NETLINX PROGRAMMER'S MANUAL

RMS ENTERPRISE SCHEDULING API

RMS-ENT (V4.6 OR HIGHER)

AV FOR AN IT WORLD

COPYRIGHT NOTICE

LIABILITY NOTICE

AMX WARRANTY AND RETURN POLICY

Table of Contents

RMS Enterprise Scheduling API

Overview

This is the RMS Enterprise Scheduling API Guide. It is intended for developers who are needing to integrate a scheduling system with RMS Enterprise. This document assumes familiarity with XML, HTTP, REST web services, and some programming language (Java, C#, Ruby, etc.).

RMS Enterprise Scheduling API Requirements

The web services described in this guide are implemented in RMS Enterprise Server 4.3 or newer. No language bindings for client development are provided. Developers can use their language of choice to consume these REST web service APIs.

Authentication

RMS Enterprise requires *Digest Authentication* for all REST API requests. Please see https://www.ietf.org/rfc/rfc2617.txt for a detailed specification. RMS Server includes a user account intended for use with this Scheduling API. The user name for this account is *scheduler*. It is associated with a role named *Scheduler*. The Scheduler role comes with the necessary permissions for interacting with all of the APIs detailed in this document, and it cannot be changed. These permissions include: Manage Clients and Locations

The password for the scheduler user account is set to "*password*" by default and can be changed from the RMS UI.

Request Headers

```
content-type = application/xml
accept = application/xml
```

Configuration Workflow

Before any appointment synchronization can occur, some configuration steps must first be accomplished.

1. Test connectivity and authentication to the RMS Server (see *Test Authentication* on page 8).
2. Create a Troller record on the RMS Server (see *Save Troller* on page 9).
3. Query the scheduling system for all the available resources.
4. Save a collection of Resource Profiles to the RMS Server for all resources that need to be synchronized (see *Save Resource Profiles for Troller* on page 11).

Synchronization Workflow

The *Synchronization Workflow* should take place on a periodic basis (e.g. every 15 minutes). An initial synchronization can be lengthy depending upon how many resources are being synchronized and how many appointments are on the calendars of those resources.

1. Test connectivity to the scheduling system.
2. If connectivity to the scheduling system fails, report that failure to the RMS Server (see *Report a Scheduling Error* on page 14).
3. Loop over all Resource Profiles that need to be synchronized.
4. Query the scheduling system for all the appointment changes for the resource associated to the Resource Profile being processed since the last troll/sync cycle.
5. Construct Booking, Event, BookingAuxiliary, and Attendee objects and save them to the RMS Server for new/modified appointments (see *Save Bookings for a Resource Profile* on page 15).
6. Issue calls to the RMS Server to delete appointments that have been removed/canceled from the scheduling system (see *Delete Bookings* on page 18).
7. For each Resource Profile, issue a sync complete message to the RMS Server once all appointment changes from the scheduling system have been processed (see *Report a Completed Synchronization* on page 19).
   If any of those appointment changes apply to "today", set the today flag to true. If any Resource Profile fails to synchronize, that should be reported as a failure to the RMS Server and will result in a Hotlist record (see *Report a Failed Synchronization* on page 19).

Heartbeat Workflow

The *Heartbeat Workflow* should also take place on a periodic basis but much more often than the Synchronization Workflow. For example, the RMS Server does not care how often a Troller performs the Synchronization Workflow. However, the RMS Server does need to receive a heartbeat from the Troller at least once every two minutes. Otherwise, the RMS Server will mark the Troller as offline, create a Hotlist record, and emit notifications. It is recommended that the Heartbeat Workflow occur every five or ten seconds. The longer the delay between Heartbeats, the slower the response time will be for AdHoc Messages.

1. Wake up and retrieve Troller Messages from the RMS Server (this does not remove the messages from the RMS Server - see *Get All Troller Messages* on page 13).
2. Process the messages one by one and in the order they are given. Processing them out of order will lead to erroneous results.
3. Once a message is processed by the Troller, delete the Troller Message off of the RMS Server (see *Delete One or More Troller Messages* on page 14).
4. When all Troller Messages have been processed, go back to sleep.

RMS Server Scheduling APIs

Test Authentication

This URI requires an authenticated user and is therefore useful for verifying that the configured credentials are valid with the RMS Server. This API is only used during the configuration phase.

Request

HTTP Method	`GET`
URI Template	`/api/v2/server/setting/application.title`
Example	`http://acme.com/rms/api/v2/server`

Response Body

The response body can be discarded for the purposes of testing authentication.

Response Codes

HTTP Code	Meaning
`200 (OK)`	The request completed successfully and the user is authenticated.
`401 (Unauthorized)`	The credentials were not accepted.

Get Server Information

The client uses this method for obtaining server configuration properties and server environment information. For the purposes of scheduling, it is important to verify the RMS Server is licensed for scheduling (see *schedulingLicensed* element). This URI does not require authentication.

Request

HTTP Method	`GET`
URI Template	`/api/v2/server`
Example	`http://acme.com/rms/api/v2/server`

Response

```
<serverInfo>
    <applicationVersion>4.3.10</applicationVersion>
    <databaseVersion>4.3.2</databaseVersion>
    <rfidEnabled>false</rfidEnabled>
    <smtpEnabled>true</smtpEnabled>
    <timesyncEnabled>true</timesyncEnabled>
    <minPollTime>15</minPollTime>
    <maxPollTime>96</maxPollTime>
    <applicationTimeZone>America/Chicago</applicationTimeZone>
    <offset>-18000000</offset>
    <trial>false</trial>
    <serverLicensed>true</serverLicensed>
    <assetLicensed>true<assetLicensed>
    <assetUnits>100</assetUnits>
    <schedulingLicensed>true<schedulingLicensed>
</serverInfo>
```

Response Codes

HTTP Code	Meaning
`200 (OK)`	The request completed successfully and the user is authenticated.
`401 (Unauthorized)`	The credentials were not found.

Save Troller

This API is used to create, or update, an instance of this Troller on the RMS Server. This API is only used during the configuration phase.

Request

HTTP Method	`PUT`
URI Template	`/api/v2/trollers/{name}`
Example	`http://acme.com/rms/api/v2/trollers/myTroller`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Request Body

```
<troller>
    <name>myTroller</name>
</troller>
```

Response Body

```
<troller>
    <id>1</id>
    <version>0</version>
    <name>myTroller</name>
</troller>
```

The RMS Server creates the *id* and *version* elements.

Response Codes

HTTP Code	Meaning
`200 (OK)`	The Troller instance was updated on the RMS Server.
`201 (Created)`	The Troller instance was created on the RMS Server.

Get Troller

This API is used to retrieve an instance of a Troller from the RMS Server. This API is not typically used for scheduling purposes. It is included here only for completeness.

Request

HTTP Method	`GET`
URI Template	`/api/v2/trollers/{name}`
Example	`http://acme.com/rms/api/v2/trollers/myTroller`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Response Body

```
<troller>
    <id>1</id>
    <version>0</version>
    <name>myTroller</name>
</troller>
```

Response Codes

HTTP Code	Meaning
`200 (OK)`	The Troller instance on the RMS Server was returned.
`404 (Not Found)`	The Troller instance does not exist on the RMS Server.

Delete Troller

This API is used to delete the Troller and cascade delete all of its Resource Profiles, Events, Bookings, and BookingAuxiliary records. This API is not used during normal appointment synchronization operation but might have value when decommissioning a scheduling system with RMS.

Request

HTTP Method	`DELETE`
URI Template	`/api/v2/trollers/{name}`
Example	`http://acme.com/rms/api/v2/trollers/myTroller`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Response Codes

HTTP Code	Meaning
`204 (No Content)`	The Troller instance does not exist on the RMS Server.

Get All Resource Profiles for Troller

This API is for retrieving all the resource profiles from the RMS Server for the given Troller. This API is only used during the configuration phase.

Request

HTTP Method	`GET`
URI Template	`/api/v2/trollers/{name}/resources`
Example	`http://acme.com/rms/api/v2/trollers/myTroller/resources`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Response Body

```
<resourceProfiles>
    <resourceProfile>
        <id>1</id>
        <version>0</version>
        <friendlyName>Room 101</friendlyName>
        <externalId>conf-room-101</externalId>
        <hashedExternalId>aeb88545d5db671026effcfdbfb</hashedExternalId>
        <location>-1</location>
    </resourceProfile>
    <resourceProfile>
        <id>2</id>
        <version>0</version>
        <friendlyName>Room 501</friendlyName>
        <externalId>conf-room-501</externalId>
        <hashedExternalId>0bb157a31132ced6c7a9b0e75f0</hashedExternalId>
        <location>-1</location>
    </resourceProfile>
</resourceProfiles>
```

If the *location* element equals "-1", this means that Resource Profile has not been mapped to a Location yet and thus no appointments will be synchronized for this particular Resource Profile. The RMS Server creates the *id* and *version* elements.

Save Resource Profiles for Troller

This API is for creating or updating one or more Resource Profiles on the RMS Server for the given Troller. This API is only used during the configuration phase. The resources are retrieved from the scheduling system and then stored into the RMS Server using this API.

Request

HTTP Method	`POST`
URI Template	`/api/v2/trollers/{name}/resources`
Example	`http://acme.com/rms/api/v2/trollers/myTroller/resources`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Request Body

```
<resourceProfiles>
    <resourceProfile>
        <friendlyName>Room 101</friendlyName>
        <externalId>conf-room-101</externalId>
        <hashedExternalId>aeb88545d5db671026effcfdbfb</hashedExternalId>
    </resourceProfile>
    <resourceProfile>
        <friendlyName>Room 501</friendlyName>
        <externalId>conf-room-501</externalId>
        <hashedExternalId>0bb157a31132ced6c7a9b0e75f0</hashedExternalId>
    </resourceProfile>
</resourceProfiles>
```

The *externalId* is the primary key of the resource from the scheduling system. The *hashedExternalId* is a SHA-256 hash of the *externalId* and is used as a request path variable in some APIs. The reason for hashing is because certain scheduling systems use characters in their external IDs that are not compatible with URLs.

Response Body

```
<resourceProfiles>
    <resourceProfile>
        <id>1</id>
        <version>0</version>
        <friendlyName>Room 101</friendlyName>
        <externalId>conf-room-101</externalId>
        <hashedExternalId>aeb88545d5db671026effcfdbfb</hashedExternalId>
        <location>-1</location>
    </resourceProfile>
    <resourceProfile>
        <id>2</id>
        <version>0</version>
        <friendlyName>Room 501</friendlyName>
        <externalId>conf-room-501</externalId>
        <hashedExternalId>0bb157a31132ced6c7a9b0e75f0</hashedExternalId>
        <location>-1</location>
    </resourceProfile>
</resourceProfiles>
```

It is incumbent upon the scheduling plugin developer to locally persist these Resource Profile IDs as they are necessary for other APIs. This ID is the primary key in RMS. In addition, the local storage must also retain knowledge of whether or not the Resource Profile is mapped to a Location.

If no Resource Profiles exist for the given Troller, the response will be:

```
<resourceProfiles>
</resourceProfiles>
```

Response Codes

HTTP Code	Meaning
`200 (OK)`	The Resource Profile(s) was updated on the RMS Server.
`201 (Created)`	At least one Resource Profile was created on the RMS Server.
`404 (Not Found)`	The Troller instance does not exist.

Save Single Resource Profile for Troller

This API is for creating or updating one Resource Profile on the RMS Server for the given Troller.

This API is only used during the configuration phase.

Request

HTTP Method	PUT
URI Template	/api/v2/trollers/{name}/resources/ext/{extId}
Example	http://acme.com/rms/api/v2/trollers/myTroller/resources/ext/123

Request Path Variables

Parameter	Data Type	Description
name	String	This is the Troller's name and must be unique.
extId	String	The hashed external ID of the Resource Profile

Request Body

```
<resourceProfile>
    <friendlyName>Room 101</friendlyName>
    <externalId>conf-room-101</externalId>
    <hashedExternalId>aeb88545d5db671026effcfdbfb</hashedExternalId>
</resourceProfile>
```

Response Body

```
<resourceProfile>
    <id>1</id>
    <version>0</version>
    <friendlyName>Room 101</friendlyName>
    <externalId>conf-room-101</externalId>
    <hashedExternalId>aeb88545d5db671026effcfdbfb</hashedExternalId>
    <location>-1</location>
</resourceProfile>
```

Response Codes

HTTP Code	Meaning
200 (OK)	The Resource Profile was updated on the RMS Server.
201 (Created)	The Resource Profile was created on the RMS Server.
404 (Not Found)	The Troller instance does not exist.

Delete Resource Profiles for Troller

This API is for deleting one or more Resource Profiles on the RMS Server for the given Troller. This operation will also cascade delete to the Events, Bookings, and BookingAuxiliary records for the Resource Profiles. This API is only used during the configuration phase. It is intended when reconciling the Resource Profiles between the scheduling system and the RMS Server. When a resource is deleted from the scheduling system, this API can be used to remove it from the RMS Server.

Request

HTTP Method	DELETE
URI Template	/api/v2/trollers/{name}/resources/ext/{extIds}
Example	http://acme.com/rms/api/v2/trollers/myTroller/resources/ext/123,ABC

Request Path Variables

Parameter	Data Type	Description
name	String	This is the Troller's name and must be unique.
extId	String	One or more hashed external ID of the Resource Profile delimited by commas

Response Codes

HTTP Code	Meaning
204 (No Content)	The Resource Profile(s) don't exist on the RMS Server.

Delete All Resource Profiles for Troller

This API is for deleting all of the Resource Profiles for the given Troller on the RMS Server. This operation will also cascade delete to the Events, Bookings, and BookingAuxiliary records for all the Resource Profiles. This API is not typically used but might be of value in some circumstances.

Request

HTTP Method	`DELETE`
URI Template	`/api/v2/trollers/{name}/resources/all`
Example	`http://acme.com/rms/api/v2/trollers/myTroller/resources/all`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Response Codes

HTTP Code	Meaning
`204 (No Content)`	The Resource Profile(s) don't exist on the RMS Server.

Get All Troller Messages

This API is used in two ways:

1. Perform a heartbeat to the RMS Server. If the Troller doesn't heartbeat at least once every two minutes, the RMS Server will mark the Troller as *Not Communicating* on the hotlist and issue a notification email. Typically this heartbeat is performed every five seconds.
2. Retrieve all the Troller's messages. These messages are instructions to the Troller from the RMS Server.

Getting these messages does not delete them from the RMS Server's database. That is a separate REST API to delete a particular Troller message.

Request

HTTP Method	`GET`
URI Template	`/api/v2/trollers/{name}/messages`
Example	`http://acme.com/rms/api/v2/trollers/myTroller/messages`

Request Path Variables

Parameter	Data Type	Description
`name`	String	This is the Troller's name and must be unique.

Response Body

```
<trollerMessages>
    <trollerMessage>
        <id>4</id>
        <version>0</version>
        <command>resource_profile_mapped</command>
        <message>conf-room-101</message>
    </trollerMessage>
    <trollerMessage>
        <id>5</id>
        <version>0</version>
        <command>resource_profile_unmapped</command>
        <message>class-room-500</message>
    </trollerMessage>
</trollerMessages>
```

If no messages exist for the given Troller, the response body will be:

```
<trollerMessages>
</trollerMessages>
```

Response Codes

HTTP Code	Meaning
`200 (OK)`	The Troller was found and any messages returned.
`404 (Not Found)`	The Troller does not exist on the RMS Server.

Delete One or More Troller Messages

This API is used to delete one or more Troller messages from the RMS Server once the Troller has successfully processed the message(s).

Request

HTTP Method	DELETE
URI Template	/api/v2/trollers/{name}/messages/{messageIds}
Example	http://acme.com/rms/api/v2/trollers/myTroller/messages/4,5

Request Path Variables

Parameter	Data Type	Description
name	String	This is the Troller's name and must be unique.
messageIds	String	One or more Troller message IDs delimited by commas

Response Codes

HTTP Code	Meaning
204 (No Content)	The Troller message(s) don't exist on the RMS Server.

Report a Scheduling Error

This API is used to report a scheduling error for the given Troller to the RMS Server. Examples of typical problems would be loss of network connectivity to the scheduling system or authentication failure with that scheduling system (i.e. the credentials changed on the scheduling system and no one updated the Troller's configuration). This scheduling error will show up on the RMS hotlist and result in email notifications if the appropriate SMTP and notification rule configurations are in place on the RMS Server.

Request

HTTP Method	PUT
URI Template	/api/v2/trollers/{name}/error
Example	http://acme.com/rms/api/v2/trollers/myTroller/error

Request Path Variables

Parameter	Data Type	Description
name	String	This is the Troller's name and must be unique.

NOTE: *There is no request body or response body for this API.*

Response Codes

HTTP Code	Meaning
200 (OK)	The scheduling error was successfully reported for this Troller.
404 (Not Found)	The Troller does not exist on the RMS Server.

Clear a Scheduling Error

This API is used to clear a scheduling error for the given Troller previously reported to the RMS Server. This API will be called when connectivity is restored and a successful synchronization operation has occurred.

Request

HTTP Method	DELETE
URI Template	/api/v2/trollers/{name}/error
Example	http://acme.com/rms/api/v2/trollers/myTroller/error

Request Path Variables

Parameter	Data Type	Description
name	String	This is the Troller's name and must be unique.

NOTE: *There is no request body or response body for this API.*

Response Codes

HTTP Code	Meaning
204 (No Content)	The scheduling error was successfully cleared for this Troller.

Save Bookings for a Resource Profile

This API is used to create or update one or more Bookings on the RMS Server for the given Resource Profile. Single appointment Bookings must have an Event and a BookingAuxiliary defined.

Recurring appointments will have a Booking and BookingAuxiliary for each occurrence in the series and a single Event whose data is repeated inside of each Booking.

Request

HTTP Method	`POST`
URI Template	`/api/v2/resources/{id}/bookings`
Example	`http://acme.com/rms/api/v2/resources/27/bookings`

Request Path Variables

Parameter	Data Type	Description
`id`	Long	This is the Resource Profile's primary key in RMS.

Request Body

```
<bookings>
  <booking>
    <event>
      <organizer>
        <friendlyName>George Washington</friendlyName>
        <externalId>attendee-1</externalId>
      </organizer>
      <externalEventId>6GpEWGn2fsyJ6sbwAAAP9LqgAAEA==</externalEventId>
      <hashedExternalEventId>012f5f8fd93a17f9b5d25e261354...</hashedExternalEventId>
      <subject>My Meeting Subject</subject>
      <details>The is the meeting's body</details>
      <allDayEvent>false</allDayEvent>
      <privateEvent>false</privateEvent>
      <attendees>
        <attendee>
          <friendlyName>George Washington</friendlyName>
          <externalId>attendee-1</externalId>
        </attendee>
        <attendee>
          <friendlyName>Thomas Jefferson</friendlyName>
          <externalId>attendee-2</externalId>
        </attendee>
      </attendees>
    </event>
    <externalBookingId>QAAy0DFO16GpEWGn2fsyJ6sbwAAAP9LqgAAEA==</externalBookingId>
    <hashedExternalBookingId>65acb2834191bd8552e49b...</hashedExternalBookingId>
    <singleEvent>true</singleEvent>
    <startDateTimeMillis>1336928400000</startDateTimeMillis>
    <endDateTimeMillis>1336932000000</endDateTimeMillis>
    <bookingAuxiliary>
      <lastTrollMillis>1338905117467</lastTrollMillis>
    </bookingAuxiliary>
  </booking>
</bookings>
```

Response Body

```
<bookings>
  <booking>
    <id>27</id>
    <version>0</version>
    <event>
      <id>13</id>
      <version>0</version>
      <organizer>
        <id>1</id>
        <version>0</version>
        <friendlyName>George Washington</friendlyName>
        <externalId>attendee-1</externalId>
      </organizer>
      <externalEventId>6GpEWGn2fsyJ6sbwAAAP9LqgAAEA==</externalEventId>
      <hashedExternalEventId>012f5f8fd93a17f9b5d25e261354...</hashedExternalEventId>
      <subject>My Meeting Subject</subject>
      <details>The is the meeting's body</details>
      <allDayEvent>false</allDayEvent>
      <privateEvent>false</privateEvent>
      <attendees>
        <attendee>
          <id>1</id>
          <version>0</version>
          <friendlyName>George Washington</friendlyName>
          <externalId>attendee-1</externalId>
        </attendee>
        <attendee>
          <id>3</id>
          <version>0</version>
          <friendlyName>Thomas Jefferson</friendlyName>
          <externalId>attendee-3</externalId>
        </attendee>
      </attendees>
    </event>
    <externalBookingId>QAAy0DFO16GpEWGn2fsyJ6sbwAAAP9LqgAAEA==</externalBookingId>
    <hashedExternalBookingId>65acb2834191bd8552e49b...</hashedExternalBookingId>
    <singleEvent>true</singleEvent>
    <startDateTimeMillis>1336928400000</startDateTimeMillis>
    <endDateTimeMillis>1336932000000</endDateTimeMillis>
    <bookingAuxiliary>
      <id>27</id>
      <version>0</version>
      <lastTrollMillis>1338905117467</lastTrollMillis>
    </bookingAuxiliary>
  </booking>
</bookings>
```

Response Codes

HTTP Code	Meaning
200 (OK)	One or more Booking(s) were updated for the Resource Profile
201 (Created)	One or more Booking(s) were created for the Resource Profile
404 (Not Found)	The Resource Profile does not exist

Get Bookings for a Resource Profile and Date Range

This API is used to retrieve Bookings between a particular date range from the RMS Server for the given Resource Profile. This API is included for completeness but is not necessary for a scheduling implementation.

Request

HTTP Method	`GET`
URI Template	`/api/v2/resources/{id}/bookings?startDateTime={start}&endDateTime={end}`
Example	`http://acme.com/rms/api/v2/resources/27/bookings?startDateTime=1336928400000&endDateTime=1336932000000`

Request Path Variables

Parameter	Data Type	Description
`id`	Long	This is the Resource Profile's primary key in RMS.

Request Parameters

Parameter	Data Type	Description
`startDateTime`	Long	This is the UTC start date and time in milliseconds.
`endDateTime`	Long	This is the UTC end date and time in milliseconds.

Request Body

There is no request body for this API.

Response Body

The response body is the same as for the *Save Bookings for a Resource Profile* API (see page 15). If no bookings were found the response will be:

```
<bookings>
</bookings>
```

Response Codes

HTTP Code	Meaning
`200 (OK)`	One or more Booking(s) were retrieved for the Resource Profile.
`404 (Not Found)`	The Resource Profile does not exist

Delete Bookings for a Resource Profile

This API will delete all of the Bookings (and associated BookingAuxiliarys and Events) for the given Resource Profile.

Request

HTTP Method	`DELETE`
URI Template	`/api/v2/resources/{id}/bookings`
Example	`http://acme.com/rms/api/v2/resources/27/bookings`

Request Path Variables

Parameter	Data Type	Description
`id`	Long	This is the Resource Profile's primary key in RMS.

Request Body

There is no request body for this API.

Response Body

```
<bookingCount>
  <count>5</count>
</bookingCount>
```

The response provides a count of how many bookings were deleted in that operation.

Response Codes

HTTP Code	Meaning
`204 (No Content)`	The Bookings were deleted or didn't exist.
`404 (Not Found)`	The Resource Profile does not exist

Delete Bookings

This API will delete all the given Bookings (and associated BookingAuxiliarys and Events). The Bookings don't have to all be for the same Resource Profile because each has its own hashed external Booking ID to identify it. The RMS Server will not delete an Event if there are still Bookings that remain. For example, if this API is used to only delete a portion of a the Bookings from a series.

Request

HTTP Method	`DELETE`
URI Template	`/api/v2/bookings`
Example	`http://acme.com/rms/api/v2/bookings`

Request Body

The request body is the same as the *Save Bookings for Resource Profile* API (see page 15).

Response Body

There is no response body for this API.

Response Codes

HTTP Code	Meaning
`204 (No Content)`	The Bookings were deleted or didn't exist.

Delete a Single Booking

This API will delete a single Booking (and associated BookingAuxiliary and Event) from RMS.

Request

HTTP Method	`DELETE`
URI Template	`/api/v2/bookings/ext/{hashedExtId}`
Example	`http://acme.com/rms/api/v2/bookings/ext/02A375BA`

Request Path Variables

Parameter	Data Type	Description
`hashedExtId`	String	This is the hashed (SHA-256) external ID in hex format from the scheduling system for the Booking.

Request Body

There is no request body for this API.

Response Body

There is no response body for this API.

Response Codes

HTTP Code	Meaning
`204 (No Content)`	The Booking were deleted or didn't exist.

Report a Completed Synchronization

This API is used to report to the RMS Server that a synchronization cycle of a Resource Profile has been completed successfully. Receipt of this message causes the RMS Server to instruct the RMS Client to retrieve its Bookings from the RMS Server. If there are any hotlist items in RMS for this Resource Profile or for the Troller, they will be cleared as a result of this message.

Request

HTTP Method	`PUT`
URI Template	`/api/v2/resources/{id}/synchronized?today={flag}`
Example	`http://acme.com/rms/api/v2/resources/27/synchronized?today=true`

Request Path Variables

Parameter	Data Type	Description
`id`	Long	This is the Resource Profile's primary key in RMS.

Request Parameters

Parameter	Data Type	Description
`today`	Boolean	The today flag should be set to true if there was a least one new or updated booking that was pushed to the RMS Server during this last synchronization cycle that happens on the current calendar day in the time zone of the location associated with this Resource Profile. Otherwise, it should be set to false. This is an optional parameter. If not given, a default of *false* will be used.

If the *today* flag is false, it saves the RMS Client from having to make call to retrieve today's meetings when nothing changed.

NOTE: *There is no request or response body for this API.*

Response Codes

HTTP Code	Meaning
`200 (OK)`	The synchronization completion was successfully reported to the RMS Server.
`404 (Not Found)`	The Resource Profile doesn't exist.

Report a Failed Synchronization

This API is used to report a synchronization failure to the RMS Server for the given Resource Profile. Receipt of this message causes the RMS Server to place an item on the Hotlist stating that this Resource Profile failed to synchronize. Configured RMS notification rules will be executed as usual.

Request

HTTP Method	`PUT`
URI Template	`/api/v2/resources/{id}/failure`
Example	`http://acme.com/rms/api/v2/resources/27/failure`

Request Path Variables

Parameter	Data Type	Description
`id`	Long	This is the Resource Profile's primary key in RMS.

NOTE: *There is no request or response body for this API.*

Response Codes

HTTP Code	Meaning
`200 (OK)`	The synchronization completion was successfully reported to the RMS Server.
`404 (Not Found)`	The Resource Profile doesn't exist.
`409 (Conflict)`	The Resource Profile isn't mapped to a location.

Submit a Response to an Adhoc Request

This API is used to submit a response to the RMS Server for an adhoc request from the given Resource Profile.

Request

HTTP Method	PUT
URI Template	/api/v2/resources/{id}/failure
Example	http://acme.com/rms/api/v2/resources/27/failure

Request Path Variables

Parameter	Data Type	Description
id	Long	This is the Resource Profile's primary key in RMS.

Response Body

See the *AdHoc Booking Requests* section on page 25 for an example of what the response should include.

NOTE: *There is no response body for this API.*

Response Codes

HTTP Code	Meaning
200 (OK)	The adhoc booking response was successfully submitted to the RMS Server.
404 (Not Found)	The Resource Profile doesn't exist or isn't mapped to a location.

Get the Time Zone for Resource Profile

This API will get the time zone for the given Resource Profile from the RMS Server. This time zone information is useful in determining the boundaries for "today" in the particular time zone for the Resource Profile.

Request

HTTP Method	GET
URI Template	/api/v2/resources/{id}/tz
Example	http://acme.com/rms/api/v2/resources/27/tz

Request Path Variables

Parameter	Data Type	Description
id	Long	This is the Resource Profile's primary key in RMS.

There is no request body for this API.

Response Body

```
<timezone>
  <id>America/Chicago</id>
  <displayName>(UTC-06:00) America/Chicago</displayName>
  <shortName>CST</shortName>
  <dstShortName>CDT</dstShortName>
  <longName>Central Standard Time</longName>
  <dstLongName>Central Daylight Time</dstLongName>
  <utcName>UTC-06:00</utcName>
  <rawOffset>-21600000</rawOffset>
  <offset>-18000000</offset>
  <inDST>true</inDST>
  <usesDST>true</usesDST>
</timezone>
```

Response Codes

HTTP Code	Meaning
200 (OK)	The time zone for the Resource Profile was successfully retrieved from the RMS Server.
404 (Not Found)	The Resource Profile doesn't exist.
409 (Conflict)	The Resource Profile isn't mapped to a location.

Data Structures

Overview

RMS Enterprise uses XML for the data interchange format. The following data structures are used in the RMS Enterprise Scheduling API.

Booking

Booking Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Booking in the RMS database
version	Long	Row version for this Booking in the RMS database
externalBookingId	String(400)	The Booking's external scheduling system ID. This must be unique
hashedExternalBookingId	String(100)	A SHA-256 one-way hash of the Booking's external scheduling system ID. This must be unique
singleEvent	Boolean	Indicator for if this Booking is a single or recurring appointment
startDateTimeMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's start date/time
endDateTimeMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's end date/time
event	Event	The Event object associated with this Booking
bookingAuxiliary	BookingAuxiliary	The BookingAuxiliary object associated with this Booking

BookingAuxiliary

The BookingAuxiliary message is a data structure that is used internally in support of the Synchronization Workflow (see page 7) and is referenced in the following APIs: *Delete Troller* (page 10), *Delete Resource Profiles for Troller* (page 12), *Delete All Resource Profiles for Troller* (page 13). *Delete Bookings for a Resource Profile* (page 17), *Delete Bookings* (page 18), *Delete a Single Booking* (page 18), as well as the *Booking* data structure (above). This data structure is not exposed to the SDK. Therefore, the elements described below are not intended to be used or modified.

BookingAuxiliary Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this BookingAuxiliary in the RMS database
version	Long	Row version for this BookingAuxiliary in the RMS database
auxText1	String(255)	An unused auxiliary text field
auxText2	String(255)	An unused auxiliary text field
auxText3	String(255)	An unused auxiliary text field
auxText4	String(255)	Field used to override an occurrence meeting subject from the Event for a recurring meeting
auxText5	String(255)	An unused auxiliary text field
auxInt1	Integer	An unused auxiliary integer field
auxInt2	Integer	An unused auxiliary integer field
auxInt3	Integer	An unused auxiliary integer field
auxInt4	Integer	An unused auxiliary integer field
auxInt5	Integer	An unused auxiliary integer field
auxMemo1	String [ntext]	Field used to override an occurrence meeting body/details from the Event for a recurring meeting
auxMemo2	String [ntext]	An unused auxiliary memo field
auxMemo3	String [ntext]	An unused auxiliary memo field
localStartDateTimeMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's start date/time.
localEndDateTimeMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's end date/time.
lastTrollMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT when this Booking was trolled.

BookingAuxiliary Data Structures		
Element	**Data Type**	**Description**
lastModifiedMillis	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT when this Booking was last modified.

Event

Event Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Event in the RMS database
version	Long	Row version for this Event in the RMS database
externalEventId	String(400)	The Event's external scheduling system ID. This must be unique
hashedExternalEventId	String(100)	A SHA-256 one-way hash of the Event's external scheduling system ID. This must be unique
allDayEvent	Boolean	Indicator for if this Event is an all day event
privateEvent	Boolean	Indicator for if this Event is private
subject	String(100)	The subject of the meeting
details	String(nvarchar max)	The body of the meeting request
organizer	Attendee	The person who scheduled this meeting
onBehalfOf	Attendee	The person for whom the organizer was acting when scheduling the meeting
attendees	List of Attendees	The persons invited to attend the meeting

Attendee

Attendee Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Attendee in the RMS database
version	Long	Row version for this Attendee in the RMS database
externalId	String(255)	The Attendee's external scheduling system ID. E.g. john.doe@email.com
friendlyName	String(100)	The friendly name of the person. E.g. John Doe

Timezone

This is a read-only object. There are only APIs to read time zones from the RMS Server.

Timezone Data Structures		
Element	**Data Type**	**Description**
id	Long	America/Chicago (The Java time zone ID)
displayName	String	(UTC-06:00) America/Chicago
shortName	String	CST
dstShortName	String	CDT
longName	String	Central Standard Time
dstLongName	String	Central Daylight Time
utcName	String	UTC-06:00
rawOffset	Int	-21600000 (The time in milliseconds to add to UTC to get standard time in this time zone)
offset	Int	-18000000 (The time in milliseconds to add to UTC to get the time in this time zone right now. This value changes for time zones that use daylight saving.)
inDST	Boolean	Is this time zone in daylight saving right now?
usesDST	Boolean	Does this time zone use daylight saving at all?

Troller Message

Troller Message Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Troller Message in the RMS database
version	Long	Row version for this Troller Message in the RMS database
command	String(50)	The type of command the RMS Server is instructing the Troller to do. There are only three currently supported. resource_profile_mapped resource_profile_unmapped booking_request
message	String(nvarchar max)	An embedded XML document appears here for a booking_request. Otherwise, it is the Resource Profile's external ID.

Booking Request

This Booking Request message originates from a Client Gateway on behalf of a user's action taken at the touch panel. This message flows from the Client Gateway through the RMS Server and to the Troller who is responsible for fulfilling the request against the backend scheduling system.

Booking Request Data Structures		
Element	**Data Type**	**Description**
type	Int	The type of adhoc booking request: 0 = CREATE, 1 = EXTEND, 2 = END
externalBookingId	String(400)	The Booking's external scheduling system ID. This must be unique
startDateTime	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's start date/time
endDateTime	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's end date/time
subject	String(100)	The subject of the meeting
details	String(nvarchar max)	The body of the meeting request
clientGatewayUid	String(100)	The client gateway's globally unique identifier.
resourceProfile	Long	The Resource Profile's RMS database ID. This informs the Troller which resource to process this booking request against.
location	Long	The Location's RMS database ID. Normally, this is extra data not used by the Troller.

Booking Response

The Booking Response message is the Troller's response to a Booking Request. It will flow from the Troller through the RMS Server and to the Client Gateway that originally kicked off this process with a Booking Request.

Booking Response Data Structures		
Element	**Data Type**	**Description**
type	Int	The type of the response should match the type of the adhoc booking request: 0 = CREATE, 1 = EXTEND, 2 = END
success	Boolean	"true" or "false". Did this adhoc request succeed?
errorMessage	String	If the adhoc request failed, this element should be populated with a brief explanation. This text will be rendered on the touch panel as a response to the initial request.
externalBookingId	String(400)	The Booking's external scheduling system ID. This must be unique
startDateTime	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's start date/time
endDateTime	Long	Number of milliseconds since January 1, 1970, 00:00:00 GMT for this Booking's end date/time
subject	String(100)	The subject of the meeting
clientGatewayUid	String(100)	The client gateway's globally unique identifier.

Resource Profile

Booking Response Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Resource Profile in the RMS database
version	Long	Row version for this Resource Profile in the RMS database
externalId	String(400)	The Resource's external scheduling system ID. This must be unique
hashedExternalId	String(100)	A SHA-256 one-way hash of the Resource's external scheduling system ID. This must be unique
friendlyName	String(255)	The name of the room / resource in the scheduling system
mapped	Boolean	A flag that indicates if this Resource has been mapped to an RMS Location in RMS.
offset	Long	-18000000 (The time in milliseconds to add to UTC to get the time where this Resource is located)

Troller

Troller Data Structures		
Element	**Data Type**	**Description**
id	Long	Primary key of this Troller in the RMS database
version	Long	Row version for this Troller in the RMS database
name	String(400)	The Troller's name.

Error

When a server-side exception occurs during one of these web service API calls, an XML error document with the fields described below.

Error Data Structures		
Element	**Data Type**	**Description**
httpStatusCode	Long	The HTTP status code
exceptionClass	String	The fully-qualified Java class name of the server-side exception
Message	String	The message of the exception

```
<error>
  <httpStatusCode>404</httpStatusCode>
  <exceptionClass>com.amx.rms.ObjectNotFoundException</exceptionClass>
  <message>Troller object does not exist</message>
</error>
```

Troller Messages

Overview

The following are some examples of the Troller Messages.

Resource Profile Mapped

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <trollerMessages>
    <trollerMessage>
    <id>3</id>
    <version>0</version>
    <command>resource_profile_mapped</command>
<message>eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google.com</message>
  </trollerMessage>
</trollerMessages>
```

Resource Profile UnMapped

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <trollerMessages>
    <trollerMessage>
    <id>4</id>
    <version>0</version>
    <command>resource_profile_unmapped</command>
<message>eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google.com</message>
  </trollerMessage>
</trollerMessages>
```

AdHoc Booking Requests

There are three types of adhoc booking requests.: *New Meeting Request*, *Extend Meeting Request* and *End Meeting Request*:

New Meeting Request

A *New Meeting Request* is initiated by a user from a touch panel. This new meeting request flows through the RMS Server and to the Troller responsible for synchronizing the appointments for that Resource Profile / Location. The Troller must make a new meeting request to the scheduling system and then respond back to the RMS Server with the result.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<trollerMessages>
  <trollerMessage>
  <id>6</id>
  <version>0</version>
  <command>booking_request</command>
  <message>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;
    standalone=&quot;yes&quot;?&gt;&lt;bookingRequest&gt;&lt;type&gt;0&lt;/
    type&gt;&lt;startDateTime&gt;1398383100000&lt;/startDateTime&gt;&lt;endDateTime&gt;1398386700000&lt;/
    endDateTime&gt;&lt;subject&gt;budget&lt;/subject&gt;&lt;details&gt;Discuss budget proposal&lt;/
    details&gt;&lt;clientGatewayUid&gt;00-60-9F-92-3A-0E&lt;/clientGatewayUid&gt;&lt;location&gt;6&lt;/
    location&gt;&lt;resourceProfile&gt;1&lt;/resourceProfile&gt;&lt;/bookingRequest&gt;</message>
  </trollerMessage>
</trollerMessages>
```

The message element contains the details of the new Booking Request. The type of 0 is a CREATE Booking Request and is used for creating a new meeting in the scheduling system. These details are in an XML document embedded into the *message* element. It looks like this once decoded:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bookingRequest>
  <type>0</type>
  <startDateTime>1398383100000</startDateTime>
  <endDateTime>1398386700000</endDateTime>
  <subject>budget</subject>
  <details>Discuss budget proposal</details>
  <clientGatewayUid>00-60-9F-92-3A-0E</clientGatewayUid>
  <location>6</location>
  <resourceProfile>1</resourceProfile>
</bookingRequest>
```

The following XML shows the Booking Response from the Troller to the RMS Server:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bookingResponse>
  <type>0</type>
  <success>true</success>
  <errorMessage></errorMessage>
  <startDateTime>1398383100000</startDateTime>
  <endDateTime>1398386700000</endDateTime>
  <subject>budget</subject>
  <clientGatewayUid>00-60-9F-92-3A-0E</clientGatewayUid>
  <externalBookingId>external-booking-id-123</externalBookingId>
</bookingResponse>
```

If a failure occurs, the *errorMessage* should be filled with the description of what went wrong and *success* should be set to *false*.

If the new Booking Request succeeds, then the Booking Response is sent to the RMS Server and the Troller kicks off a full synchronization process for the Resource Profile affected by this adhoc request.

Extend Meeting Request

An *Extend Meeting Request* is initiated by a user from the touch panel in the Location for the current meeting in progress. This action attempts to extend the end time of the given meeting by a certain amount of time. The Troller is responsible for issuing a meeting modification to the scheduling system. The results, either success or failure (possibly due to a conflict), are reported back to the RMS Server in a Booking Response.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<trollerMessages>
  <trollerMessage>
    <id>8</id>
    <version>0</version>
    <command>booking_request</command>
    <message>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;
    standalone=&quot;yes&quot;?&gt;&lt;bookingRequest&gt;&lt;type&gt;1&lt;/
    type&gt;&lt;externalBookingId&gt;eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google
    .com|ca3cp6tv250tp5tvr7dj408pv0&lt;/externalBookingId&gt;&lt;startDateTime&gt;0&lt;/
    startDateTime&gt;&lt;endDateTime&gt;1398382200000&lt;/endDateTime&gt;&lt;clientGatewayUid&gt;00-60-9F-92-
    3A-0E&lt;/clientGatewayUid&gt;&lt;location&gt;6&lt;/location&gt;&lt;resourceProfile&gt;1&lt;/
    resourceProfile&gt;&lt;/bookingRequest&gt;</message>
  </trollerMessage>
</trollerMessages>
```

The *message* element contains the details of the extend meeting Booking Request. The *type* of *1* is an *EXTEND* Booking Request and is used for extending the end time of an existing meeting in the scheduling system. The embedded XML document in the *message* element looks like this once decoded:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bookingRequest>
  <type>1</type>

<externalBookingId>eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google.
com|ca3cp6tv250tp5tvr7dj408pv0</externalBookingId>
  <startDateTime>0</startDateTime>
  <endDateTime>1398382200000</endDateTime>
  <clientGatewayUid>00-60-9F-92-3A-0E</clientGatewayUid>
  <location>6</location>
  <resourceProfile>1</resourceProfile>
</bookingRequest>
```

The Booking Response is the same as with New Meeting Request.

End Meeting Request

An *End Meeting Request* is initiated by a user from the touch panel in the Location for the current meeting in progress. This action attempts to end the current meeting early. The Troller is responsible for issuing a meeting modification to the scheduling system. The results, either success or failure are reported back to the RMS Server in a Booking Response.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<trollerMessages>
  <trollerMessage>
    <id>9</id>
    <version>0</version>
    <command>booking_request</command>
    <message>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;
    standalone=&quot;yes&quot;?&gt;&lt;bookingRequest&gt;&lt;type&gt;2&lt;/
    type&gt;&lt;externalBookingId&gt;eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google
    .com|ca3cp6tv250tp5tvr7dj408pv0&lt;/externalBookingId&gt;&lt;startDateTime&gt;1398377700000&lt;/
    startDateTime&gt;&lt;endDateTime&gt;1398380648766&lt;/endDateTime&gt;&lt;subject&gt;Enter Meeting
    Subject&lt;/subject&gt;&lt;details&gt;Enter Meeting Details&lt;/details&gt;&lt;clientGatewayUid&gt;
    00-60-9F-92-3A-0E&lt;/clientGatewayUid&gt;&lt;location&gt;6&lt;/location&gt;&lt;resourceProfile&gt;1&lt;/
    resourceProfile&gt;&lt;/bookingRequest&gt;
    </message>
  </trollerMessage>
</trollerMessages>
```

The *message* element contains the details of the end meeting Booking Request. The *type* of *2* is an *END* Booking Request and is used for ending the current meeting early in the scheduling system.

The embedded XML document in the *message* element looks like this once decoded:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<bookingRequest>
    <type>2</type>
<externalBookingId>eng.googl.amxservice.com_36373136373734352d323730@resource.calendar.google.com|ca3cp6tv250t
p5tvr7dj408pv0</externalBookingId>
    <startDateTime>1398377700000</startDateTime>
    <endDateTime>1398380648766</endDateTime>
    <subject>Enter Meeting Subject</subject>
    <details>Enter Meeting Details</details>
    <clientGatewayUid>00-60-9F-92-3A-0E</clientGatewayUid>
    <location>6</location>
    <resourceProfile>1</resourceProfile>
</bookingRequest>
```

The Booking Response is the same as with New Meeting Request.

Appendix

Glossary

Troller	The Troller represents an instance of a scheduling interface to the RMS Server. A Troller's responsibility is to synchronize appointments between the scheduling system and the RMS Server for all of the Resource Profiles that are mapped to Locations in RMS. The Troller is written to consume both the scheduling system's API and the RMS Scheduling API. There can be multiple Trollers synchronizing appointments to the same RMS Server simultaneously. All of the other scheduling related objects are in effect owned by a Troller instance.
Resource Profile	A Resource Profile represents a schedulable resource in the scheduling system (e.g. a conference room). A Resource Profile must be mapped to a Location in the RMS UI before appointment synchronization for the resource can take place.
Troller Message	A Troller Message is the means by which the RMS Server communicates instructions to the Troller. These messages are picked up on every heartbeat to the RMS Server. There are three types of Troller Messages: adhoc booking requests, map resource profile requests, and un-map resource profile requests.
Heartbeat	A Troller must issue a Get Troller Messages request on a periodic interval (e.g. every five seconds) in order to remain online. If two minutes elapses without a heartbeat from a Troller, the RMS Server will mark the Troller as offline and emit notifications stating that the Troller has stopped communicating with the RMS Server and appointment synchronization has stopped.
Adhoc Booking Request	An adhoc booking request represents an action taken from a touch panel in a location that has some scheduling impact. There are three types: 1) *Add Booking* - This is a new meeting request submitted anonymously (i.e. no organizer) from a touch panel in the location (as opposed to a meeting submitted from a user's scheduling application, e.g. Outlook). 2) *End Meeting* - Request that the current meeting be ended early. This frees up the location for others to reserve. 3) *Extend Meeting* - Request that the current meeting be extended by some amount. This request must be validated against the scheduling system to ensure there is no conflict. The Troller must respond to these adhoc booking requests from the RMS Server by taking appropriate action with the scheduling system.
Map Resource Profile Request	The request indicates to the Troller that a Resource Profile has been mapped to a Location in the RMS UI and should now be synchronized. An initial synchronization should be started immediately and this Resource Profile should be included in all subsequent full troll cycles.
UnMap Resource Profile Request	A Resource Profile has been unmapped from a Location in the RMS UI and all of its Events and Bookings have been deleted from the RMS database. The Troller should stop synchronizing appointments for this Resource Profile in all subsequent full troll cycles.
Event	An Event represents an appointment being synchronized from the scheduling system. It contains details such as the subject, body, organizer, start date, end date, etc. If an appointment is a recurring appointment, there will only be a single Event record representing the entire series.
Booking	A Booking contains the other required details of an appointment such as the start time and end time of the appointment. There is a one to many relationship between Event and Booking for a recurring appointment. In the case of a single appointment, there will only be one Event and one Booking.
BookingAuxiliary	A BookingAuxiliary is an object for scenarios where supplemental data fields are needed for information that will not fit inside the Booking record itself. The BookingAuxiliary has five text fields (Nvarchar(255)), five integer fields, and three memo fields (Ntext). There is a one to one relationship between Booking and BookingAuxiliary. *Note: This data structure is not exposed to the SDK. Therefore, the elements described on page 21 are not intended to be used or modified.*

3000 RESEARCH DRIVE, RICHARDSON, TX 75082 AMX.com | 800.222.0193 | 469.624.8000 | +1.469.624.7400 | fax 469.624.7153

AMX (UK) LTD, AMX by HARMAN - Unit C, Auster Road, Clifton Moor, York, YO30 4GD United Kingdom • +44 1904-343-100 • www.amx.com/eu/

Last Revised:
9/14/2016