

N S 2022

ACCELERATING INNOVATION

Security Dashboard API

Melanie Coggan
Software Engineer – Tridium

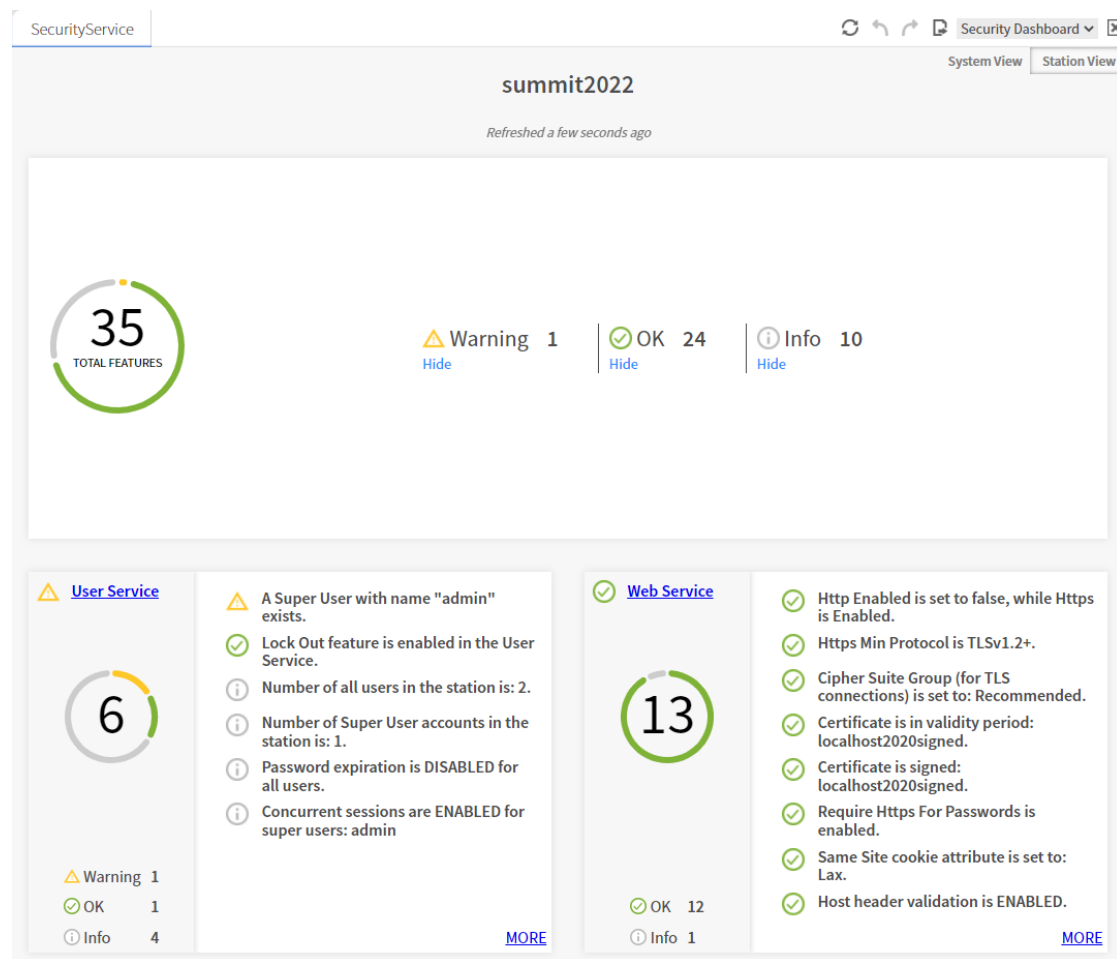


A nighttime photograph of a city skyline with illuminated skyscrapers against a dark blue sky. The text is overlaid on the left side of the image.

NS2022
ACCELERATING INNOVATION

What Is the Security Dashboard?

Bird's Eye View of Security



Security settings all in **one spot**

Easy to read **color coding**

Direct links to components that need to be fixed

Security Dashboard Card

Web Service

13

Alert	1
Warning	1
OK	10
Info	1

- Host header validation is DISABLED.
- Same Site cookie attribute is set to: None.
- Http Enabled is set to false, while Https is Enabled.
- Https Min Protocol is TLSv1.2+.
- Cipher Suite Group (for TLS connections) is set to: Recommended.
- Certificate is in validity period: localhost2020signed.
- Certificate is signed: localhost2020signed.
- Require Https For Passwords is enabled.
- All HTTP security headers are enabled.
- The X-Content-Type-Options HTTP header is set to:

[MORE](#)

Contains many **related items**

Links back to a **single component**

Security Dashboard Item

Consists of a **status**, **summary**, and **description**

Web Service

13

- Alert 1
- Warning 1
- OK 10
- Info 1

Host header validation is DISABLED.

Same Site cookie attribute is set to: None.

Http Enabled is set to false, while Https is Enabled.

Https Min Protocol is TLSv1.2+.

Cipher Suite Group (for TLS connections) is set to: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256. Recommended.

Certificate is in validity period: local.

Certificate is signed: localhost2020si.

Require Https For Passwords is enabled.

All HTTP security headers are enabled.

The X-Content-Type-Options HTTP header is set to: no-sniff.

Host header validation is DISABLED.
The Host header is used to generate certain redirects and URLs. Turning on validation and setting a list of allowed values for the Host header ensures that these URLs are being constructed with known, secure values.

Same Site cookie attribute is set to: None.
Setting Same Site to "Lax" or "Strict" is recommended to help protect against cross-origin information leakage and cross-site request forgery attacks.

Http Enabled is set to false, while Https is Enabled.
Setting Https Enabled to true is recommended to enable secure connections.

Https Min Protocol is TLSv1.2+.

MORE

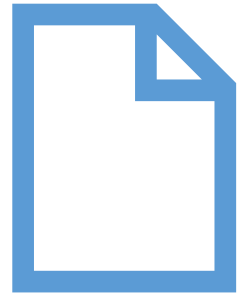
What's the Benefit?



Makes security easier!

Why We're Here

Public API allows custom modules to:



Add a **new card**
to the dashboard



Add items to an
existing card



NS2022
ACCELERATING INNOVATION

Sample Module

HTTP Client

HttpClientDevice

The screenshot displays a web management interface with a navigation pane on the left and a property sheet on the right. The navigation pane shows a tree view under 'Station (summit2022)' with 'Config' expanded to 'Services' and 'Drivers'. 'HttpClientNetwork' is expanded, and 'SAMLMetadataFetcher' is selected and highlighted. The property sheet on the right shows the configuration for 'SAMLMetadataFetcher (Http Client Device)'. Several fields are highlighted with orange boxes:

- Auth Type:** Set to 'httpClient' and 'NoHttpAuth'.
- Config:** Set to 'No Http Auth'.
- Transport:** Set to 'Http Transport'.
- Transport Type:** Set to 'httpClient' and 'OkHttpTransport'.
- Comm:** Set to 'Ok Http Transport'.
- Http2 Enabled:** Set to 'false'.
- Tls Connection Spec:** Set to 'Restricted_tls'.
- Ping Address:** Set to 'Http Client Ping Address'.
- Health:** Set to 'Http Response Health'.
- Address:** Set to 'https://localhost/saml/samlrp/metadat...'.
- Mode:** Set to 'Secure'.
- Host Address:** Set to 'localhost'.
- Port:** Set to '443'.
- Path:** Set to '/saml/samlrp/metadata?scheme=SAMLAuthent:'.

HttpClient

The screenshot displays a configuration interface for an HttpClient. On the left, a navigation pane shows a tree view with 'HttpClient' selected under the 'Apps' folder. On the right, the 'Property Sheet' for 'HttpClient (Http Client)' is shown with several settings highlighted by orange boxes:

- Enabled:** true
- Out:** (Empty field)
- Health:** Http Response Health
- Address:** https://station1.domain.com/saml/saml...
- Mode:** Secure
- Host Address:** station1.domain.com
- Port:** 443 [-1 - 65536]
- Path:** /saml/samlrp/metadata?scheme=SAMLAuthent...
- Method:** GET
- Headers:** Http Headers
- Parameters:** Http Parameters
- Http Tuning Policy:** Http Standalone Tuning Policy
- Authenticator:** Http Authenticator
- Auth Type:** httpClient NoHttpAuth
- Config:** No Http Auth
- Transport:** Http Transport
- Transport Type:** httpClient OkHttpTransport
- Comm:** Ok Http Transport
- Http2 Enabled:** false
- Tls Connection Spec:** Modern_tls
- Request Body:** Request Body

HttpClientService

The screenshot displays the configuration interface for the HttpClientService. On the left, the navigation pane shows the hierarchy: My Network > My Host (summit2022) > Config > Services > HttpClientService. The 'Services' folder and 'HttpClientService' are highlighted with an orange box. On the right, the 'Property Sheet' for 'HttpClientService (Http Client Service)' is shown. The 'Enable Non Driver Clients' property is highlighted with an orange box and is set to 'false'.

Property Sheet	
HttpClientService (Http Client Service)	
Status	{ok}
Fault Cause	
Enabled	<input checked="" type="checkbox"/> true
Enable Non Driver Clients	<input type="checkbox"/> false
SMA Expiration Monitor	SMA Expiration Monitor
Global Request Throttle	Request Throttle
Client Request History	Http Client Request History

Security Dashboard API

BISecurityDashboardItemProvider

Adds items to an existing card.

BISecurityDashboardProvider

Adds a new card.

BISecurityDashboardProviderAgent

Adds a new card. Lets you split your code.



NS2022
ACCELERATING INNOVATION

Add Items to an Existing Card

BI Security Dashboard Item Provider

Web Service Card

✓ [Web Service](#)

13

✓ OK 12

ⓘ Info 1

- ✓ Http Enabled is set to false, while Https is Enabled.
- ✓ Https Min Protocol is TLSv1.2+.
- ✓ Cipher Suite Group (for TLS connections) is set to: Recommended.
- ✓ Certificate is in validity period: localhost2040signed.
- ✓ Certificate is signed: localhost2040signed.
- ✓ Require Https For Passwords is enabled.
- ✓ Same Site cookie attribute is set to: Lax.
- ✓ Host header validation is ENABLED.
- ✓ All HTTP security headers are enabled.
- ⓘ The X-Content-Type-Options HTTP header is set to:

[MORE](#)

BI Security DashboardItemProvider

Must implement **BI Agent**

Must be an agent on a
BI Security DashboardProvider or
BI Security DashboardProviderAgent

BI SecurityDashboardItemProvider

```
@NiagaraType
public interface BI SecurityDashboardItemProvider
    extends BInterface
{
    ...

    /**
     * Sets the object from which security dashboard items will be generated. The default
     * implementation does nothing and is suitable for types that implement
     * BI SecurityDashboardProvider (because that type is itself the items source) or types that will
     * retrieve the source(s) themselves.
     */
    → default void setSecurityDashboardItemsSource(BIObject object)
    {
    }

    /**
     * Returns the version of the security dashboard items.
     */
    → int getSecurityDashboardItemsVersion();

    /**
     * Returns a List of SecurityDashboardItem objects.
     * @see SecurityDashboardItem
     * @param cx The context, used for localization
     * @return List of the SecurityDashboardItems
     */
    → List<SecurityDashboardItem> getSecurityDashboardItems(Context cx);
}
```

BI Security DashboardItemProvider

```
@NiagaraType(agent = @AgentOn(types = "web:WebService"))
public class BHttpClientSecurityDashboardItemProvider
    extends BObject
    implements BI SecurityDashboardItemProvider, BIAgent
{
    @Override
    public void setSecurityDashboardItemsSource(BIObject object)
    {
        webservice = object.as(BWebService.class);
        builder = new SecurityDashboardItemBuilder(this);
    }

    @Override
    public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }

    @Override
    public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();

        checkClientsForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkClientsForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkClientsForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);
        checkDevicesForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkDevicesForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkDevicesForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);

        BHttpClientService service = BHttpClientService.service();
        if (service != null && service.getEnableNonDriverClients())
        {
            checkClientsForWarning(items, NON_DRIVER_CLIENTS_SUMMARY_OK, NON_DRIVER_CLIENTS_SUMMARY_WARNING,
                NON_DRIVER_CLIENTS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isNonDriverClient, cx);
        }

        return Collections.unmodifiableList(items);
    }
}
```


BI Security DashboardItemProvider

```
@NiagaraType(agent = @AgentOn(types = "web:WebService"))
public class BHttpClientSecurityDashboardItemProvider
    extends BObject
    implements BSecurityDashboardItemProvider, BIAgent
{
    @Override
    public void setSecurityDashboardItemsSource(BIObject object)
    {
        webservice = object.as(BWebService.class);
        builder = new SecurityDashboardItemBuilder(this);
    }

    @Override
    public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }

    @Override
    public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();

        checkClientsForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkClientsForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkClientsForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);
        checkDevicesForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkDevicesForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkDevicesForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);

        BHttpClientService service = BHttpClientService.service();
        if (service != null && service.getEnableNonDriverClients())
        {
            checkClientsForWarning(items, NON_DRIVER_CLIENTS_SUMMARY_OK, NON_DRIVER_CLIENTS_SUMMARY_WARNING,
                NON_DRIVER_CLIENTS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isNonDriverClient, cx);
        }

        return Collections.unmodifiableList(items);
    }
}
```

“object” is a **BWebService** because we’re an agent on **BWebService**, which is a **BISecurityDashboardProvider**

BI Security Dashboard Item Provider

```
@NiagaraType(agent = @AgentOn(types = "web:WebService"))
```

```
public class BHttpClientSecurityDashboardItemProvider  
    extends BObject  
    implements BSecurityDashboardItemProvider, BIAgent
```

```
{  
    @Override  
    public void setSecurityDashboardItemsSource(BIObject object)  
    {  
        webservice = object.as(BWebService.class);  
        builder = new SecurityDashboardItemBuilder(this);  
    }  
}
```

```
@Override  
public int getSecurityDashboardItemsVersion()  
{  
    return SECURITY_DASHBOARD_ITEMS_VERSION;  
}
```

```
→ @Override  
public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)  
{  
    List<SecurityDashboardItem> items = new ArrayList<>();
```

```
    checkClientsForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);  
    checkClientsForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);  
    checkClientsForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);  
    checkDevicesForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);  
    checkDevicesForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);  
    checkDevicesForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);
```

```
    BHttpClientService service = BHttpClientService.service();  
    if (service != null && service.getEnableNonDriverClients())  
    {  
        checkClientsForWarning(items, NON_DRIVER_CLIENTS_SUMMARY_OK, NON_DRIVER_CLIENTS_SUMMARY_WARNING,  
            NON_DRIVER_CLIENTS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isNonDriverClient, cx);  
    }  
}
```

```
    return Collections.unmodifiableList(items);  
}
```

```
private void checkClientsForWarning(final List<SecurityDashboardItem> items, String okSummaryKey,  
    String warningSummaryKey, String descriptionKey,  
    Predicate<BIHttpClient> secCheck, Context cx)  
{  
    List<String> warningClients = HttpClientRegister.getClientRegister().stream()  
        .filter(secCheck)  
        .map(commClient -> getClientName(commClient, cx))  
        .sorted()  
        .collect(Collectors.toList());  
  
    if (!warningClients.isEmpty())  
    {  
        items.add(builder.makeWarning()  
            .withSummary(warningSummaryKey, String.join(", ", warningClients))  
            .withDescription(descriptionKey));  
    }  
    else  
    {  
        items.add(builder.makeOk()  
            .withSummary(okSummaryKey, String.join(", ", warningClients))  
            .withDescription(descriptionKey));  
    }  
}
```

BI Security Dashboard Item Provider

```
@NiagaraType(agent = @AgentOn(types = "web:WebService"))
```

```
public class BHttpClientSecurityDashboardItemProvider  
    extends BObject  
    implements BSecurityDashboardItemProvider, BIAgent
```

```
{  
    @Override  
    public void setSecurityDashboardItemsSource(BIObject object)  
    {  
        webservice = object.as(BWebService.class);  
        builder = new SecurityDashboardItemBuilder(this);  
    }  
}
```

```
@Override  
public int getSecurityDashboardItemsVersion()  
{  
    return SECURITY_DASHBOARD_ITEMS_VERSION;  
}
```

```
→ @Override  
public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)  
{  
    List<SecurityDashboardItem> items = new ArrayList<>();
```

```
    checkClientsForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);  
    checkClientsForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);  
    checkDevicesForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);  
    checkDevicesForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);  
    checkDevicesForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);
```

```
    BHttpClientService service = BHttpClientService.service();  
    if (service != null && service.getEnableNonDriverClients())  
    {  
        checkClientsForWarning(items, NON_DRIVER_CLIENTS_SUMMARY_OK, NON_DRIVER_CLIENTS_SUMMARY_WARNING,  
            NON_DRIVER_CLIENTS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isNonDriverClient, cx);  
    }  
}
```

```
return Collections.unmodifiableList(items);  
}
```

```
private void checkClientsForWarning(final List<SecurityDashboardItem> items, String okSummaryKey,  
    String warningSummaryKey, String descriptionKey,  
    Predicate<BIHttpCommClient> secCheck, Context cx)  
{  
    List<String> warningClients = HttpClientRegister.getClientRegister().stream()  
        .filter(secCheck)  
        .map(commClient -> getClientName(commClient, cx))  
        .sorted()  
        .collect(Collectors.toList());  
  
    if (!warningClients.isEmpty())  
    {  
        items.add(builder.makeWarning()  
            .withSummary(warningSummaryKey, String.join(", ", warningClients))  
            .withDescription(descriptionKey));  
    }  
    else  
    {  
        items.add(builder.makeOk()  
            .withSummary(okSummaryKey, String.join(", ", warningClients))  
            .withDescription(descriptionKey));  
    }  
}
```

BI SecurityDashboardItemProvider

```
@NiagaraType(agent = @AgentOn(types = "web:WebService"))
public class BHttpClientSecurityDashboardItemProvider
    extends BObject
    implements BI SecurityDashboardItemProvider, BIAgent
{
    @Override
    public void setSecurityDashboardItemsSource(BIObject object)
    {
        webservice = object.as(BWebService.class);
        builder = new SecurityDashboardItemBuilder(this);
    }

    @Override
    public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }


    @Override
    public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();


        checkClientsForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkClientsForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkClientsForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);
        checkDevicesForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isInsecure, cx);
        checkDevicesForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingBasicAuth, cx);
        checkDevicesForWarning(items, COMPAT_TLS_SUMMARY, COMPAT_TLS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isUsingCompatibleTls, cx);




        BHttpClientService service = BHttpClientService.service();
        if (service != null && service.getEnableNonDriverClients())
        {
            checkClientsForWarning(items, NON_DRIVER_CLIENTS_SUMMARY_OK, NON_DRIVER_CLIENTS_SUMMARY_WARNING,
                NON_DRIVER_CLIENTS_DESCRIPTION, BHttpClientSecurityDashboardItemProvider::isNonDriverClient, cx);
        }










        return Collections.unmodifiableList(items);
    }
}
```

The Result

 [Web Service](#)



 Warning 3
 OK 12
 Info 1

-  Secure Encrypted Connection (TLS/HTTPS) is NOT USED for clients: SAMLMetadataFetcher
-  HTTP Basic Authentication is being used for clients: SAMLMetadataFetcher
-  Compatible TLS is enabled for clients: SAMLMetadataFetcher
-  Http Enabled is set to false, while Https is Enabled.
-  Https Min Protocol is TLSv1.2+.
-  Cipher Suite Group (for TLS connections) is set to: Recommended.
-  Certificate is in validity period: localhost2040signed.
-  Certificate is signed: localhost2040signed.
-  Require Https For Passwords is enabled.

[MORE](#)

A nighttime photograph of a city skyline with illuminated buildings against a dark blue sky. The text 'NS 2022' is overlaid in large white letters, with 'ACCELERATING INNOVATION' in smaller white letters below it.

NS 2022
ACCELERATING INNOVATION

Create a New HttpClient Card

*BHttpClient as a
BISecurityDashboardProvider*

BISecurityDashboardProvider

Extends **BISecurityDashboardItemProvider**

Is not a **BIAgent**

Specifies a **hyperlink ord** and **header name**

BI Security Dashboard Provider

@NiagaraType

```
public interface BISEcurityDashboardProvider
```

```
    extends BISEcurityDashboardItemProvider
```

```
{  
    /**  
     * Get the display name of the dashboard section.  
     * @param cx The context, used for localization  
     * @return localization info for the dashboard section display name  
     */
```

→ LexiconFormatInfo getSecurityDashboardSectionHeader(Context cx);

```
    /**  
     * Returns the ORD of the service/security item that should be hyperlinked to  
     * for more details. This ORD will be considered relative to the  
     * {@code BISEcurityDashboardProvider} instance, so in most cases,  
     * {@code relativizeToSession()} should be used.  
     *  
     * @return the ORD to the service/security item. This may also return  
     * {@code null} or {@code BOrd.DEFAULT} to indicate that no hyperlink should  
     * be shown.  
     */
```

→ BOrd getSecurityDashboardSectionHyperlinkOrd();

```
}
```

Must also implement
getSecurityDashboardItemsVersion(),
getSecurityDashboardItems(), but not
setSecurityDashboardItemsSource()

BI SecurityDashboardProvider

```
public class BHttpClient
    extends BComponent
    implements BIHttpCommClient, BIHttpCommClientTransport, BIHttpCommClientWrite, BI SecurityDashboardProvider
{
    @Override
    → public LexiconFormatInfo getSecurityDashboardSectionHeader(Context cx)
    {
        return LexiconFormatInfo.make(TYPE, SECTION_HEADER, getClientName(this, cx));
    }

    @Override
    → public BOrd getSecurityDashboardSectionHyperl
    {
        return getNavOrd().relativizeToSession();
    }

    @Override
    → public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }

    @Override
    → public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();
        checkClientForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, isInsecure(), cx);
        checkClientForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, isUsingBasicAuth(), cx);
        checkClientForWarning(items, COMPAT_TLS_SUMMARY_OK, COMPAT_TLS_SUMMARY_WARNING, COMPAT_TLS_DESCRIPTION, isUsingCompatibleTls(), cx);

        return Collections.unmodifiableList(items);
    }
}
```

```
private static final String SECTION_HEADER = "securityDashboard.httpClient.sectionHeader.client";

module.lexicon
securityDashboard.httpClient.sectionHeader=HTTP/WebSocket client ({0})
```

BI SecurityDashboardProvider

```
public class BHttpClient
    extends BComponent
    implements BIHttpCommClient, BIHttpCommClientTransport, BIHttpCommClientWrite, BI SecurityDashboardProvider
{
    @Override
    public LexiconFormatInfo getSecurityDashboardSectionHeader(Context cx)
    {
        return LexiconFormatInfo.make(TYPE, SECTION_HEADER, getClientInfo());
    }

    @Override
    public BOrd getSecurityDashboardSectionHyperlinkOrd()
    {
        return getNavOrd().relativizeToSession();
    }

    @Override
    public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }

    @Override
    → public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();
        checkClientForWarning(items, INSECURE_SUMMARY_OK, INSECURE_SUMMARY_WARNING, INSECURE_DESCRIPTION, isInsecure(), cx);
        checkClientForWarning(items, BASIC_SUMMARY_OK, BASIC_SUMMARY_WARNING, BASIC_DESCRIPTION, isUsingBasicAuth(), cx);
        checkClientForWarning(items, COMPAT_TLS_SUMMARY_OK, COMPAT_TLS_SUMMARY_WARNING, COMPAT_TLS_DESCRIPTION, isUsingCompatibleTls(), cx);

        return Collections.unmodifiableList(items);
    }
}
```

```
private void checkClientForWarning(final List<SecurityDashboardItem> items,
    String okSummaryKey, String warningSummaryKey, String descriptionKey,
    boolean isInsecure, Context cx)
{
    if (isInsecure)
    {
        items.add(builder.makeWarning()
            .withSummary(warningSummaryKey)
            .withDescription(descriptionKey));
    }
    else
    {
        items.add(builder.makeOk()
            .withSummary(okSummaryKey)
            .withDescription(descriptionKey));
    }
}
```


The Result

The screenshot displays a security audit result for an HTTP/WebSocket client. On the left, a circular progress indicator shows a score of 3, with a warning icon and a checkmark icon. Below the score, a legend indicates 1 warning and 2 OK items. The main content area lists three findings: a warning for 'Compatible TLS is enabled', and two OK items for 'Secure Encrypted Connection (TLS/HTTPS) is USED' and 'HTTP Basic Authentication is NOT USED'. A 'MORE' link is located at the bottom right.

HTTP/WebSocket client (.HttpClient)

3

- ⚠ Warning 1
- ✅ OK 2

- ⚠ Compatible TLS is enabled.
- ✅ Secure Encrypted Connection (TLS/HTTPS) is USED.
- ✅ HTTP Basic Authentication is NOT USED.

[MORE](#)

A nighttime photograph of a city skyline with illuminated buildings against a dark blue sky. The text 'NS 2022' is overlaid in large white letters, with 'ACCELERATING INNOVATION' in smaller white letters below it.

NS 2022
ACCELERATING INNOVATION

Create a New HttpClient Card

*BHttpClientService as a
BISecurityDashboardProviderAgent*

BI SecurityDashboardProviderAgent

Extends **BI SecurityDashboardProvider**

Is a **BI Agent**

Specifies a **hyperlink ord**, a **header name**, and an **item source**

BI Security Dashboard Provider Agent

```
@NiagaraType
public interface BISEcurityDashboardProviderAgent
    extends BIAgent, BISEcurityDashboardProvider
{
    //region /*+ ----- BEGIN BAJA AUTO GENERATED CODE ----- +*/

    Type TYPE = Sys.LoadType(BISEcurityDashboardProviderAgent.class);

    //endregion /*+ ----- END BAJA AUTO GENERATED CODE ----- +*/
}
```

Must implement
`getSecurityDashboardItemsVersion()`,
`getSecurityDashboardSectionHeader()`,
`getSecurityDashboardSectionHyperlinkOrd()`,
`getSecurityDashboardItems()`, and
`setSecurityDashboardItemsSource()`

BI SecurityDashboardProviderAgent

```
@NiagaraType(agent = @AgentOn(types = "httpClient:HttpClientService"))
public final class BHttpClientSecurityDashboardAgent extends BObject implements BI SecurityDashboardProviderAgent
{
    @Override
    → public void setSecurityDashboardItemsSource(BIObject object)
    {
        httpClientService = object.as(BHttpClientService.class);
        builder = new SecurityDashboardItemBuilder(this);
    }

    @Override
    → public LexiconFormatInfo getSecurityDashboardSectionHeader(Context cx)
    {
        return LexiconFormatInfo.make(TYPE, SECTION_HEADER);
    }

    @Override
    → public BOrd getSecurityDashboardSectionHyperlinkOrd()
    {
        return httpClientService.getNavOrd().relativizeToSession();
    }

    @Override
    → public int getSecurityDashboardItemsVersion()
    {
        return SECURITY_DASHBOARD_ITEMS_VERSION;
    }

    @Override
    → public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        ...
    }
}
```

object is a **BHttpClientService**
because we're an agent on that type

```
private static final String SECTION_HEADER = "securityDashboard.httpClient.sectionHeader";
module.lexicon
securityDashboard.httpClient.sectionHeader=HTTP/WebSocket clients
```

BI Security Dashboard Provider Agent

```
@NiagaraType(agent = @AgentOn(types = "httpClient:HttpClientService"))
public final class BHttpClientSecurityDashboardAgent extends BObject implements BISEcurityDashboardProviderAgent
{
    ...


    @Override
    → public List<SecurityDashboardItem> getSecurityDashboardItems(Context cx)
    {
        List<SecurityDashboardItem> items = new ArrayList<>();


        checkClientsForWarning(items, INSECURE_SUM_OK, INSECURE_SUM_WARNING, INSECURE_DESC, BHttpClientSecurityDashboardAgent::isInsecure, cx);
        checkClientsForWarning(items, BASIC_SUM_OK, BASIC_SUM_WARNING, BASIC_DESC, BHttpClientSecurityDashboardAgent::isUsingBasicAuth, cx);
        checkClientsForWarning(items, COMPAT_TLS_SUM_OK, COMPAT_TLS_SUM_WARNING, COMPAT_TLS_DESC, BHttpClientSecurityDashboardAgent::isUsingCompatibleTls, cx);
        checkDevicesForWarning(items, INSECURE_SUM_OK, INSECURE_SUM_WARNING, INSECURE_DESC, BHttpClientSecurityDashboardAgent::isInsecure, cx);
        checkDevicesForWarning(items, BASIC_SUM_OK, BASIC_SUM_WARNING, BASIC_DESC, BHttpClientSecurityDashboardAgent::isUsingBasicAuth, cx);
        checkDevicesForWarning(items, COMPAT_TLS_SUM_OK, COMPAT_TLS_SUM_WARNING, COMPAT_TLS_DESC, BHttpClientSecurityDashboardAgent::isUsingCompatibleTls, cx);



        if (httpClientService.getEnableNonDriverClients())
        {
            checkClientsForWarning(items, NON_DRIVER_SUM_OK, NON_DRIVER_SUM_WARNING, NON_DRIVER_DESC, BHttpClientSecurityDashboardAgent::isNonDriverClient, cx);
        }







        return Collections.unmodifiableList(items);
    }
}
```


The Result

 [HTTP/WebSocket clients](#)



 Warning 1
 OK 5

-  Secure Encrypted Connection (TLS/HTTPS) is NOT USED for clients: HttpClient
-  HTTP Basic Authentication is NOT USED for any clients
-  Compatible TLS is disabled for all clients
-  Secure Encrypted Connection (TLS/HTTPS) is USED for all devices
-  HTTP Basic Authentication is NOT USED for any device
-  Compatible TLS is disabled for all devices

[MORE](#)

A nighttime photograph of a city skyline with illuminated skyscrapers against a dark blue sky. The image is used as a background for the left side of the slide.

NS2022
ACCELERATING INNOVATION

Which Interface Should I Use?

Summary

BISecurityDashboardItemProvider

Adds items to existing card

BISecurityDashboardProvider

Adds a new card

BISecurityDashboardProviderAgent

Adds a new card, new class

How to Choose

How do you want the dashboard to look?

How many cards will it create?

What component will it link back to?

What do the individual items look like?

A nighttime photograph of a city skyline with several skyscrapers illuminated against a dark blue sky. The buildings have glowing windows, and some have distinctive architectural features like rounded tops or setbacks.

NS 2022

ACCELERATING INNOVATION

Questions?