

NIAGARA N4.8 BREAKING CHANGES

A breaking change is a change in a public API which results in dependent products not compiling at build time, or a change in expected functionality of our software which results in third party dependent products not working as intended at run-time.

Examples might be:

- A change in an interface or class definition which causes a compile-time failure in dependent third-party code
- A change is made which requires an explicit request of a security permission at run-time in order to perform a certain action.
- Removing a method or class (even those marked as @deprecated)
- A change that affects the persistence (bog format) of a station (breaking changes to the station serialization/deserialization)
- A network/wire compatibility change (for example, changes to fox communication)

BREAKING CHANGE: SESSION TIMEOUT API

In Niagara 4.8, the return types of 2 method signatures related to the session timeout API were changed.

- The return type of `BWbShell.notifyTimeout(BWidget, BSession)` has changed from void to boolean. Any subclass overriding this method must now provide a return value. This method is intended to notify the user when a session is about to expire and give them the option to continue. This method should return true if the user elected to continue the session, and false otherwise.
- The return type of `BFoxProxySession.NotifyListener.onNotify(BFoxProxySession)` has changed from void to boolean. Any class implementing `NotifyListener` must now provide a return value for this method. This method is intended to notify the user when a session is about to expire, and give them the option to continue. This method should return true if the user elected to continue the session, and false otherwise.

The most likely use case for these methods is a `BWbView` that implements `NotifyListener` and calls `notifyTimeout()` on the Tridium implementation of `BWbShell`, like below

```
public class BMyView extends BWbView implements NotifyListener
{
    @Override
    public void onNotify(BFoxProxySession session)
    {
        getWbShell().notifyTimeout(this, session);
    }
}
```

Cases like this should simply be updated to return the value from `BWbShell.notifyTimeout()` like below

Honeywell

```
public class BMyView extends BWbView implements NotifyListener
{
    @Override
    public boolean onNotify(BFoxProxySession session)
    {
        return getWbShell().notifyTimeout(this, session);
    }
}
```

BREAKING CHANGE: 3RD PARTY LIBRARIES REMOVED FROM TEST-WB/ TEST-SE

IMPACT: This change affects Niagara Java developers with unit tests developed using Niagara versions prior to 4.8 who directly used classes from the removed 3rd party libraries.

When attempting to compile these unit tests under Niagara 4.8, one or more "package does not exist" or "class not found" errors will be generated. For example:

Example error output

```
C:\Users\Developer\Niagara4.8\tridium>gradlew moduleTestJar
Configuration on demand is an incubating feature.

> Task :foo-rt:compileNiagaraModuleTestJava
C:\Users\Developer\Niagara4.8\tridium\foo\foo-rt\srcTest\com\foo\FooTest.java:9: error: package org.mockito does not exist
import org.mockito.Mockito;
                  ^
1 error

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':foo-rt:compileNiagaraModuleTestJava'.
> Compilation failed; see the compiler error output for details.

* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED in 1s
2 actionable tasks: 2 executed
```

MITIGATION: For these unit tests to compile and run in Niagara 4.8, developers will need to add dependencies for any 3rd party libraries they use to their module gradle files.

For example, if the "foo-rt" module uses classes from the Mockito library, Mockito would need to be added as "testUberjar" dependency in foo-rt.gradle:

foo-rt.gradle

```

/* Module Build File */

description = "Foo Module"
ext {
}
niagaraModule {
    moduleName = "foo"
    preferredSymbol = "fb"
    runtimeProfile = "rt"
}

dependencies {
    compile "Tridium:nre:4.8"
    compile "Tridium:baja:4.8"
    testUberjar "org.mockito:mockito-all:1.10.19"
}

```

REMOVED LIBRARIES/ NEW DEPENDENCIES

The following table lists the libraries that were removed, the packages / classes that will be missing during the compile, and the gradle dependency that will need to be added to the module's gradle file.

Removed Library	Missing Packages/Classes During Compile	Required Dependency In Gradle File
AOP Alliance	org.aopalliance	testUberjar "aopalliance:aopalliance:1.0"
JCommander	com.beust	testUberjar "com.beust:jcommander:1.72"
PrivilegedAccessor	org.junit.extensions.PA	testUberjar "com.e-movimento.tinytools:privilegedaccessor:1.2.2"
Google Guava	com.google.guava	testUberjar "com.google.guava:guava:19.0"
Google Guice	com.google.inject	testUberjar "com.google.inject:guice:4.1.0"
Mockito	org.mockito	testUberjar "org.mockito:mockito-all:1.10.19"
Spring Framework	org.springframework	testUberjar "org.springframework:spring-test:5.0.7.RELEASE"
Javaax.inject API	javax.inject	testUberjar "javax.inject:javax.inject:1"
XML Unit	org.xmlunit	testUberjar "xmlunit:xmlunit:1.6"

BREAKING CHANGE: BACnet schedule Import and Export fails for Effective Periods with Wildcard

SUMMARY: Schedule Objects can no longer contain an Unspecified or Special Value. Most importantly for the Date Range type properties, the Schedule object should not contain an Unspecified or a Special Value.

DETAILS: Two important changes are as follows:

- BACnet controller profiles will not allow the Write Property service call to succeed when the properties of type DateRange of a Schedule Object is written with Special Date Time Values. The

Honeywell

Date values in the Date Range have to be a specific date value with actual day, month, year details available.

- A discovered BACnet Device will not be allowed to export a schedule which has Date Range properties like Effective Period having Special Date values (or Wild Card values)

WORKAROUND:

Ensure that the properties like Effective Period in the Schedule Object do not contain Special Date values. In Niagara, one can view the Schedule and update the Effective Period as follows:

Effective Period

◀ Prev Page ◀ Prev Month Today Next Month ▶ Next Page ▶

Oct 2018 Nov 2018 Dec 2018 Jan 2019 Feb 2019

Any Day Any Month Any Year [Calendar Icon]

Through

Any Day Any Month Any Year [Calendar Icon]

This Effective Period, when set, should have some specific date values as follows:

Scan Limit 090d 00h 00m [1day--inf]

Facets units=null,precision=1,min=-inf,max=+inf >> ⌚

Last Modified 18-Oct-2018 06:06 PM IST

Effective Date Range: 1 Nov 2018 - 5 Dec 2018

Always Effective false

Start Date: 1 Nov 2018

End Date: 5 Dec 2018

Out Source Default Output

Out - {null}

In - {null}

A Niagara station is not able to add a schedule if the schedule contains an Effective Period with Special Dates. In such a case, Niagara will show the schedules to be in Fault, as follows:

Discovered					5 objects		
Object Name	Object ID	Schedule Type	Data Type	Description			
BooleanSchedule	schedule:0	schedule:BooleanSchedule	BOOLEAN				
EnumSchedule	schedule:1	schedule:EnumSchedule	Unsigned				
NumericSchedule	schedule:2	schedule:NumericSchedule	REAL				
StringSchedule	schedule:3	schedule:StringSchedule	Character String				
CalendarSchedule	calendar:0	schedule:CalendarSchedule					

Database								5 objects
Name	Type	Object Id	State	Status	Last Success	Priority For Writing	Execution Time	
BooleanSchedule	Boolean Schedule	schedule:0	In Prog	{fault}	null	16	10secs {Sun Mon Tue Wed Thu Fri Sat}	
EnumSchedule	Enum Schedule	schedule:1	Idle	{fault}	null	16	1min {Sun Mon Tue Wed Thu Fri Sat}	
NumericSchedule	Numeric Schedule	schedule:2	Idle	{fault}	null	16	Manual	
StringSchedule	String Schedule	schedule:3	Idle	{fault}	null	16	1min {Sun Mon Tue Wed Thu Fri Sat}	
CalendarSchedule	Calendar Schedule	calendar:0	Idle	{ok}	26-Apr-18 11:41 PM EDT	16	1min {Sun Mon Tue Wed Thu Fri Sat}	

The logs in the console may show exceptions as follows:

```
SEVERE [04:36:07 26-Apr-18 EDT][bacnet.schedule] BacnetException reading supervisor schedule data for BooleanSchedule from schedul
ASN:Date contains Special Values.
at com.tridium.bacnet.schedule.ScheduleSupport0.checkForSpecialValuesInDateRange(ScheduleSupport0.java:178)
at com.tridium.bacnet.schedule.ScheduleSupport0.decodeDateRange(ScheduleSupport0.java:149)
at com.tridium.bacnet.schedule.BBacnetScheduleDeviceExt.readSchedule(BBacnetScheduleDeviceExt.java:739)
at com.tridium.bacnet.schedule.BBacnetScheduleDeviceExt.readRemote(BBacnetScheduleDeviceExt.java:383)
at com.tridium.bacnet.schedule.BBacnetScheduleImportExt.doExecute(BBacnetScheduleImportExt.java:170)
at auto.com_tridium_bacnet_schedule_BBacnetScheduleImportExt.invoke(AutoGenerated)
at com.tridium.sys.schema.ComponentSlotMap.invoke(Unknown Source)
at com.tridium.sys.engine.EngineUtil.doInvoke(Unknown Source)
at javax.baja.sys.BComponent.doInvoke(Unknown Source)
at javax.baja.util.Invocation.run(Unknown Source)
at javax.baja.util.Worker.process(Unknown Source)
at javax.baja.util.Worker$Processor.run(Unknown Source)
at java.lang.Thread.run(Thread.java:748)
```