ALARM LINK USER GUIDE

DSLINK-JAVA-ALARM

CONTENTS

September 8, 2017

OVERVIEW

This is a DSA link. It records details about interesting events so they may be considered by humans. The underlying model is influenced by the BACnet alarming design.

This link serves two purposes:

1. It is a framework upon which alarm links for different data stores can be developed.
2. It has three ready to use implementations, one of which, the default, can be deployed as-is.

USING THE PROVIDED IMPLEMENTATIONS

To use an implementation, one has the change the handler_class in dslink.json.

The provided implementations are:

- org.dsa.iot.alarm.jdbc.H2Main - Uses an embedded instance of the H2 database for persistence. This is default handler_class in dslink.json.
- org.dsa.iot.alarm.jdbc.JdbcMain - Uses remote JDBC connections for persistence. The JDBC driver jar file will need to be added to the deployment.
- org.dsa.iot.alarm.inMemory.Main - Uses Java collections for transient storage. This was developed primarily for testing purposes and may help with development of additional alarm links.

LINK STRUCTURE

The nodes of this link follow this hierarchy. Descriptions of each can be found in the Component Guide section of this document.

- *Alarm Service* – The single root node of the link.
    - *Alarm Class* – There can be many alarm classes in a link, each representing some sort of grouping criteria such as location, system or users.
        - *Alarm Algorithm* – There can be many algorithms per alarm class. Each algorithm has its own logic for determining when an alarm condition exists.
            - *Alarm Watch* – Path and meta-data about an entity being watched by the parent algorithm.

LINK USAGE

The purpose of this link is to create and manage alarms.

INITIAL SETUP

- Add an alarm class by invoking Add Alarm Class on the Alarm Service object.
- Add an alarm algorithm to the alarm class by invoking the Add Algorithm action.
    - Choose what type of algorithm in the action drop down. That will depend on the values you want to alarm on.
- Add an alarm watch to the new algorithm by invoking the Add Watch action.
    - To get a path, drop the value you wish to monitor onto any data flow, its path will be available on the resulting block.
- Manually create and manage an alarm.
    - Invoke Create Alarm on an alarm class.
        - The path is informational in this case and can be anything.
    - Invoke Get Open Alarms on the same alarm class.
        - Double-click to select the alarm UUID and copy it to your clipboard.
    - Acknowledge the alarm by invoking Acknowledge Alarm on the Alarm Service object.
        - Use the UUID you just copied.
        - Invoke Get Open Alarms again to see the update.
    - Return the alarm to normal by invoking Return to Normal on the Alarm Service object.
        - Use the same UUID as before.
        - The alarm will no longer be available in new invocations of Get Open Alarms.
        - Invoke Get Alarms instead to see the alarm.

CREATING ALARMS

- Action invocation – Alarm sources such as data flow logic and humans can invoke the Create Alarm action on any Alarm Class to create an alarm.
- Alarm algorithms – Algorithms, such as "out of range" can subscribe to paths, monitor their condition and automatically create alarms as well as return them to normal..

RECEIVING ALARMS

- Viewing – Alarm Service and Alarm Class objects have actions for retrieving tables of alarm records.
- Notifications – Alarm Service and Alarm Class objects also have actions for receiving streams of new alarms, state changes and escalations.

MANAGING ALARMS

- Acknowledgement – Fault and offnormal alarms require acknowledgement. Acknowledgement can be achieved with an action on the Alarm Service.
- Return to normal – All alarms must return to normal before they can be closed, this can be achieved with an action on the Alarm Service or by an alarm algorithm automatically detecting it.
- Closing – An alarm is considered closed when normal and acknowledged (unless it is an alert which does not require acknowledgement).

ALARM STATES

An alarm source is an entity that can be in an alarm condition. There are four possible states a source can be in.

ALERT

An alert is informational, it does not require acknowledgement. Once an alarm source in alert returns to normal, an operator would not see the alert on their console unless explicitly queried.

FAULT

Faults represent a malfunction or failure within the system. To close a fault, it must return to the normal state and be acknowledged.

NORMAL

Normal is healthy, and none of the other states.

OFFNORMAL

Offnormal represents an unexpected condition, or something outside the bounds of normal operation. To close an offnormal alarm, it must return to the normal state and be acknowledged.

OPEN VS CLOSED ALARMS

An alarm is considered open if it is not normal and/or if it required an acknowledgement and has not been acknowledged. A fault alarm that has returned to normal but is unacknowledged is considered open. It will be closed once the acknowledgement has occurred.

COMPONENT GUIDE

This section documents the major components of the link.

ALARM SERVICE

This is the visible root node of the link. Its purpose is to create alarm classes and manage alarm records independent of alarm class.

PROPERTIES

- Enabled - When false, no new alarms will be created.
- In Alarm Count – Number of alarms in the alarm state (not normal).
- Open Alarm Count – Number of open alarms.
- Total Alarm Count – Total not of alarms in any state.
- Unacked Alarm Count – Number of unacknowledged alarms.

ACTIONS

- Acknowledge – This set the acknowledge time and user on a specific alarm record. Has no effect if the record is already acknowledged.
    - Parameters
        - UUID – Required unique alarm ID.
        - User – Entity performing the update.
- Acknowledge All – Acknowledges all open records that are unacknowledged.
    - Parameters
        - User – Entity performing the update.
- Add Alarm Class – Add a new Alarm Class.
    - Parameters
        - Name – The alarm class name.
- Add Note – Add a note to an alarm record.
    - Parameters
        - UUID – Alarm record.
        - User – User name.
        - Note – Test message.
- Delete All Records – Deletes all records from the database.
- Delete Record – Deletes everything associated with the given UUID.
    - Parameters
        - UUID – Specific alarm id
- Get Alarm – Return a single row table representing the alarm record for the give UUID.
    - Parameters
        - UUID – Specific alarm id
    - Return – a single record. See the alarm record section of this document.
- Get Alarms – Returns a table of alarms for all alarm classes.
    - Parameters
        - Time Range – A DSA time range.

    - Returns – a table of alarm records.
        - See the alarm record section of this document.
- Get Notes – This returns a table of notes for a specific alarm.
    - Parameters
        - UUID – Specific alarm id
    - Return – a table with the following columns:
        - Timestamp – The time of the alarm.
        - User – The entity providing the note.
        - Note – The text of the note.
- Get Open Alarms – This returns a table of open alarms for all alarm classes. The table stream will remain open and any updates as well as new records will be passed along. The primary intent of this is for an alarm console.
    - Returns – a stream of table rows representing the alarm records. The stream state will switch to open once the initial set of open alarms is sent.
        - See the alarm record section of this document.
- Return To Normal – This sets the normal time of a specific alarm record. It has no effect on records that are already normal.
    - Parameters
        - UUID – The record to return to the normal state.

ALARM CLASS

An alarm class represents a group of alarms that are related in some way. Alarms can only be created with an alarm class but other alarm lifecycle operations such as acknowledgement are handled on the service. The alarm class offers many streams (as actions) for monitoring various states of alarms including escalation.

Escalation happens when alarm goes unacknowledged for a certain period of time and can be used to notify backup or higher seniority staff. There are two levels of escalation and their duration is calculated by combining the corresponding days, hours and minutes fields. Escalation happens when the duration from the alarm created time elapses. Escalation 2 is relative to escalation 1.

PROPERTIES

- Enabled – When false, no new alarms will be created.
- Purge Closed Days – If greater than zero, age in days after which closed alarms are deleted from the database.
- Purge Open Days – If greater than zero, age in days after which open alarms are deleted from the database.
- Escalation 1 Days – The number of days to add to the escalation duration
- Escalation 1 Hours – The number of hours to add to the escalation duration
- Escalation 1 Minutes – The number of minutes to add to the escalation duration
- Escalation 2 Days – The number of days to add to the escalation duration. Escalation 2 is relative to escalation 1.
- Escalation 2 Hours – The number of hours to add to the escalation duration. Escalation 2 is relative to escalation 1.
- Escalation 2 Minutes – The number of minutes to add to the escalation duration. Escalation 2 is relative to escalation 1.
- In Alarm Count – Number of alarms in the alarm state (not normal).
- Open Alarm Count – Number of open alarms.
- Total Alarm Count – Total not of alarms in any state.
- Unacked Alarm Count – Number of unacknowledged alarms.

ACTIONS

- Acknowledge All – Acknowledges all open records that are unacknowledged.
    - Parameters
        - User – Entity performing the update.
- Add Algorithm – Adds an algorithm for generating alarms.
    - Parameters
        - Name – The alarm class name.
        - Type – The specific alarm class desired.
- Create Alarm – Creates a new alarm record.
    - Parameters
        - Source Path – Path to the alarm source. This can actually be any text to describe the source but if the source can be accessed by a path, that path should be used for future proofing.

        - Create State – Alert, Fault or Offnormal
        - Message – Short text description.
    - Returns – a one row table representing the alarm record.
        - See the alarm record section of this document.
- Delete Alarm Class – Removes the alarm class and its child nodes.
- Get Alarms – Returns a table of alarm for this alarm class.
    - Parameters
        - Time Range – A DSA time range.
    - Returns – a table of alarm records.
        - See the alarm record section of this document.
- Get Open Alarms – This returns a table of open alarm alarms for this alarm class. The table stream will remain open and any updates as well as new records will be passed along. The primary intent of this is for an alarm console.
    - Returns – a stream of table rows representing the alarm records. The stream state will switch to open once the initial set of open alarms is sent.
        - See the alarm record section of this document.
- Stream Escalation 1 – Returns a stream of alarm records as they escalate in real time.
    - Returns – table rows representing the alarm records.
        - See the alarm record section of this document.
- Stream Escalation 2 – Returns a stream of alarm records as they escalate in real time.
    - Returns – table rows representing the alarm records.
        - See the alarm record section of this document.
- Stream New Alarms– This returns a stream of new alarm records for this alarm class. The table stream will remain open and any updates and new records will be passed along.
    - Returns – table rows representing the alarm records.
        - See the alarm record section of this document.

ALARM ALGORITHMS

Alarm algorithms evaluate the state of Alarm Watch objects and automatically generate alarms when the conditions of the algorithm are met. This describes the common functionality of all alarm algorithms, individual algorithms will be described in a separately.

PROPERTIES

- Enabled – When false, no new records will be created.
- Alarm Type – What type of alarm this algorithm creates: alert, fault or offnormal.
- Auto Update Interval – If greater than zero, will automatically re-evaluate the alarm state of each watch on this interval. An auto interval should be used if using inhibits. Watches will always update themselves they detect a change of value on the source.
- To Alarm Inhibit – How long (in seconds) to delay going into alarm after the alarm condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- To Normal Inhibit – How long (in seconds) to delay a return to normal after the normal condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.

ACTIONS

- Add Watch – Takes path for subscription in the parent broker.
- Delete Algorithm – Remove the algorithm from the parent alarm class.
- Update All – Has all watches re-evaluate their state.

ALARM WATCH

Represents an alarm source that an algorithm will monitor for alarm conditions. There is a primary alarm source, but other paths may be used by subclasses for determining more complex conditions.

PROPERTIES

- Enabled – When false, no new records will be created.
- Source Path – The path to the primary alarmable entity.
- Alarm State – The current state of the source.
- Alarm State Time – The best known time that the source transitioned to the alarm state.
- Last Alarm Record www– The UUID of the last related alarm record.
- Last Cov – The timestamp of the last change of value of the source.

ACTIONS

- Delete Watch – Remove the watch from the parent algorithm.

ALARM RECORD

An alarm record represents details about an alarm. This is an abstract description of the Java class as well table columns in the DSA protocol.

PROPERTIES

- UUID – Unique ID, generated by the link.
- Source – Path to the alarm source.
- Alarm_Class – The name of the alarm class the record was created in.
- Created_Time – Timestamp of creation.
- Created_State – The state of the source at creation. Possible values are:
    - Alert – Informational, acknowledge not required.
    - Fault – A malfunction representing a failure within the system.
    - Offnormal – An unexpected condition, or outside the bounds of normal operation.
- Normal_Time – If not null, the timestamp that the source returned to normal.
- Ack_Time – If not null, the timestamp of acknowledgement.
- Ack_User – The entity that acknowledged the alarm.
- Message – Text describing the alarm at the time of creation.
- Has_Notes – Whether or not the alarm has any notes associated with it.
- Watch_Path – The path of the algorithm watch that created the alarm, or null if the alarm was created another way.
- Is_Normal – Boolean indicating whether or not the alarm source returned to normal.
- Is_Acknowledged – Boolean indicated whether or not the alarm has been acked.

BOOLEAN ALGORITHM

This algorithm creates alarms when boolean sources turn true or false.

PROPERTIES

- Enabled – When false, no new records will be created.
- Alarm Type – What type of alarm this algorithm creates: alert, fault or offnormal.
- Auto Update Interval – If greater than zero, will automatically re-evaluate the alarm state of each watch on this interval. Will always update when the watch detects a change of value on the source.
- To Alarm Inhibit – How long (in seconds) to delay going into alarm after the alarm condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- To Normal Inhibit – How long (in seconds) to delay a return to normal after the normal condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- Alarm Value – What to alarm on, true or false.
- Message – The pattern used to generate the alarm message. Use %s to have the value at the time of alarm creation inserted into the message.

ACTIONS

- Add Watch – Takes path for subscription in the parent broker.
- Delete Algorithm – Remove the algorithm from the parent alarm class.
- Update All – Re-evaluate all child watches.

OUT OF RANGE ALGORITHM

This algorithm creates alarms for sources whose numeric value is less than a minimum value, or greater than a maximum value.

PROPERTIES

- Enabled – When false, no new records will be created.
- Alarm Type – Enum indicating whether records should be alert, fault or offnormal.
- Auto Update Interval – If greater than zero, will automatically re-evaluate the alarm state of each watch on this interval. Will always update when the watch detects a change of value on the source.
- To Alarm Inhibit – How long (in seconds) to delay going into alarm after the alarm condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- To Normal Inhibit – How long (in seconds) to delay a return to normal after the normal condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- Min Value – Value to use if Use Node Range is false, or the target node does not define a min value.
- Max Value – Value to use if Use Node Range is false, or the target node does not define a max value.

- Message – The pattern used to generate the alarm message. Use %s to have the value at the time of alarm creation inserted into the message.

ACTIONS

- Add Watch – Takes path for subscription in the parent broker.
- Delete Algorithm – Remove the algorithm from the parent alarm class.
- Update All – Re-evaluate all child watches.

REMOTE JDBC SERVICE

This is an alarm service with a connection to a remote database. All properties and actions are inherited from the base Alarm Service.

PROPERTIES

The following are unique to this type.

- JDBC Driver – Class name of the driver.
- Database URL – Enum indicating whether records should be alert, fault or offnormal.
- Database Name – The database will be created if it does not already exist.
- Database User – Credentials to access the database base. If blank, will only attempt to acquire a connection using the URL.
- Database Pass – Password for the database user.

STALE ALGORITHM

This algorithm creates alarms for sources whose value does not change after a certain period of time. This can be useful for detecting sensor failure.

PROPERTIES

- Enabled – When false, no new records will be created.
- Alarm Type – What type of alarm this algorithm creates: alert, fault or offnormal.
- Auto Update Interval – If greater than zero, will automatically re-evaluate the alarm state of each watch on this interval. Will always update when the watch detects a change of value on the source.
- To Alarm Inhibit – How long (in seconds) to delay going into alarm after the alarm condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- To Normal Inhibit – How long (in seconds) to delay a return to normal after the normal condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- Stale Days – The number of days to add to the stale duration.
- Stale Hours – The number of hours to add to the stale duration.
- Stale Minutes – The number of minutes to add to the stale duration.
- Message – The pattern used to generate the alarm message. Use %s to have the value at the time of alarm creation inserted into the message.

ACTIONS

- Add Watch – Takes path for subscription in the parent broker.
- Delete Algorithm – Remove the algorithm from the parent alarm class.
- Update All – Re-evaluate all child watches.

STRING ALGORITHM

This algorithm creates alarms based on Strings. Use the Value Mode property to determine how the source value is compared to the alarm value defined on this object. The source can be any value type, it will be converted to a string.

PROPERTIES

- Enabled – When false, no new records will be created.
- Alarm Type – What type of alarm this algorithm creates: alert, fault or offnormal.
- Auto Update Interval – If greater than zero, will automatically re-evaluate the alarm state of each watch on this interval. Will always update when the watch detects a change of value on the source.
- To Alarm Inhibit – How long (in seconds) to delay going into alarm after the alarm condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- To Normal Inhibit – How long (in seconds) to delay a return to normal after the normal condition is first detected. This can help minimize alarm creation. Use zero to disable, otherwise you should have a positive auto update interval.
- Alarm Value – The string to the source value is compared against.
- Value Mode – How to compare the source value to the alarm value. For example, if the mode is EndsWith, the evaluation is in alarm = source value ends with alarm value.
- Message – The pattern used to generate the alarm message. Use %s to have the value at the time of alarm creation inserted into the message.

ACTIONS

- Add Watch – Takes path for subscription in the parent broker.
- Delete Algorithm – Remove the algorithm from the parent alarm class.
- Update All – Re-evaluate all child watches.

CREATING A CUSTOM ALARM LINK

Creating a custom link primarily requires implementing a single interface. After that, plenty of hooks exist if customizations of other built in types is required.

1. Create an implementation of org.iot.dsa.alarm.Alarming.Provider.
2. Create a "main" class that subclasses org.iot.dsa.alarm.AlarmLinkHandler.
3. In the main Main class:
    a. In a static initializer, call Alarming.setProvider with an instance of your provider.
    b. In the main method, call DSLinkFactory.start with an instance of the main class.For example:

```
static {
  Alarming.setProvider(new MyProvider());
}

public static void main(String[] args) {
  DSLinkFactory.start(args, new MyMainClass());
}
```

In the dslink-java-alarm module there are some providers that can be used for reference:

1. org.dsa.iot.alarm.inMemory.Main
2. org.dsa.iot.alarm.jdbc.JdbcMain
3. ord.dsa.iot.alarm.jdbc.H2Main