Creating a Security Check

Overview

A security check is an object responsible for obtaining credentials from a client and validate them.

Security checks are defined inside **an adapter** and are implemented in Java code. Any adapter can theoretically define a SecurityCheck.
An adapter can either be a *resource* adapter (meaning it serves resources/content to send to the client), a *SecurityCheck* adapter, or **both**.

Prerequisites: Familiarize yourself with the MobileFirst Platform Foundation authentication framework before continuing.
Read the Authentication concepts (../authentication-concepts/) tutorial.

Jump to:

Defining a Security Check

Create a Java or JavaScript adapter (../../adapters/creating-adapters/) or use an exiting one.

> When creating a Java adapter, the default template assumes the adapter will serve **resources**. It is the developer's choice to bundle security checks and resources in the same adapter, or to separate them into distinct adapters.

To remove the default **resource** implementation, delete the files **[AdapterName]Application.java** and **[AdapterName]Resource.java**. Remove the `<JAXRSApplicationClass>` element from **adapter.xml** as well.

In the Java adapter's adapter.xml file, add an XML element called `securityCheckDefinition`. For example:

```
<securityCheckDefinition name="sample" class="com.sample.sampleSecurityCheck">
   <property name="successStateExpirationSec" defaultValue="60"/>
   <property name="failureStateExpirationSec" defaultValue="60"/>
   <property name="maxAttempts" defaultValue="3"/>
</securityCheckDefinition>
```

- The `name` attribute will be the name of your security check.
- The `class` attribute specifies the implementation Java class of the security check. You need to create this class.
- Some SecurityChecks can be configured with a list of `property` elements.

Security Check Implementation

Create the security check's **Java class**. The implementation should extend one of the provided base classes, below.
The parent class you choose will determine the balance between customization and simplicity.

Security Check

`SecurityCheck` is a Java **interface**, defining the minimum required methods to represent the server-side state of a security check. Using this interface alone does not provide any implementation code and it is the sole responsibility of the implementor to handle each scenario.

ExternalizableSecurityCheck

This abstract class implements a basic version of the security-check interface.
It provides, among other options: externalization as JSON, inactivity timeout, expiration countdown and more.

Subclassing this class leaves a lot of flexibility in your Security Check implementation.

> Learn more in the `ExternalizableSecurityCheck` user documentation topic.

CredentialsValidationSecurityCheck

This abstract class extends `ExternalizableSecurityCheck` and implements most of its methods to simplify usage. Two methods are required to be implemented: `validateCredentials` and `createChallenge`.

The `CredentialsValidationSecurityCheck` class is meant for simple flows to need to validate arbitrary credentials in order to grant access to a resource. Aslo provided is a built-in capability to block access after a set number of attempts.

> Learn more in the CredentialsValidationSecurityCheck (../credentials-validation/) tutorials.

UserAuthenticationSecurityCheck

This abstract class extends `CredentialsValidationSecurityCheck` and therefore inherits all of its features.

In addition, the `UserAuthenticationSecurityCheck` class provides the MobileFirst framework an `AuthenticatedUser` object which represents the logged-in user. Methods that are required to be implemented are `createUser`, `validateCredentials` and `createChallenge`.

Also provided is a built-in capability to optionally enable a "Remember Me" login behavior.

> Learn more in the UserAuthentication security check (../user-authentication/) tutorials.

Security Check Configuration

Each security-check implementation class can use a `SecurityCheckConfiguration` class that defines properties available for that security check. Each base `SecurityCheck` class comes with a matching `SecurityCheckConfiguration` class. You can create your own implementation that extends one of the base `SecurityCheckConfiguration` classes and use it for your custom security check.

For example, `UserAuthenticationSecurityCheck`'s `createConfiguration` method returns an instance of `UserAuthenticationSecurityCheckConfig`.

```
public abstract class UserAuthenticationSecurityCheck extends CredentialsValidationSecurityCheck {
  @Override
  public SecurityCheckConfiguration createConfiguration(Properties properties) {
     return new UserAuthenticationSecurityCheckConfig(properties);
  }
}
```

`UserAuthenticationSecurityCheckConfig` enables a property called `rememberMeDurationSec` with a default of `0`.

```
public class UserAuthenticationSecurityCheckConfig extends CredentialsValidationSecurityCheckConfig {

    public int rememberMeDurationSec;

    public UserAuthenticationSecurityCheckConfig(Properties properties) {
        super(properties);
        rememberMeDurationSec = getIntProperty("rememberMeDurationSec", properties, 0);
    }

}
```

These properties can be configured at several levels:

adapter.xml

In the Java adapter's adapter.xml file, inside `<securityCheckDefinition>`, you can add one or more `<property>` elements.
The `<property>` element takes the following attributes:

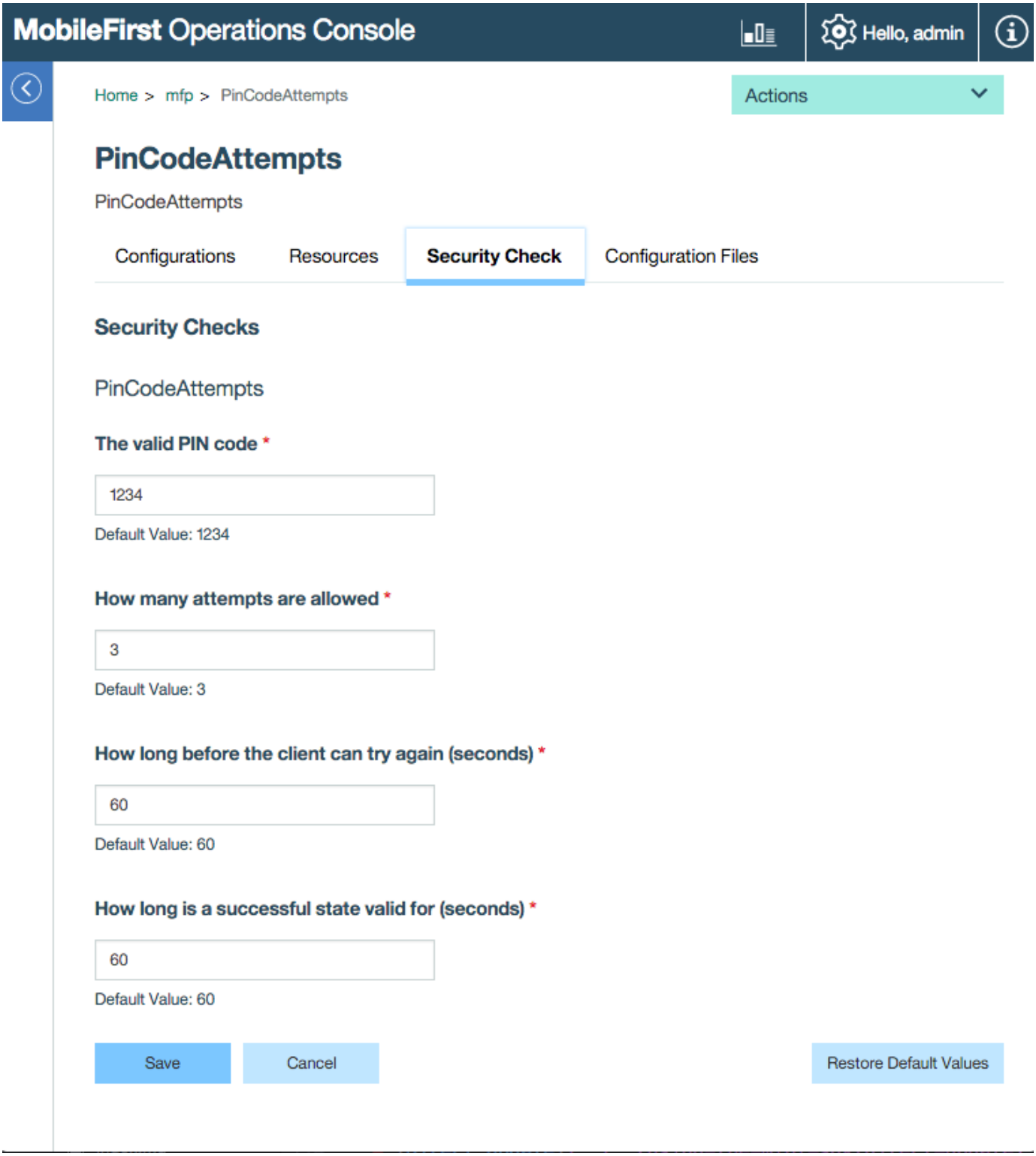
- **name**: The name of the property, as defined in the configuration class.
- **defaultValue**: Overrides the default value defined in the configuration class.
- **displayName**: A friendly name to be displayed in the console.

Example:

```
<property name="maxAttempts" defaultValue="3" displayName="How many attempts are allowed"/>
```

MobileFirst Operations Console - Adapter

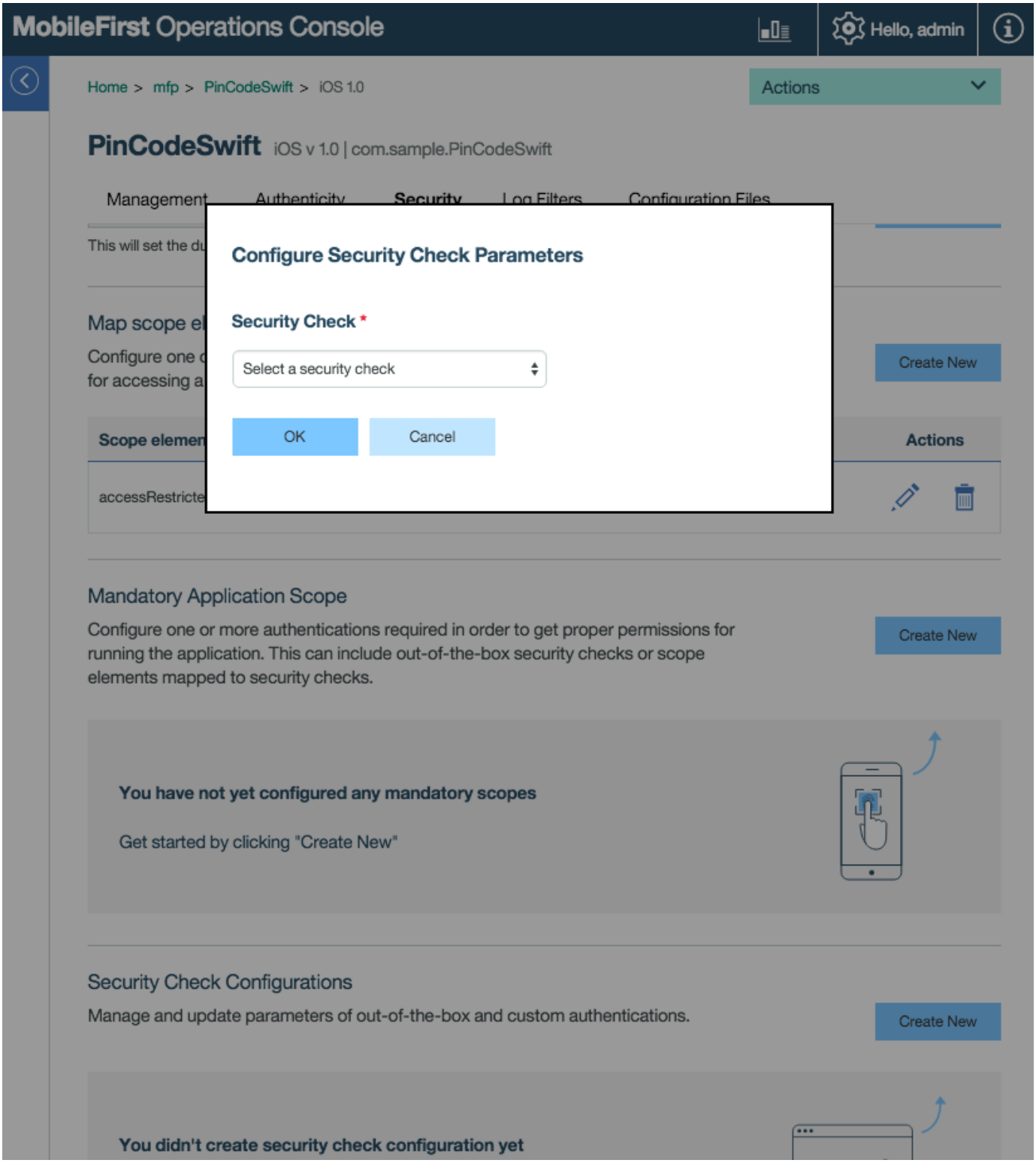
In the MobileFirst Operations Console → **[your adapter] → Security Check tab**, you will be able change the value of any property defined in the adapter.xml.
Note that **only** the properties defined in adapter.xml appear on this screen; properties defined in the configuration class won't appear here automatically.



MobileFirst Operations Console - Application

Property values can also be overridden at the application level.

In the MobileFirst Console → **[your application]** → **Security tab**, under the **Security Check Configurations** section, you can modify the values defined in each security check available.



Predefined Security Checks

Also available are these predefined security checks:

- Application Authenticity (../application-authenticity/)
- Direct Update (../../using-the-mfpf-sdk/direct-update)
- LTPA

Tutorials to follow next

Continue reading about security checks in the following tutorials.
Remember to deploy your adapter when you're done developing or making changes.

- Implementing the CredentialsValidationSecurityCheck (../credentials-validation/).
- Implementing the UserAuthenticationSecurityCheck (../user-authentication/).
- Learn about additional MobileFirst Platform Foundation authentication and security features (../).