 400 1772 (North America) |
| Info: | info@IntegrationNewMedia.com |
| Sales: | sales@IntegrationNewMedia.com |
| Support: | support@IntegrationNewMedia.com |
| Web: | http://www.IntegrationNewMedia.com/ |

# Java™ types and signatures

Understanding the concept of Java™ types and signatures is fundamental to using Moka Xtra. Most of the Moka Xtra methods require the signature of the Java™ class, method or member you are accessing, as one of the parameters. Because Java™ methods can be "overloaded" (there are more than one way to use them), the signature uniquely identifies which implementation of a method you are using.

To help you obtain the correct signatures and reduce the risk of errors in typing them, we have developed the **JClassBrowser** tool.

## The JClassBrowser tool

The JClassBrowser tool retrieves the names of all methods and members for Java™ classes accessible on your local machine, and gives you their signatures. Because Java™ signatures are not easy to remember and they are case sensitive, this tool is very helpful to use while coding Moka Xtra method calls in Lingo.

The JClassBrowser tool (JClassBrowser.dcr) is a Shockwave movie that comes with the Moka Xtra download package. Before starting Director, you should place this file in a folder named Moka Xtra, within Director's Xtras folder. The JClassBrowser tool is then accessible from Director, as a MIAW (movie in a window), by clicking **Xtras > Moka Xtra > JClassBrowser**, from the menu bar.

**IMPORTANT NOTE:**
The path format differs depending on the platform. Follow the examples at the top of the Paths window.

Follow these steps to retrieve the signature of a Java™ method or member:

1. Click the **Paths...** button to add the paths where your own Java™ classes reside. You don't have to enter the path for the currently installed JVM. Once your paths are defined, click **OK**.
2. In the **Class Name** field, type in the name of a class. Remember it is case sensitive! Click **Get Info** to see a listing of all the methods and members of that class.
3. Click the **Super Class** button if you want information about the parent class.
4. Use the back and forward arrow buttons to go back to previous classes.
5. Click a method or member name and the JClassBrowser displays the possible **declarations** in a field below.
6. If there are multiple declarations, click the one you want to execute and its **signature** is displayed.

Click **Copy** to copy the signature to the clipboard. Then paste it directly into your Lingo code.

## Data types

This table shows the Java™ equivalents of Lingo types.

| Lingo | Java™ |
|---|---|
| Integer | int, byte, boolean, short, long |
| Float | float, double |
| Date | long (contains the number of milliseconds that have passed since January 1st, 1970). |
| String | java.lang.String<br>Object (string in the format '<Java Class "ClassName" 0x……..>') |
| List (all list elements must be of the same type) | array |

## Object types

When Java™ returns a reference to an object (except java.lang.String), Lingo receives this reference in the form of a string, the content of which resembles an xtra instance value (e.g.: <Java Class "java.util.Date" 0x45AE5400>). If the reference is equal to <null>, Lingo receives the string "NULL".

### java.lang.String type

The java.lang.String type is automatically converted by the xtra to a string upon input and output.

## Signature types

Most of Moka Xtra's methods require the signature of the Java™ method to be executed. This table shows how each variable type is represented in the signature:

| Signature | Type |
|---|---|
| Z | boolean |
| B | byte |
| C | char |
| S | short |
| I | int |
| J | long |
| F | float |
| D | double |

| | |
|---|---|
| V | void |
| L fully-qualified-class ; | fully-qualified-class |
| [type | type[] |
| (arg-types) ret-type | method type |

**Examples :**

| Method | Signature |
|---|---|
| Public Fct() | ()V |
| Public int Fct(String) | (Ljava/lang/String;)I |
| Public int[] Fct(int, String) | (ILjava/lang/String;)[I |

To take advantage of the JVM classes or your own custom-developed Java™ classes, you must first create an instance of the Moka Xtra, and an instance of each Java™ class you want to use. When you are finished using a Java™ class, you must destroy its instance. When you no longer need the Moka Xtra instance you must destroy it also.

The basic procedure for using Moka Xtra is as follows:

1. Check for a valid Java™ Runtime Engine (JRE™)
2. Create an instance of the Moka Xtra
3. Create instances of any classes you want to use (except for static methods/members)
4. Call the relevant Moka Xtra methods to call Java™ methods or get/set member variables
5. Destroy the class instances you created previously
6. Set the global Moka Xtra instance to zero

## Checking for a valid JRE™

On Mac OS 8 and higher, the Java™ Runtime Engine (JRE™) ships with the operating system. It is called Mac OS Runtime for Java™ (MRJ *+ theversion number).* Therefore, you don't need to check if there is a Java™ runtime engine on Mac, unless you need a specific version of the MRJ.

On Windows, you have two options:

1. Distribute the JVM alongside your projector, on CD-ROM (see Distributing the Java™ Runtime Engine).
2. Check to make sure the user has a valid Java™ Runtime Engine installed.

The method, `moka_GetCurrentJVMInstalled()` returns a string containing the default installed Java™ Virtual Machine. If there is no JVM installed, it returns an empty string. Your application should call this method before attempting to create an instance of Moka Xtra. This way, if there is no JVM present, you can alert users to download the version of JVM they need or even send them to the URL where they can download the JVM, before running your application (see moka_GetCurrentJVMInstalled in the Methods Reference section of this manual).

## Creating an instance of Moka Xtra

Call `New` to create an instance of Moka Xtra. Generally, you store a newly created Xtra instance in a global variable for future use. The following example uses the `New` method of Moka Xtra.

Example:

```
gMoka = new(xtra "MokaXtra", the moviepath)
```

The first parameter is Moka Xtra. The second parameter is a string that contains the **absolute** path (or paths) to your custom classes. Separate multiple paths using a semi-colon.

You must specify the full, absolute path. However, you can use Director's properties, `the moviePath` and `the applicationPath`, so you only need to append the relative path.

Example:

```
myPath = the applicationpath & "myClass1;" & the
   applicationpath & "myClass2;" & the applicationpath &
   "myClass3"
gMoka = new(xtra "MokaXtra", myPath)
```

See `New` in the Methods Reference section of this document for more examples of how to specify the class path.

### Checking if New was successful

You should always ensure that the Xtra was created successfully immediately after calling `New`. `New` can fail for many reasons, such as a lack of free memory or as a result of misplaced files.

Example:

```
if NOT ObjectP(gMoka) then alert "Moka Xtra Init Failed"
```

## Creating an instance of a Java™ class

**Note:** If you are using static Java methods and members, you do not need to create an instance of those classes before referencing them. In this case, just use the object name directly, rather than the instance variable. See moka_CallStaticJavaMethod, moka_GetStaticJavaMember, moka_,

Once the Moka Xtra is instantiated, use the method `moka_CreateJavaObject()` to create an instance of the class you need to use. The first parameter is the full class name. The second is the signature of the class (see Java™ types and signatures). Be sure to verify if the class was created successfully before calling any Moka Xtra methods that use it.

Example:

```
-- Create an instance of the java.util.Date class
gJavaDate =
   gMoka.moka_CreateJavaObject("java/util/Date","()V")
-- check if the class instance was created
if (stringP(gJavaDate)) then
   --
   -- call Java methods of the class
   --
else
   alert("Error: Java class not created.")
end if
```

## Calling Java™ methods

Moka Xtra methods allow you to call Java™ class methods as well as get and set Java™ class members.

### Simple Java™ method call

This example converts a string entered by the user, to upper case. Within the java/lang/String class, there is a function `toUpperCase`,

declared as:

```
public toUpperCase()
```

Its corresponding signature is:

```
"()Ljava/lang/String;"
```

In Lingo, the code would look like this:

```
gMoka = new(xtra "MokaXtra", the moviepath)
JString = gMoka.moka_CreateJavaObject("java/lang/String",
   "(Ljava/lang/String;)V",member("UserID").text)
if (stringP(JString)) then
  -- call toUpperCase method to convert string
  capUserID = gMoka.moka_CallJavaMethod(JString,
   "toUpperCase","()Ljava/lang/String;")
-- destroy the instance of the class
gMoka.moka_DestroyJavaObject(JString)
else
  alert("Error: Java class not created.")
end if
```

### Passing arrays to Java™ classes

To pass a parameter that is an array in Java™, use a Lingo list.

Example:

The Java™ function you are calling is declared as:

```
public String DoSomething(String[] passedString)
```

The corresponding signature is:

```
"([Ljava/lang/String;)Ljava/lang/String;"
```

In Lingo, the code looks like this:

```
gMoka = new(xtra "MokaXtra", the moviepath)
gMyClass = gMoka.moka_CreateJavaObject("MyClass","()V")
if (stringP(gMyClass)) then
   -- call DoSomething method, passing 3 string values in
   a list
   errCode =
   gMoka.moka_CallJavaMethod(gMyClass,"DoSomething","([Lj
   ava/lang/String;)Ljava/lang/String;",["aaa","bbb","ccc
   "])
else
      alert("Error: Java class not created.")
end if
```

### Starting a Java™ class main() function

To start a Java application that contains the `main()` function, use the Moka Xtra method `CallJavaStaticMethod()`.

Example:

```
(gMoka.CallJavaStaticMethod("myClass", "main",
   "([Ljava/lang/String;)V", [<arguments>]))
```

## Passing an absolute path to a Java™ object

File paths are formatted differently depending on the operating system, and there are special considerations when coding cross-platform Java™ applications that require absolute file paths. For this reason, Moka Xtra has a built in function, `moka_Path()`, which automatically tests the end-user's operating system and converts a file path to the format needed by Java™, if necessary.

**IMPORTANT NOTE:**
`moka_Path()` ignores all strings that contain a Unix-style path or any forward-slash characters. Therefore, applying `moka_Path()` more than once on the same string will have no effect.

You should use `moka_Path()` as you would use a type conversion function, whenever you call a Java™ object that requires a file path.

Example:

```
pFile = gMoka.moka_CreateJavaObject("java.io.File",
"(Ljava/lang/String;)V", moka_Path(theFileName))
```

## Destroying the instance of a class

When you are finished using a Java™ class instance, use the `moka_DestroyJavaObject` method to destroy it and free up the memory it occupied.

```
-- Destroy the instance of the java.util.Date class
gMoka.moka_DestroyJavaObject(JavaDate)
```

## Closing the Xtra instance

When the Xtra instance has completed its function and is no longer required, close it by calling setting the global variable to 0. Closing an Xtra performs mandatory housekeeping tasks and closes unneeded files. It also frees the memory occupied by the Xtra. All Xtra instances created with `New` must be ultimately set to *0* once they are no longer needed.

```
Example:
set gMoka = 0
```

## Debugging your code

Moka Xtra has a small set of error codes that are returned when errors are encountered accessing Java™ objects (see Error Codes). However, debugging your Java™ code from Director is not trivial. Use the Moka Xtra method, `moka_GetLastJavaException()` to return the last exception thrown by a Java™ function. It returns "none" if everything is OK.

Call `moka_GetLastJavaException()` after each call to a Java™ method and do your own error handling based on the string value returned.

Once you modify your Java™ class, you have to close and restart Director in order to load the new version of the class.

### Problems on Mac OS X

If your project seems to work correctly on Windows and Mac OS 8/9, but you are experiencing problems on Mac OS X, verify any file or folder paths that are used. Java™ expects Unix-formatted absolute paths. On Mac OS X, all volumes, except the system volume, are prefixed by `"/Volumes"`.

We recommend using `moka_Path()` to convert any absolute path passed to a Java™ object. However, beware that `moka_Path()` will ignore any strings containing the forward slash character. Therefore, you cannot append a Mac Classic-style path, with colon

(:) folder delimiters, to an already converted Unix path and pass that string to `moka_Path()`. The ZipSample demo project, available on our website illustrates how to deal with this issue:

http://www.MokaXtra.com/demos

## Basic documentation

The Methods Reference section of this manual describes all the Moka Xtra methods, and provides examples of their usage. In addition, Director has a built-in mechanism that provides quick documentation for Lingo developers as you work. In the Message Window, type:

```
put interface (Xtra "MokaXtra")
```

where Xtra "MokaXtra" is the name of the Xtra library, not of an instance of the Xtra.

The above command returns the following Xtra description:

```
-- "xtra MokaXtra
-- Moka Xtra 1.1 Release 7 (c)1999-2003 Integration New Media,
   Inc.

new object me, string
moka_CreateJavaObject object me, string, string, *
moka_DestroyJavaObject object me, string
moka_CallJavaMethod object me, string, string, string, *
moka_CallStaticJavaMethod object me, string, string, string, *
moka_GetJavaMember object me, string, string, string
moka_SetJavaMember object me, string, string, string, any
moka_GetStaticJavaMember object me, string, string, string
moka_SetStaticJavaMember object me, string, string, string, any
moka_GetClassInfo object me, string
moka_TranslateToSignature object me, string
-- Global handlers --
*moka_Register string
*moka_GetCurrentJVMInstalled
*moka_GetLastJavaException
"
```

Methods that expect a fixed number of parameters are those for which each parameter is listed. Methods that accept a variable number of parameters are those followed by a *****.

## Standalone projectors

For playback as a standalone projector, you must deliver the Moka Xtra file (on Windows: **MokaXtra.X32**, on Mac: **MokaXtra.XTR**) in a folder named "Xtras" located in the same folder as the projector itself. You also need to put your custom Java™ classes somewhere within that folder and specify that path in the Moka Xtra `New` method.

### Mac projectors

Director MX treats Mac OS X and Classic MacOS (Mac OS 8.6 – 9.x) as separate platforms, requiring separate executables. However, through Director MX, there is a way to distribute a single a cross-platform projector that runs on both Mac OS 9.x and Mac OS X. For complete details on this solution, go to the technote named "Considerations for creating a Macintosh projector bundle" on Macromedia's website:
http://www.macromedia.com/support/director/ts/documents/bundle_proj.htm

## Distributing the Java™ Runtime Engine

On Mac OS 8 and higher the Sun Java™ Runtime Engine is included with the operating system, so there is less of a need to distribute it with your application. However, there is no guarantee on Windows, that your end-user has a valid JVM, or what version is installed.

**NOTE:** Since version 1.4 .1, Sun permits the redistribution of the JRE with a value-added application.

To ensure that your Windows end-user has the JVM for your application, you can distribute the Java™ Runtime Engine with your project, on CD.

When Moka Xtra is initialized, using `NEW()`, it looks for a valid JRE™ using the following algorithm:

1. Check if there is a JRE™ right beside the movie (or in Director's folder, in authoring mode) – i.e., if the JRE™ folders "lib" and "bin" are at the same level as the application (either your projector or Director).
2. If step 1 fails, it checks if the JRE™ is located in a folder named "JRE", beside the movie (or in Director's folder, in authoring mode) - i.e., if the JRE™ folders "lib" and "bin" are in a folder named "JRE"
3. If step 2 fails, it checks in the Windows registry to locate the current version used on the system.
4. If step 3 fails, it returns an error code (-5001).

For more information on distributing the Sun JRE™ go to
http://servlet.java.sun.com/help/legal_and_licensing/#63

## Delivering Shockwave projectors and movies

Because **Moka Xtra** allows you to call any Java™ class from Director (both the standard JVM classes and your own customized classes), it cannot be Shockwave safe. That is, we cannot guarantee that the Java™ classes it accesses will not be harmful. Therefore, if you need to deliver your project as a Shockwave movie, you must create your own Verisign approved auto-download package.

The support team at Integration New Media can assist you in creating the Shockwave auto-download package. In order to do this you would first need to obtain your own Verisign certificate. For more information, contact our support team at: Support@IntegrationNewMedia.com

# Where to learn more

## FAQ's, Demos and Tech Notes

The Moka Xtra FAQs page, on INM's website is where you can find answers to questions you might have about the product's capabilities and requirements. Visit the FAQ page at: http://www.MokaXtra.com/FAQ/

For answers to technical questions, visit the Tech Notes section. Tech notes address specific technical issues that have been identified by INM's development team and by support cases that are reported. Technotes often contain sample code and sample movies that can help increase your productivity with the Xtra.

http://www.IntegrationNewMedia.com/support/MokaXtra/TechNotes/

To help you get started quickly with Moka Xtra, we have posted open-source demo projects that use Moka Xtra and custom Java™ classes. Download these demos from: http://www.MokaXtra.com/demos

## MokaXtra-L discussion list

MokaXtra-L is a mailing list devoted to discussions on Moka Xtra and related topics. New announcements concerning Moka Xtra are also posted in this forum. Subscribe to this list by clicking: http://www.IntegrationNewMedia.com/support/list/.

## Java™ web sites

On the web you can find many sites that provide Java™ classes, tutorials and projects. Here are just a few that may be useful:

- Java.Sun.com home page
- Sun's Java API doc
- Apache Jakarta Project
- Jakarta: Excalibur
- GameLan Repository
- Cloudscape Java database
- PointBase Java database
- Apple's Java developer doc

The example code shown in this section uses the “dot syntax”, where the object name is specified, followed by a “.”, followed by the method name and its parameters, in parentheses.

In this example, `gMoka` is the instance of the Moka Xtra and the method `moka_DestroyJavaObject` is called to destroy the instance of the `JavaDateFormat` class:

```
gMoka.moka_DestroyJavaObject(JavaDateFormat)
```

The equivalent expression, without using the dot syntax, looks like this:

```
moka_DestroyJavaObject(gMoka, JavaDateFormat)
```

## New

**Syntax** `New(xtra “MokaXtra”, UserClassPath,)`

**Parameters** *`UserClassPath`*: String containing the list of access paths to the classes.

On all the platforms the paths must be absolute. The formats vary depending on the operating system:

- On Windows: `"drive:path1\path2"`. The folder separator is backslash, `"\"`. Multiple paths are separated by a semi-colon, ";".
- On Mac OS 8, 9 or X: `"drive:path1:path2"`. The folder separator is colon, ":". Multiple paths are separated by a semi-colon, ";".
- On Mac OS X (using the Unix way): `"/volumes/drive/path1/path2"`. The folder separator is forward slash, `"/"`. The path must start with `"/volumes/"`. Multiple paths are separated by a colon, ":".

There should be no spaces between the paths and delimiters, but there can be spaces within folder names.

If the Java™ classes are contained in a file of type jar or zip, the access path must also include the full name of the jar or zip file.

**IMPORTANT NOTE:**
The path formats specified here are only to be used in Moka Xtra’s `New()` method. If you are passing an absolute file path as a parameter to a Java™ method, you should use `moka_Path()` method to ensure that the path is properly formatted for the user’s operating system.

**Description** Create a new instance of the Moka Xtra xtra. At this time, the Java™ Virtual Machine is started.

On Mac there is only one version of the JVM installed, so you don’t have to worry about where it is located.

On Windows, Moka Xtra uses the following algorithm to locate the JRE™ to start:

1. Moka Xtra checks if there is a JRE™ right beside the movie (or in Director’s folder in authoring mode) – i.e., if the JRE™ folders "lib" and "bin" are at the same level as the application (either your projector or Director).
2. If step 1 fails, it checks if the JRE™ is located in a folder named “JRE”, beside the movie (or in Director’s folder, in authoring mode) - i.e., if the JRE™ folders "lib" and "bin" are in a folder named “JRE”
3. If step 2 fails, it checks in the Windows registry to locate the current default version used on the system.

4 If step 3 fails, it returns an error code (-5001).

**Example**

```
-- on Windows
gMoka = xtra ("MokaXtra").new(the moviepath)
or
gMoka = xtra ("MokaXtra").new(the moviepath&";C:\Utils")
or
gMoka = xtra ("MokaXtra").new(the moviepath&"myClasses.jar")
-- on Mac OS 9
gMoka = xtra ("MokaXtra").new(the moviepath & ":Utils")
or
gMoka = xtra ("MokaXtra").new(the moviepath & ":Utils” &
    “;HardDrive:Jacob:myclasses”)
-- on Mac OS X
gMoka = xtra ("MokaXtra").new("/volumes/Judy/Utils/”)
or
gMoka = xtra ("MokaXtra").new(
    "/volumes/Judy/Utils/:/volumes/Jorge/Tools")
```

**See also** Signature types

## moka_CreateJavaObject

**Syntax** moka_CreateJavaObject(*MokaInstance*, *ClassName*, *Signature*, *Arg1*, *Arg2*,...)

**Parameters**

*MokaInstance*: Instance of the Moka Xtra xtra.

*ClassName*: Name of the class. (N.B. the name of the class is case sensitive). If the class is a class that is distributed with Java™, it is necessary to use the complete name (see example).

*Signature*: String containing the constructor class signature.

*Arg1..Arg2*: Parameters to pass to the method. (Possible Lingo types: Integer, Float, Date, String or List).

**Description** Create an instance of the Moka Xtra xtra. The reference is returned in the form of a string. A negative value is returned in the case of an error.

**Example**

```
-- Create an instance of the java.util.Date class
JavaDate = gMoka.moka_CreateJavaObject("java/util/Date","()V")
if (stringP(JavaDate)) then
   -- manipulate the date
   -- Destroy the instance of the class
   gMoka.moka_DestroyJavaObject(JavaDate)
else
   alert("Error: Java class not created")
end if
```

**See also** Signature types, Data types, moka_DestroyJavaObject

## moka_DestroyJavaObject

**Syntax** `moka_DestroyJavaObject(`*`MokaInstance, ClassInstance`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassInstance`*: Instance of the class to be destroyed (string)

**Description** Destroy an instance of a class. This method must be called subsequent to a call to the `moka_CreateJavaObject` method, or when an instance of a class is returned to Lingo by the methods `moka_CallJavaMethod`, `moka_CallStaticJavaMethod`, `moka_GetJavaMember` or `moka_GetStaticJavaMember`. A negative value is returned in the case of an error.

**Example**

```
-- Create an instance of the java.util.Date class
JavaDate = gMoka.moka_CreateJavaObject("java/util/Date","()V")
-- Retrieve an instance of java.text.DateFormat
JavaDateFormat =
    gMoka.moka_CallStaticJavaMethod("java/text/DateFormat",
    "getDateInstance", "(I)Ljava/text/DateFormat;", 0)
-- manipulate the date
-- Destroy the instances of the classes
gMoka.moka_(JavaDateFormat)
gMoka.moka_DestroyJavaObject(JavaDate)
```

**See also** moka_CreateJavaObject, moka_CallJavaMethod, moka_CallStaticJavaMethod, moka_GetJavaMember, moka_GetStaticJavaMember

## moka_CallJavaMethod

**Syntax** `moka_CallJavaMethod(`*`MokaInstance, ClassInstance, MethodName, Signature, Arg1, Arg2,...`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassInstance`*: Instance of the class (string) that contains the method *`MethodName`*.

*`MethodName`*: Name of the method to call (N.B. The name of the method is case sensitive).

*`Signature`*: String containing the method signature.

*`Arg1..Arg2`*: Parameters to pass to the method. (Possible Lingo types: Integer, Float, Date, String or List).

**Description** Execute a call to a class's method. The data type returned depends on the signature used. If an instance of a class is returned, this instance must be destroyed in the call to the `moka_DestroyJavaObject` method. A negative value is returned in the case of an error.

**Example**

```
-- Create an instance of the java.util.Date class
JavaDate = gMoka.moka_CreateJavaObject("java/util/Date","()V")

-- Retrieve an instance of java.text.DateFormat
JavaDateFormat =
    gMoka.moka_CallStaticJavaMethod("java/text/DateFormat",
    "getDateInstance", "(I)Ljava/text/DateFormat;", 0)
```

```
-- put today's date into the message window
put gMoka.moka_CallJavaMethod(JavaDateFormat, "format",
    "(Ljava/util/Date;)Ljava/lang/String;", JavaDate)

-- Destroy the instances of the classes
gMoka.moka_DestroyJavaObject(JavaDateFormat)
gMoka.moka_DestroyJavaObject(JavaDate)
```

**See also** Signature types, Data types, moka_DestroyJavaObject

## moka_CallStaticJavaMethod

**Syntax** `moka_CallStaticJavaMethod(`*`MokaInstance, ClassName, MethodName, Signature, Arg1, Arg2,...`*`)`

**Parameters**

*`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassName`*: Name of the class (not an instance). Note that the name of the class is case sensitive. If the class is a class that is distributed with Java™, it is necessary to use the complete name (see example).

*`MethodName`*: Name of the method to call (N.B. The name of the method is case sensitive).

*`Signature`*: String containing the method signature.

*`Arg1..Arg2`*: Parameters to pass to the method. (Possible Lingo types: Integer, Float, Date, String or List).

**Description** Execute a call to a class's static method. The data type returned depends on the signature used. If an instance of a class is returned, this instance must be destroyed in the call to the `moka_DestroyJavaObject` method. A negative value is returned in the case of an error.

**Example**

```
-- Retrieve an instance of java.text.DateFormat
JavaDateFormat =
    gMoka.moka_CallStaticJavaMethod("java/text/DateFormat", \
"getDateInstance", "(I)Ljava/text/DateFormat;", 0)
```

**See also** Signature types, Data types, moka_DestroyJavaObject

## moka_GetClassInfo

**Syntax** `moka_GetClasInfo(`*`MokaInstance, ClassName`*`)`

**Parameters**

*`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassName`*: Name of the class (not an instance). Note that the name of the class is case sensitive. If the class is a class that is distributed with Java™, it is necessary to use the complete name (see example).

**Description** Returns a multi-dimensional list of information about a Java™ class. NOTE: This method is used by the JClassBrowser tool and is NOT supported for developer usage.

**Example**

```
-- Retrieve a multi-dimensional list of info about the class
list = moka_GetClassInfo(x,"java.util.Dictionary")
put list
-- [#modifier: "public abstract", #superClass: "java.lang.Object",
```

```
#Constructors: [[]], #methods: [[#name: "put", #Args:
["java.lang.Object", "java.lang.Object"], #return:
"java.lang.Object", #modifier: "public abstract"], [#name:
"get", #Args: ["java.lang.Object"], #return:
"java.lang.Object", #modifier: "public abstract"], [#name:
"size", #Args: [], #return: "int", #modifier: "public
abstract"], [#name: "remove", #Args: ["java.lang.Object"],
#return: "java.lang.Object", #modifier: "public abstract"],
[#name: "elements", #Args: [], #return:
"java.util.Enumeration", #modifier: "public abstract"],
[#name: "keys", #Args: [], #return: "java.util.Enumeration",
#modifier: "public abstract"], [#name: "isEmpty", #Args: [],
#return: "boolean", #modifier: "public abstract"]], #members:
[]]
```

**See also** Signature types, Data types

## moka_GetCurrentJVMInstalled

**Syntax** `moka_GetCurrentJVMInstalled()`

**Parameters** None.

**Description** Returns a string containing the version of the default installed Java™ Virtual Machine (JVM) or Java™ Runtime Engine (JRE™) that will be used by Moka Xtra. If there is no JVM installed, `moka_GetCurrentJVMInstalled()` returns an empty string.

Examples

- Windows: "Sun JRE 1.4"
- Mac OS 8 - 9: "MRJ 2.2.5"
- Mac OS X: "MRJ 3.x"

On windows a user can install more than one JVM. Moka Xtra uses the one declared as the default in the registry:
`HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment\CurrentVersion`

**Example**

```
-- retrieve the default Java Virtual Machine installed on Windows
--  and initalize the Moka Xtra instance
On StartMovie
  curJVM = moka_GetCurrentJVMInstalled()
  if (curJVM contains "Sun JRE 1.3") OR (curJVM contains "Sun JRE
    1.4") then
    jvmInstalled = true
  else
    jvmInstalled = False
  end if

  if jvmInstalled then
    gMoka = New(xtra "MokaXtra", the MoviePath)
  else
    alert "This application requires Java Virtual Machine version
    1.3 or 1.4.  Please install the appropriate JVM from
    http://Java.Sun.com"
    quit
  end if
end StartMovie
```

**See also** New

## moka_GetJavaMember

**Syntax** `moka_GetJavaMember(`*`MokaInstance, ClassInstance, MemberName, Signature`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassInstance`*: Instance of the class (string) that contains the member *`MemberName`*.

*`MemberName`*: Name of the member to read (N.B. The name of the member is case sensitive).

*`Signature`*: String containing the member signature.

**Description** Returns the content from a member of a class instance. The type of data returned depends on the signature used. If an instance of a class is returned, this instance must be destroyed in the call to the `moka_DestroyJavaObject` method. A negative value is returned in the case of an error.

**Example**

```
-- Create an instance of the Test class
Test = gMoka.moka_CreateJavaObject("Test","()V")
-- Retrieve the value
myField = gMoka.moka_GetJavaMember(Test, "field", "I")
-- Detroy the instance of the class
gMoka.moka_DestroyJavaObject(Test)
```

**See also** Signature types, Data types, moka_DestroyJavaObject, moka_SetJavaMember

## moka_GetLastJavaException

**Syntax** `moka_GetLastJavaException()`

**Parameters** None.

**Description** Returns a string containing the last exception thrown by a Java™ function. If no exception was thrown, `moka_GetLastJavaException()` returns "None".

**Example** In message window:

```
put getLastJavaException()
-- "java.util.zip.ZipException: ZIP file must have at least one
    entry"
```

In code:

```
 err = pJavaVM.moka_CallJavaMethod(pZipOutputStreamEx, "zip",
    "(Ljava/lang/String;I)I", theName, theBufferSize)
if err <> 0 then
   if (getLastJavaException() contains "ZIP file must have at
    least one entry") then
     alert "File not found. Check that the path is correct."
   end if
end if
```

## moka_GetStaticJavaMember

**Syntax** `moka_GetStaticJavaMember(`*`MokaInstance, ClassName, MemberName, Signature`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassName`*: Name of the class. (N.B. the name of the class is case sensitive). If the class is a class that is distributed with Java™, it is necessary to use the complete name (see example).

*`MemberName`*: Name of the member to read (N.B. the name of the member is case sensitive).

*`Signature`*: String containing the member signature.

**Example**

```
-- Create an instance of the java.util.Date class
JavaDate = gMoka.moka_CreateJavaObject("java/util/Date","()V")


-- Retrieve the value of member FULL
dateFull =
    gMoka.moka_GetStaticJavaMember("java/text/DateFormat","FULL","
    I")
-- Retrieve an instance of java.text.DateFormat
JavaDateFormat =
    gMoka.moka_CallStaticJavaMethod("java/text/DateFormat",
    "getDateInstance", "(I)Ljava/text/DateFormat;", 0)


-- put today's date into the message window
put gMoka.moka_CallJavaMethod(JavaDateFormat, "format",
    "(Ljava/util/Date;)Ljava/lang/String;", JavaDate)
-- Detroy the instances of the classes
gMoka.moka_DestroyJavaObject(JavaDateFormat)
gMoka.moka_DestroyJavaObject(JavaDate)
```

**See also** Signature types, Data types, moka_DestroyJavaObject, moka_TranlsateToSignature

## moka_Path

**Syntax** `moka_Path(`*`thePath`*`)`

**Parameters** *`thePath`*: String containing a valid absolute path.

**Description** Converts an absolute path to the correct format for Java™ depending on the operating system. This function allows you to use Director's global variables, `the moviePath` and `the applicationPath` within your code and not have to worry about the conversions needed for Java™.

On Windows, this function does not alter the path.

On Mac, the path is converted to a Unix path. The conversion is slightly different depending on if it's OS 8/9 or Mac OS X and depending on which volume the path references (the system volume is treated differently on OS X).

Use `moka_Path()` as you would use a type conversion function, whenever you call a Java™ object that requires a file path. `moka_Path()` ignores all strings that contain a Unix-style path or any forward-slash characters. Therefore, applying `moka_Path()` more than once on the same string will have no effect.

**Example**

```
theFileName = the moviePath & “myfolder:myfile”
-- convert the path within the Moka Xtra method call
gFileObj = gMoka.moka_CreateJavaObject("java.io.File",
"(Ljava/lang/String;)V", moka_Path(theFileName))
```

## moka_Register

**Syntax** `moka_Register(`*`theKey`*`)`

**Parameters** *`theKey`*: String containing a valid license number.

**Description** Change the content of a member of a class instance. A negative value is returned in the case of an error.

**Example**

```
moka_Register(“J1R9-T6K8-2HC2-J391”)
```

**See also** How to register your Moka Xtra license

## moka_SetJavaMember

**Syntax** `moka_SetJavaMember(`*`MokaInstance, ClassInstance, MemberName, Signature, Arg`*`)`

**Parameters**

*`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassInstance`*: Instance of the class (string) that contains the member *`MemberName`*.

*`MemberName`*: Name of the member to read (N.B. The name of the member is case sensitive).

*`Signature`*: String containing the member signature.

*`Arg`*: Value to put into the member instance. (Possible Lingo types: Integer, Float, Date, String or List).

**Description** Change the content of a member of a class instance. A negative value is returned in the case of an error.

**Example**

```
-- Create an instance of the Test class
Test = gMoka.moka_CreateJavaObject("Test","()V")
-- Retrieve a value
myField = gMoka.moka_GetJavaMember(Test, "field", "I")
-- Modify the value in Lingo
myField = myField + specialValue
-- Set the value in the class
myField = gMoka.moka_SetJavaMember(Test, "field", "I")
    ...   -- Call some methods of the class
-- Detroy the instance of the class
gMoka.moka_DestroyJavaObject(Test)
```

**See also** Signature types, Data types, moka_GetJavaMember

## moka_SetStaticJavaMember

**Syntax** `moka_SetStaticJavaMember(`*`MokaInstance, ClassName, MemberName, Signature, Arg`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`ClassName`*: Name of the class. (N.B. The name of the class is case sensitive). If the class is a class that is distributed with Java™, it is necessary to use the complete name (see example).

*`MemberName`*: Name of static member to change (N.B. The name of the member is case sensitive).

*`Signature`*: `String containing the member's signature.`

**Description** Change the content of a static member of a class. A negative value is returned in the case of an error.

**Example**

```
-- change the value of the current date
gMoka.moka_SetStaticJavaMember("java/util/Calendar",
    "DAY_OF_MONTH", "I", 15)
```

**See also** Signature types, Data types, moka_GetStaticJavaMember

## moka_TranlsateToSignature

**Syntax** `moka_TranslateToSignature(`*`MokaInstance,JavaType`*`)`

**Parameters** *`MokaInstance`*: Instance of the Moka Xtra xtra.

*`JavaType`*: string containing a Java™ type.

**Description** Returns the signature of a Java™ variable type. The JClassBrowser tool uses this method to build up the signature of Java™ classes, methods and members.
NOTE: This method is used by the JClassBrowser tool and is NOT supported for developer usage.

**Example**

```
put moka_TranslateToSignature(x,"byte[]")
-- "[B"
```

**See also** Signature types, Data types

# Error Codes

- 5001 Unable to create Java™ virtual machine
- 5002 The Java™ virtual machine was already created
- 5003 No more memory available
- 5004 Unable to find class
- 5005 Unable to find method
- 5006 Invalid call to a Java™ method
- 5007 Invalid argument
- 5008 Invalid class instance
- 5009 Invalid license number

# Index


**C**

Checking for local JVM, 12
Class instance
  creating, 13
  destroying, 15
Closing an Xtra, 15
Contact Information, 8
Creating
  Xtra-Instance, 12

**D**

Data types, 10
Debugging, 15
Delivering to end user, 17
Destroying class instance, 15
Destroying Moka Xtra instance, 15
Dot Syntax, 20

**E**

Error checking, 15
Error Codes, 29

**I**

Installation, 6
Integration New Media, Inc, 8

**J**

Java™ class paths, 7
JClassBrowser tool, 9

**L**

License Agreement, 4
Licensing, 7

**M**

Methods, 16
  moka_CallJavaMethod, 22
  moka_CallStaticJavaMethod, 23
  moka_CreateJavaObject, 21
  moka_DestroyJavaObject, 22
  moka_GetClassInfo, 23
  moka_GetCurrentJVMInstalled, 24
  moka_GetJavaMember, 25
  moka_GetLastJavaException, 25
  moka_GetStaticJavaMember, 26
  moka_Path, 26
  moka_Register, 27
  moka_SetJavaMember, 27
  moka_SetStaticJavaMember, 28
  moka_TranlsateToSignature, 28
  New, 12, 20
Methods Reference, 20
Moka Xtra Instance, 12
moka_Path, 14, 15

**O**

Object types, 10

**P**

paths, 13, 14, 15

**R**

Registration, 7

**S**

Signature
  Java™ class browser tool, 9
Signature types, 10
Standalone projectors, 17
Syntax, 20
System Requirements, 6