Table of Contents

It is usually good enough if your client creates one single connection to the server. This connection could be created via application login. A login dialog is the right place for creating a connection. But sometimes you need a connection to the server independent of the authenticated user, maybe to retrieve properties or GUI settings.

There are different solutions for this problem. The preferred one would be an anonymous connection. Such connections are supported from DBSecurityManager out of the box, but you need a database with a user table. If you don't have a database, it won't work.

Another solution is the session isolation feature of JVx. We don't have a ready-to-use security manager for this case but it's very easy to implement. The session isolation feature allows using a life cycle object like a common bean, without implicit access to application and session life cycle objects.

There are different things to do! First, we need a custom security manager:

```
public interface IAppConstants extends IConnectionConstants
{
    public static final String QUICKACCESS = PREFIX_CLIENT + "quickaccess";
}

public class QuickSecurityManager extends AbstractSecurityManager
{
    public void validateAuthentication(ISession pSession) throws Exception
    {
        if ("Y".equals(pSession.getProperty(IAppConstants.QUICKACCESS)))
        {
            //Quick-connect is a special connection that allows access
            //to the server, but not the business logic
            //It's not a "real" application connection.

            pSession.setProperty(IConnectionConstants.LIFECYCLENAME,
                                 QuickAccess.class.getName());
        }
        else
        {
           //Username/Password or token authentication
        }
    }
}
```

Configure the security manager in your config.xml

config.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<application>
  <securitymanager>
    <class>com.firstapp.security.QuickSecurityManager</class>
  </securitymanager>
  ...
```

```
</application>
```

Create a connection from your client, e.g., in the constructor of your application.

```
MasterConnection macon = new MasterConnection(createConnection());
macon.setApplicationName(getApplicationName());
macon.setProperty(IAppConstants.QUICKACCESS, "Y");
macon.open();
```

The life cycle object for the QuickAccess connection:

```
@StrictIsolation
public class QuickAccess extends HashMap<String, Object>
{
    public String getProductName()
    {
        return "JVx";
    }
}
```

The object is annotated with StrictIsolation. This annotation marks the objects as object without access to the session life cycle object.

Call an action.

```
String sProductName = (String)macon.callAction("getProductName")
```

Information

The session isolation feature should be used with care because it's possible to get access to the server without “real” authentication. However, it's not risky because it's not possible to call methods that are not available in your isolated life cycle object. There is only one rule for you: never offer internal data. Use the isolation feature to send public data to the client, e.g., product names, version numbers, translation data, and so forth.