Java API: FAQs

Version 1.0

Released: December 29, 2011

Legal Notices

ION Trading U.K. Limited provides the information contained in this document “as is” without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of non-infringement, merchantability or fitness for a particular purpose.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes will be incorporated in new editions of the publication. ION Trading may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-ION Trading Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this ION Trading product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

ION Trading and ION are registered trademarks of ION Trading U.K. Limited and no permission is granted to use such marks other than to identify the products and services of ION Trading U.K. Limited.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Java and JVM are registered trademarks of Oracle America, Inc.

All company, product, and service names are acknowledged.

Contents

1. General Questions

1.1. Who can I call if my question is not answered in these FAQs?

Please contact ION Support.

1.2. Does ION provide training on the ION Java API?

Yes. Ask your manager to contact training@iontrading.com to request Java® API training for you.

Note: ION only provides training to organizations, not to individuals.

2. Component Configuration and Startup

2.1. How can I pass ION Java API configuration properties to my component?

You can do this in various ways.

For example, suppose you want to set `mkv.opt` to `foo`. If you start the component manually (only acceptable for development purposes) you can do one of the following:

- Define a Java system property, for example passing `-Dmkv.opt=foo` on the JVM® command line, or
- Add the following line to `mkv.jinit` file:

  ```
  mkv.opt=foo
  ```

  By default the Java API will look for the `mkv.jinit` file in the process working directory.
- Pass `-opt foo` as a command line parameter. You must omit the initial `mkv.`

Values specified in the `mkv.jinit` file override values provided using the Java properties. Values set on the command line override both.

If the component is managed using the Daemon, configure its properties using the ION System Administrator Tool. For more information, refer to the *Platform Tools: User Guide and Reference*.

2.2. I followed the instructions in 1.1 but from my Java code, I cannot access the Java API configuration properties that I set using the ION System Administrator Tool

Check that you are actually passing the command line arguments to the API when you initialize it. For example, use:

```
MkvQoS qos = new MkvQoS();
qos.setArgs(args);
Mkv.start(qos);
```

where `args` is the `String[]` parameter of your component `main()` method.

2.3. I cannot access the Java API configuration properties

When you access properties from your custom code, you must not specify the initial `mkv.` in the property name. For example:

```
mkv.getProperties().getProperty("opt");
```

obtains the value for the `mkv.opt` property.

2.4. My custom component works fine if I start it manually but it does not connect to the Platform when it is managed by the Daemon

The Daemon passes configuration parameters to components using the command line. Ensure that you are following the advice in Question 2.2.

2.5. I cannot find the log files or they are in an unusual place

The Daemon passes configuration parameters to components using the command line `-init` parameter to point the component to an `mkv.jinit` file created by the Daemon.

Ensure you are following the advice in Question 2.2, otherwise the component will simply ignore the `-init` parameter.

2.6. Huge E2E delays are reported by the Java API (either in PerfMeter or the log files), but my component is responsive. Is the E2E calculation provided by the ION Bus broken?

Check that the `mkv.usemsecs` configuration property is set to the same value for all the components involved. The timer used for log file timestamps is also used for End-to-End (E2E) delays, and spurious values will be calculated if components in a publisher/subscriber pair use different precisions.

2.7. I am using a version of the Java API earlier than 129 and I do not get memory statistics

To produce memory statistics, versions of the ION Java API prior to 129 needed a supporting native library, which was distributed as a shared object. If the following line appears in the PSH log file for your component:

```
PSH::SysMsg: 300,The component will not generate the Mem
Percentage statistics at run time. Please install the jmkv
library
```

This indicates that the ION Java API was unable to find the required shared object. Note that:

- The ZIP file available on the ION Tracker Web site contains a library version for each of the supported operating systems.
  For example, the Linux® library is:

  `linux/libjmkv.so.`
- You must set up your environment so that the JVM can find the correct library.

  For example, on Linux and assuming you unpacked the redistributable in `/opt/ion`, you can set `LD_LIBRARY_PATH` to:

  `/opt/ion/jmkv121p5/linux`
- Check that your component can access the shared library at runtime.
  If you are deploying a managed component, ensure it is accessible and executable by the user as which the Daemon process is running.

2.8. One of the API methods throws an java.lang.IllegalStateException

Some API methods (such as `MkvRecord` constructors) throw this exception if your component attempts to use the API either:

- Before correctly initializing it. That is, before calling `Mkv.start()`

  Or

- After it was already stopped. That is, after calling `Mkv.stop()`

This is by design, because you must initialize the API only once.

3. Subscribing to Platform Data

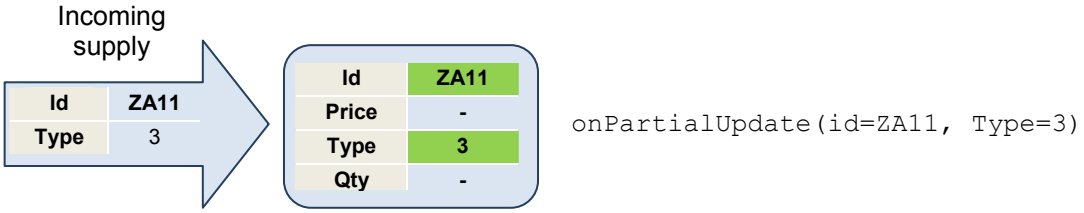
3.1. Is MkvRecordListener.onPartialUpdate() deprecated?

No. However, for most use-cases you need to implement `onFullUpdate()` only. For information about how to find out whether this applies in your case, see Question 3.2

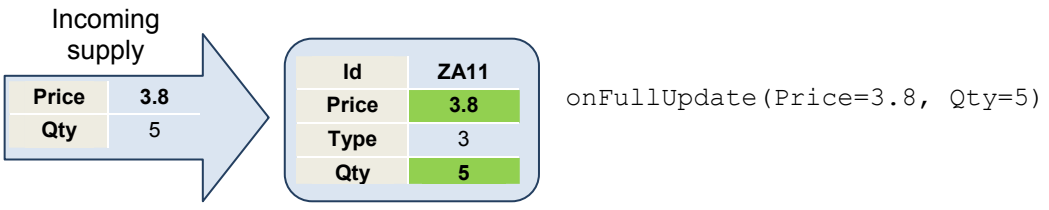
3.2. When is MkvRecordListener.onPartialUpdate() called? And MkvRecordListener.onFullUpdate() ?

When you subscribe to a record, you specify a set of fields for which you want to receive values. Due to the way the ION middleware protocol works, there is no guarantee that your subscriber component will receive all the subscribed fields in a single message, *even if they are supplied by the publisher all at once*.

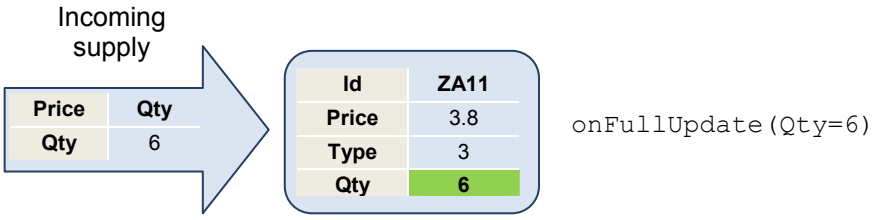
For example, suppose you just subscribed to the `Id`, `Price`, `Type`, and `Qty` fields on a record. The Java API will call `onPartialUpdate()` as soon as an update is received on any number of the subscribed fields (potentially only one). The `MKVSupply` object contains values for the fields received in the current message. If you need all the fields received up to this point, you can access `mkvRecord.getSupply()`:



The first time the API receives values for *all* the subscribed fields, `onPartialUpdate()` is *not* called, but `onFullUpdate()` is called instead. The `MKVSupply` object contains values for the fields that your component has received in the current supply. Again, if you need all the fields received up to this point, you can access `mkvRecord.getSupply()`:



From this point onwards, all the subscribed fields are available. Accordingly, if a subscribed field is updated the API will invoke `onFullUpdate()` while `onPartialUpdate()` is *not* called.
The `MKVSupply` object contains values for the fields received in the current message *only*:



To recap: `onPartialUpdate()` is called (potentially also zero times) until values are received for all the subscribed fields. Then `onFullUpdate()` is called as soon as all the subscribed fields are available and for any subsequent update as well.

Note: If the publisher does not supply some of the fields that the subscriber is subscribing to, then `onFullUpdate()` is *never* invoked on the subscriber. You will need to verify the publisher behavior to check whether this is actually possible. If the publisher is an ION component, check the relevant documentation on the ION Tracker Web site.

3.3. MkvRecordListener.onPartialUpdate() is sometimes called with snapshot set to true and MkvRecordListener.onFullUpdate() is sometimes called with snapshot set to false. Why?

`onPartialUpdate()`/`onFullUpdate()` and the snapshot flag are unrelated concepts.

The snapshot flag is `true` when the supplied fields are received in an SNP message from the ION Bus. For a description of `onPartialUpdate()` versus. `onFullUpdate()` see Question 3.2.

You should ignore the snapshot flag except in some very unusual scenarios.

3.4. The MkvSupply parameter I receive in my MkvRecordListener.onFullUpdate() does not contain all the fields I subscribed to. Why not?

The supply object contains only the subscribed fields whose values have changed since the last time `onFullUpdate()` or `onPartialUpdate()` was received for a given subscription.

3.5. As per 3.4, when MkvRecordListener.onFullUpdate() is called the MkvSupply object contains only the fields I just received. How can I access the other fields I subscribed to?

In addition to the `MkvSupply` parameter, `onFullUpdate()` also receives an `MkvRecord` parameter referring to the record that was just updated by the supply.

You can use `MkvRecord.getSupply()` to access a cached supply object containing all the fields received up to this point.

Because `onFullUpdate()` is not called until all the subscribed records are received, in `onFullUpdate()`, this supply object will always contain all the subscribed fields.

See also Question 3.6 for a similar question involving `onPartialUpdate()`.

3.6. When MkvRecordListener.onPartialUpdate() is called, the MkvSupply object only contains the fields I just received. How can I access all the other fields I received previously for the record?

In addition to the `MkvSupply` parameter, `onPartialUpdate()` also receives an `MkvRecord` parameter referring to the record that has just been updated by the supply.

You can then use `MkvRecord.getSupply()` to access a cached supply object containing all the fields received up to this point.

See also Question 3.5 for a similar question involving `onFullUpdate()`.

3.7. My MkvRecordListener.onPartialUpdate() / onFullUpdate() is called multiple times for the same incoming supply notification

This may occur if you call `MkvRecord.subscribe()` multiple times on a given record passing the same field and different listener objects each time. In this case, the ION Java API will call `onPartialUpdate()` / `onFullUpdate()` on each of the given listeners every time an update is received for the field. This is by design and enables application code to install multiple listeners (possibly with a different implementation) for the same record.

This may lead to multiple notifications if you are repeatedly installing a new listener, for example with:

```
rec.subscribe(new MyRecListener());
```

In simple cases you can guard against multiple subscriptions, for example, by doing the following:

```
if (!rec.isSubscribed())
  rec.subscribe(new MyRecListener());
```

However, sometimes your application does need to install multiple listeners. See Question 3.8.

3.8. MkvRecord.isSubscribed() is returning true even if my listener is not subscribing to the record yet

You may be doing something like the following:

```
if (!rec.isSubscribed())
  rec.subscribe(recListener);
```

The problem with this snippet is that `MkvRecord.isSubscribed()` returns `true` if the record is subscribed using *any* listener object. As a consequence, if somewhere else in your application you subscribe to the same record, even using a different listener, the condition in the `if` will fail.

For most use-cases, you want to subscribe using `recListener` even if the record is already subscribed using other listeners and therefore the guard is not needed. Alternatively, you can explicitly check whether the record was subscribed with a specific listener using, for example:

```
if (!rec.isSubscribed(recListener))
  rec.subscribe(recListener);
```

3.9. I am subscribing records of the same type from different components and I get errors or meaningless data when accessing their fields

Similar types does not mean *the same* type. In particular, multiple components are probably publishing their own types[1] and even if types from different components have exactly the same field names and types, the field IDs may differ.

If they do differ and you access them by field index[2] looking up the ID only once (and thus looking them up for the first component you happen to receive a record from), as soon as you receive a record from the other component you will be using the wrong ID (or to put it another way, the correct ID on the wrong record).

You must change your ID caching logic to take into account the actual record type the IDs refer to.

[1] This is recommended best practice for the ION Platform.

[2] Using `MkvSupply.getObject(int)`, `MkvSupply.getString(int)`, and so on.

3.10. I am subscribing to a chain but I receive record supplies in an order that does not match the order of the records in the chain. How can I change my code to receive field values in the correct order?

The short answer is you cannot do this. Even if you are careful to subscribe to the records in the order in which they appear on the chain, there is simply no guarantee that you will receive the corresponding supply in a *first subscribed, first supplied* order. For example, the order may vary due to:

- The exact timing of the supplies on the publisher
- Whether intervening routers have cached one or more of the records and which fields were cached.

If you need to process the supplied data in a specific order, you must impose an order at the business logic level on the subscriber. For example, you can cache the supplies and process them in the order you want according to an ordered list of records you subscribed.

4. Functions calls and Transactions

4.1. I need to call a function published by an ION-developed component setting the optional parameter “-Xyz”. How can I do this?

Optional parameters published, for example, by ION Common Market Gateways, are simply a convention implemented by most ION components.

The convention mandates that, after all the mandatory parameters, you can pass the optional parameters as (parameter name, parameter value) pairs. Parameter name is the documented name for the optional parameter, including the hyphen (-) prefix. In our case the parameter name is `"-Xyz"`.

For example, suppose a component accepts two mandatory integer parameters and an additional, optional parameter `"-Xyz"`, and you want to pass `"val"` as its value. You can use the following code to set up the parameters and perform the call:

```
MkvSupply args = MkvSupplyFactory.create(
       new Object[] { 6, 7, "-Xyz", "val" });
theFn.call(args, new TheFunctionListener());
```

4.2. How can I implement a function with more than one return value?

Use the `MkvFunction()` constructor accepting an array of return arguments, and then return a supply object with the correct number of values and the correct types, including the main one:

```
MkvFunction f = new MkvFunction("echo4",
       MkvFieldType.STR, // main return value
       false, // Has variable n. of input parameters?
       // Input parameters names and types
       new String[] { "in0", "in1", "in2", "in3"},
       new MkvFieldType[] {
              MkvFieldType.STR,
              MkvFieldType.DATE,
              MkvFieldType.INT, MkvFieldType.REAL },
       false, // Has variable n. of return values?
       // Return value names and types
       //(in addition to the main one)
       new String[] { "out0", "out1", "out2", "out3"},
       new MkvFieldType[] {
              MkvFieldType.STR,
              MkvFieldType.DATE,
              MkvFieldType.INT, MkvFieldType.REAL },
       "help message",
       new MkvFunctionListener () {
       public void onCall(MkvFunctionCallEvent event) {
              MkvSupply sup = event.getArgs();
              try {
                     event.setResult(sup);
              } catch (Exception e) {
                     e.printStackTrace(System.out);
              }
       }
       });
f.publish();
```

4.3. How can I retrieve the original MkvSupply, containing function call parameters, from MkvFunctionCallListener.onResult() / onError() ?

The first parameter to `onResult()` / `onError()` is an `MkvFunctionCallEvent` object.

You can then use the `MkvFunctionCallEvent.getArgs()` method and access the parameters passed to the function call invocation.

4.4. How can I correlate invocations of MkvFunctionCallListener.onResult() / onError() and the corresponding function call invocations?

You can use standard Java techniques.

For example, you can pass any context you need as an additional parameter to your listener class constructor:

```
myFn.call(args, new MyFunctionListener(myContext));

...

Private class MyFunctionListener implements
MkvFunctionCallListener {
      MyCtx ctx;
       private MyFunctionListener(MyContext ctx) {
             this.ctx = ctx;
              //...
       }
       public void onCall(MkvFunctionCallEvent e) {
              if (e.getArgs().getInt(0) > ctx.max)
                    /* ... */;
       }
};
```

4.5. I start the API multiple times from multiple classes / threads but it does not work

The ION Java API is internally multi-threaded and you need to initialize it only once for every process instance (for example, in your `main()` method).

Initializing / starting it multiple times, either from a single thread or multiple threads, from a single class or multiple classes, will raise exceptions.

Customer Information

This section contains information about accessing documentation online and how to contact Technical Support.

Accessing Documents Online

To access documents using a Web browser:

1. Go to the ION Trading® Web site at:

   http://www.iontrading.com/
2. Enter the ION Tracker Web site by logging in to the Client section.
3. Select **Docs & Packages** in the left hand frame.
4. Select one of the following:

   Tree Browser
   Search
   Configuration
   Data Reference

Providing Feedback about Documentation

If you have *comments or suggestions* about ION documentation, send an e-mail to techdocs@iontrading.com.

Contacting Technical Support

If you have a *problem*, you can contact ION Technical Support in one of the following ways:

- Telephone
- Web
- E-mail

Telephone Support

The service hours for all our clients in Europe and the U.S.A. are from 7:00 am to 6:00 pm (local time), Monday to Friday, and from 9:00 am to 5:00 pm in Japan.

EUROPE	7:00 am to 6:00 pm	+ 44 207 398 0222
JAPAN	9:00 am to 5:00 pm	+ 81 3 5404 8488
U.S.A.	7:00 am to 6:00 pm (EST)	+ 1 212 906 0050

Should you be unable to contact ION on any of the numbers above, you can also use the following number for emergencies:

+ 39 050 220 3800

Web Support

You can access the ION Tracker Web site at the following Web site:

http://www.iontrading.com/

The following information and services are available on this site:

- **News**
  The latest announcements on market changes and ION products.
- **Documentation**
  All the guides and release notes available in PDF format.
- **New releases**
  Each new version of a component will be available for download in this section.
- **Development plans and enhancements**
  Current development plans and tracking system of the enhancement requests submitted to ION.

E-mail Support

You can send e-mail to Technical Support at the following address:

support@iontrading.com

When you send e-mail, please provide the following information to accelerate the process:

- What platform is affected
- What component is involved
- The version of the component
- Detailed description of the problem
- Instrument codes involved
- Operators involved
- Detail of any error messages

Please provide logs whenever possible.

To allow us to gather as many elements as possible from the aftermath of crashes, customers are advised to enable the following options on machines running Microsoft® Windows®:

(Dr Watson configuration)

- Dump Symbol Table
- Dump All Thread Contents
- Create Crash Dump File

Remember to send all heavy log files or attachments to the following address:

bigfiles@iontrading.com.

Once you have logged an issue, you will be assigned a reference number. Please ensure you quote this number in all related communications.

Please send all other communications to sales@iontrading.com.