Security Check Contract

Overview

Every security check must implement the `com.ibm.mfp.server.security.external.SecurityCheck` interface (the security-check interface). This interface constitutes the basic contract between the security check and the MobileFirst security framework. The security-check implementation must fulfill the following requirements:

- **Functions**: the security check must provide the client-`authorization` and `introspection` functions.
- **State management**: the security check must manage its state, including creation, disposal, and current-state management.
- **Configuration**: the security check must create a security-check configuration object, which defines the supported security-check configuration properties, and validates the types and values of customizations of the basic configuration.

For a complete reference of the security-check interface, see `SecurityCheck` in the API reference (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/SecurityCheck.html?view=kc).

Security-check functions

A security check provides two main functions to the security framework:

Authorization

The framework uses the `SecurityCheck.authorize` method to authorize client requests. When the client requests access to a specific OAuth scope, the framework maps the scope elements into security checks. For each security check in the scope, the framework calls the `authorize` method to request authorization for a scope that contains the scope elements that mapped to this security check. This scope is provided in the method's **scope** parameter.

The security check adds its response to the [] `AuthorizationResponse` object] (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/AuthorizationResponse.html?view=kc) that is passed to it within the response parameter. The response contains the name of the security check and the response type, which can be success, failure, or a challenge (see `AuthorizationResponse.ResponseType` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/AuthorizationResponse.ResponseType.html?view=kc)).

When the response contains a challenge object or custom success or failure data, the framework passes the data to the client's security-check challenge handler within a JSON object. For success, the response also contains the scope for which the authorization was requested (as set in the **scope** parameter), and the expiration time for the granted authorization. To grant the client access to the requested scope, the `authorize` method of each of the scope's security checks must return success, and all expiration times must be later than the current time.

Introspection

The framework uses the `SecurityCheck.introspect` method to retrieve introspection data for a resource server. This method is called for each security check that is contained in the scope for which introspection was requested. As with the `authorize` method, the `introspect` method receives a **scope** parameter that contains the scope elements that mapped to this security check. Before returning the introspection data, the method verifies that the current state of the security check still supports the authorization that was previously granted for this scope. If the authorization is still valid, the `introspect` method adds its response to the IntrospectionResponse object (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/IntrospectionResponse.html?view=kc) that is passed to it within the **response** parameter.

The response contains the name of the security check, the scope for which the authorization was requested (as set in the **scope** parameter), the expiration time for the granted authorization, and the requested custom introspection data. If authorization can no longer be granted (for example, if the expiration time for a previous successful state elapses), the method returns without adding a response.

Note:

- The security framework collects the processing results from the security checks, and passes relevant data to the client. The framework processing is entirely ignorant of the states of the security checks.
- Calls to the `authorize` or `introspect` methods can result in a change in the current state of the security check, even if the expiration time of the current state did not elapse.

> Learn more about the `authorize` and `introspect` methods in the ExternalizableSecurityCheck (../../externalizable-security-check) tutorial.

Security-check state management

Security checks are stateful, meaning that the security check is responsible for tracking and retaining its interaction state. On each authorization or introspection request, the security framework retrieves the states of relevant security checks from external storage (usually, distributed cache). At the end of request processing, the framework stores the security-check states back in external storage.

The security check contract requires that a security check:

- Implement the `java.io.Externalizable` interface. The security check uses this interface to manage the serialization and deserialization of its state.
- Define an expiration time and an inactivity timeout for its current state. The state of the security check represents a stage in the authorization process, and cannot be indefinite. The specific periods for the state's validity and maximum inactivity time are set in the security-check implementation, according to the implemented logic. The security check informs the framework of its selected expiration time and inactivity timeout via the implementation of the `getExpiresAt` and `getInactivityTimeoutSec` methods of the SecurityCheck interface.

Security-check configuration

A security check can expose configuration properties, whose values can be customized both at the adapter and at the application level. The security-check definition of a specific class determines which of the supported configuration properties of this class to expose, and can customize the default values set in the

class definition. The property values can be further customized, dynamically, both for the adapter that defines the security checks, and for each application that uses the check.

A security-check class exposes its supported properties by implementing a `createConfiguration` method, which creates an instance of a security-check configuration class that implements the `com.ibm.mfp.server.security.external.SecurityCheckConfiguration` interface (the security-check configuration interface). This interface complements the `SecurityCheck` interface, and is also part of the security-check contract. The security check can create a configuration object that does not expose any properties, but the `createConfiguration` method must return a valid configuration object and cannot return null. For a complete reference of the security-check configuration interface, see `SecurityCheckConfiguration` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/SecurityCheckConfiguration.html?view=kc).

The security framework calls the security-check's `createConfiguration` method during deployment, which occurs for any adapter or application configuration change. The method's properties parameter contains the properties that are defined in the adapter's security-check definition, and their current customized values (or the default value if there was no customization). The implementation of the security-check configuration should validate the values of the received properties, and provide methods for returning the validation results.

The security-check configuration must implement the `getErrors`, `getWarnings`, and `getInfo` methods. The abstract security-check configuration base class, `SecurityCheckConfigurationBase` (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/html/refjava-mfp-server/html/com/ibm/mfp/server/security/external/checks/impl/SecurityCheckConfigurationBase.html?view=kc) also defines and implements custom `getStringProperty`, `getIntProperty`, and `addMessage` methods. See the code documentation of this class for details.

Note: The names and values of the configuration properties in the security-check definition and in any adapter or application customization, must match the supported properties and allowed values, as defined in the configuration class.

> Learn more about creating custom properties (../#security-check-configuration) in Security Checks.

Last modified on