White Paper - Cisco Context Aware Mobility API

June, 2010

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: White Paper

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCDE, CCSI, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco Stackpower, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0903R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco Context Aware Mobility Services- University Asset Tracker Application User Guide
© 2010 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Executive Summary and Introduction** 1-1

CHAPTER 2 **Architectural Overview** 2-1

- Introduction to Architectural Overview 2-1
- Mobility Services Engine to Application 2-3

CHAPTER 3 **Technical Details** 3-1

- Introduction to Technical Details 3-1
- SOAP Background 3-4
- Common Services API Overview 3-6
- Physical Environment API Overview 3-7
- Context-Aware Mobility API Overview 3-8
  - Query Methods 3-9
  - Query Method Examples 3-9
  - Notification Methods 3-10
    - Event Trigger Categories 3-10
    - Event Trigger Device Match 3-10
    - Event Trigger Combinations 3-11
    - Notification Delivery Protocols 3-11
    - Notification Content 3-11

CHAPTER 4 **Benefits** 4-1

CHAPTER 5 **Examples** 5-1

- Example 1: Time based query 5-1
- Example 2: Floor and Time based query 5-1
- Example 3: User based query 5-2
- Example 4: Movement based notification 5-3
- Example 5: Containment based notification 5-3

CHAPTER 6 **Conclusion** 6-1

CHAPTER 1

Executive Summary and Introduction

The Mobility Services Engine (MSE) introduces a platform that runs one or more mobility services including the Context-Aware Mobility Service. The Context-Aware Mobility Service includes technology that can capture information on network equipment, network sensors, environmental sensors, mobile network devices, and mobile assets. By knowing where a device is located, the network can provide contextual information to that device or other systems to improve network, device, and service usefulness.

The Context-Aware Mobility Service can integrate network device information (such as location) with other systems to improve or enhance their business functionality. For example, a manufacturing process tool can use this service to receive real-time contextual information during product manufacturing on all the assets in the network.

For another example, a hospital monitoring system that has active RFID tags with telemetry sensor information for temperature can communicate this information to the temperature monitoring system using the Context-Aware Mobility Service.

The Context-Aware Mobility Service provides this integration by exposing a programmatic API that allows these external systems to access the information and capabilities provided by the service.

The following documentation includes the following:

- Introduction to Context-Aware Mobility Service capabilities.
- Introduction to the information that is available through the API.
- Information on how the Context-Aware Mobility Service can be used by external systems to facilitate and enhance new applications by integrating the available contextual information.

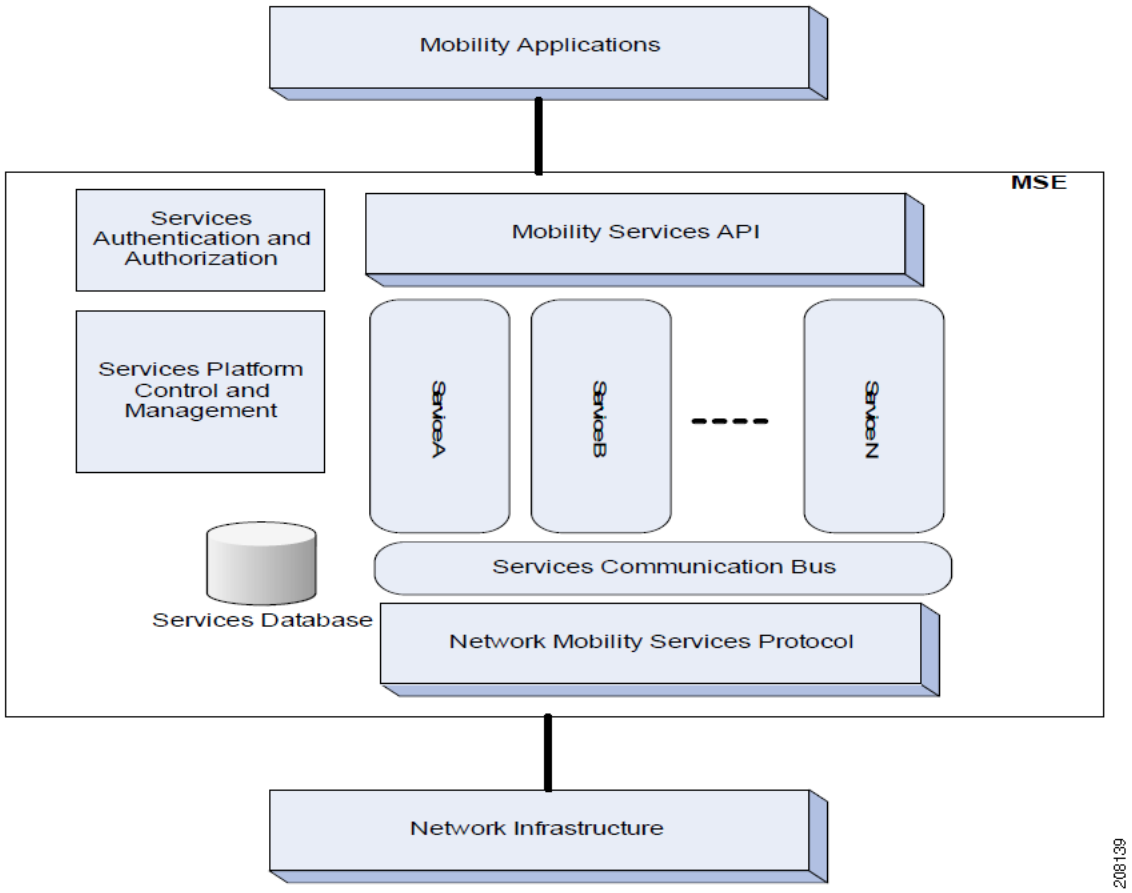
CHAPTER 2

Architectural Overview

Introduction to Architectural Overview

The high level architecture of the MSE is shown in Figure 2-1.

Figure 2-1 MSE High Level Architecture



The Context-Aware Mobility Service, like other mobility services, is a software instance running on the MSE.

The Context-Aware Mobility Service has the following characteristics:

- It functions across multiple edge technologies such as 802.11 wireless and 802.3 wired networks.

- It functions across multiple network elements.
- It provides an interface to external applications and servers using a mobility service API.
- It adds intelligence to the network to enhance network usability.
- It increases network visibility for applications and servers.
- It can be combined with other mobility services to achieve higher order functions.
- It is managed by the mobility service API.
- It may be deployed across multiple MSE to scale its function.

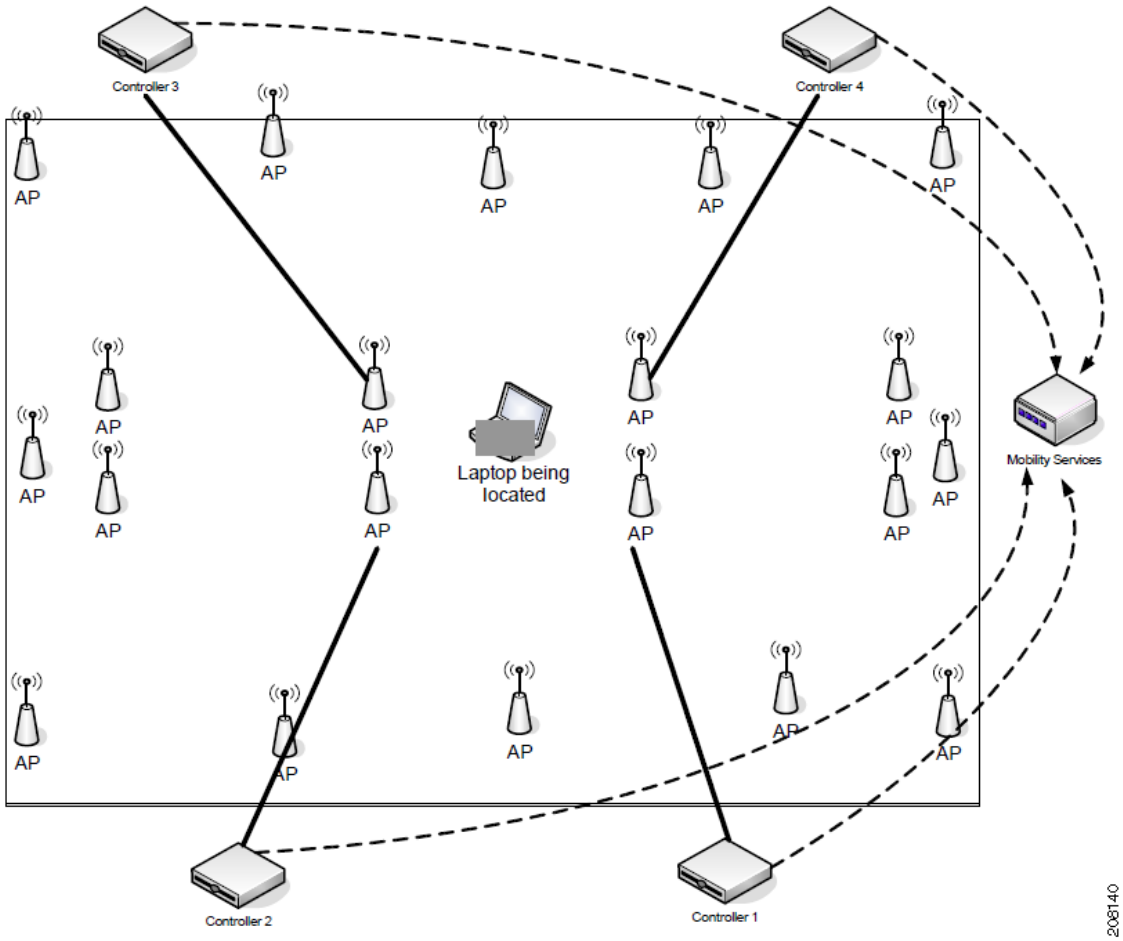
The Context-Aware Mobility Service can capture and integrate detailed contextual information into business processes. This information may include location, temperature, availability, and applications used. It is a mobility service that covers multiple edge technologies.

The Context-Aware Mobility Service, through its mobility services API, provides a single unified view of contextual information (such as location) for network devices. It enables queries for contextual information as well as registration for asynchronous events based on movement, containment, absence, and other advanced context-aware event triggers.

Applications and servers that consume context-aware information no longer have to be written to multiple APIs across a wide variety of technologies. This service, running on the MSE, simplifies the task of building context-aware applications and services.

Because the service runs on the MSE rather than on an individual switch or wireless controller, it provides a more scalable network design. This occurs because location is determined for a network comprised of wireless access points or other location receiver technologies that are aggregated at the switch or wireless controller. In a typical enterprise wireless deployment, there are many access points deployed in a physical space (such as a floor). For redundancy and scalability, those access points are connected to multiple wireless controllers rather than a single wireless controller as shown in Figure 2-2.

Figure 2-2 Typical wireless controller to AP deployment for location



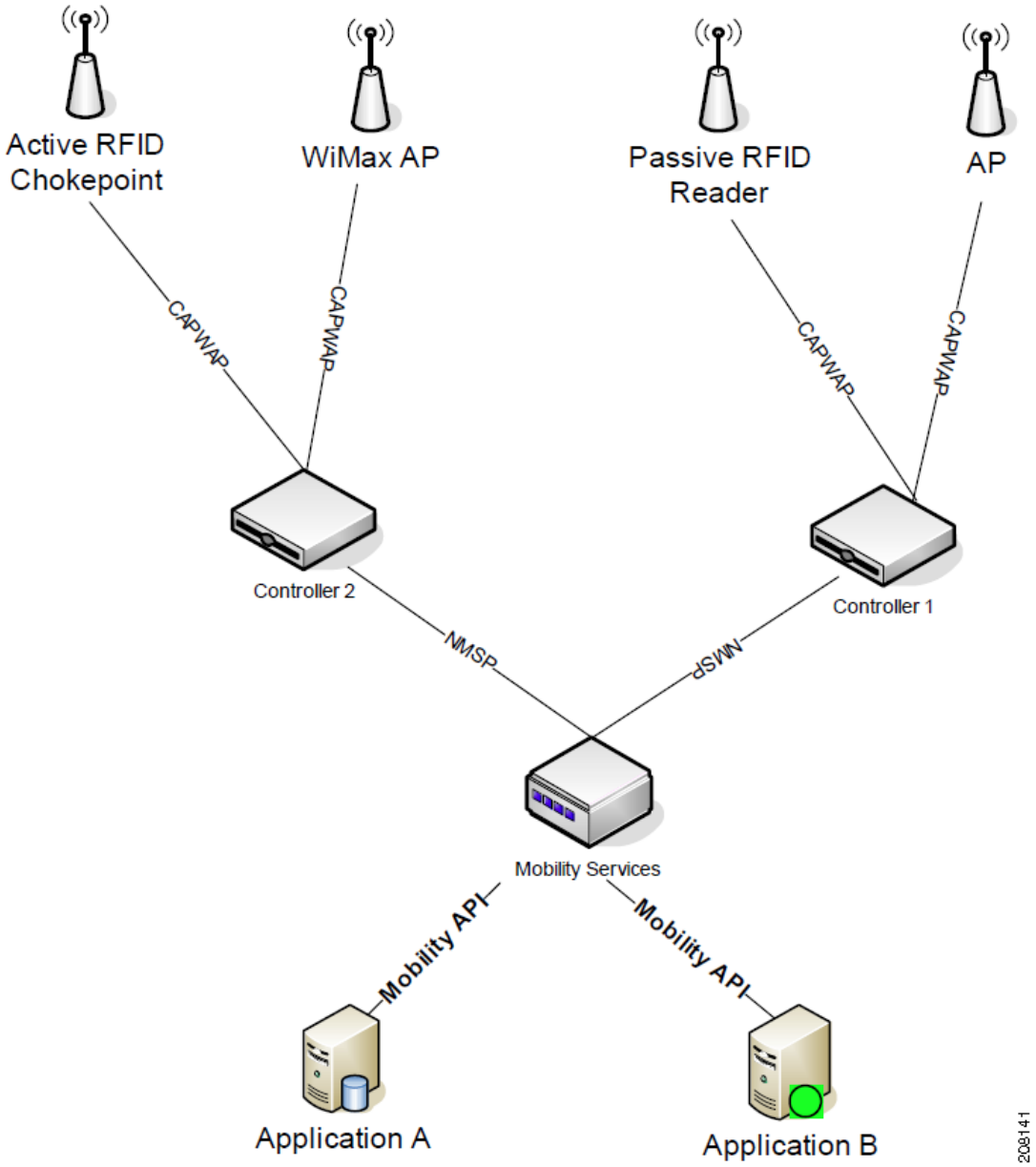
To calculate location for the laptop shown in Figure 2-2, a MSE running the Context-Aware Mobility Service must collect information from all controllers (and their access points) in the surrounding physical environment, rather than a single wireless controller (and its access points). Because of this, it is necessary to run the Context-Aware Mobility Service on an appliance or server that aggregates all access point measurements from multiple wireless controllers.

In addition, location calculations must be performed at a high rate (and within a matter of a few seconds) to enable context consumers to take advantage of context-aware information. The results of a dedicated platform include a more scalable service that can meet the needs of high performance applications that use contextual information.

Mobility Services Engine to Application

The Mobility Services API is an interface that provides management and data access to the services running on the MSE as shown in Figure 2-3.

Figure 2-3 Mobility API



The Mobility Services API is the key interface to external applications and servers that use the mobility services. It is the administrative interface for services management as well as a real-time interface for service consumers.

The key objective of the Mobility Services API is to consistently provide services and data to consumers. This consistent view of data replaces individual services that have separate mechanisms for authentication and authorization, separate style guides for how common functions are performed, and separate data representations for common objects such as a controller or access point.

Application integrators now have a simplified task when considering the use of multiple services on the MSE. For example, network administrators can apply policies for access control across a set of services rather than having to adapt separate policy management for each individual service.

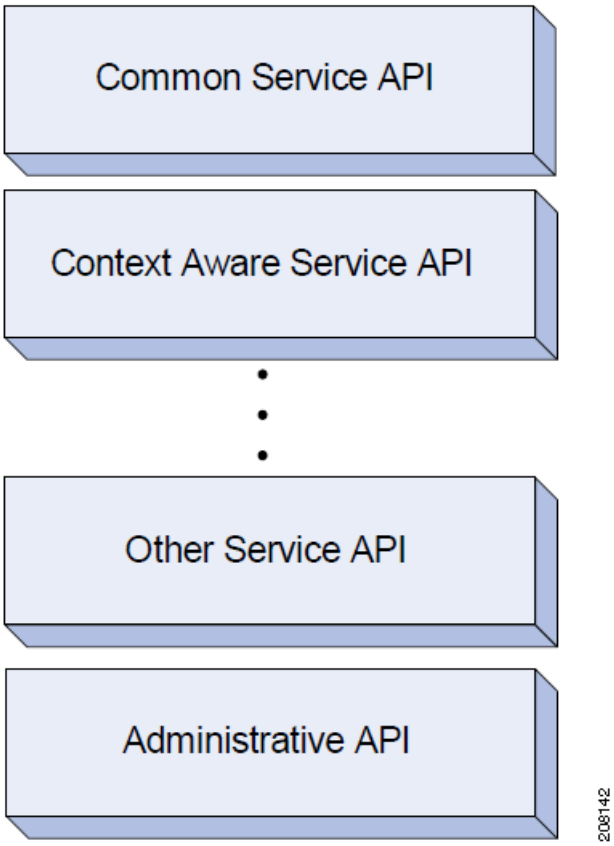
CHAPTER 3

Technical Details

Introduction to Technical Details

Figure 3-1 displays a summary of the APIs that are exposed on the MSE and by the services that run on the MSE.

Figure 3-1 MSE API

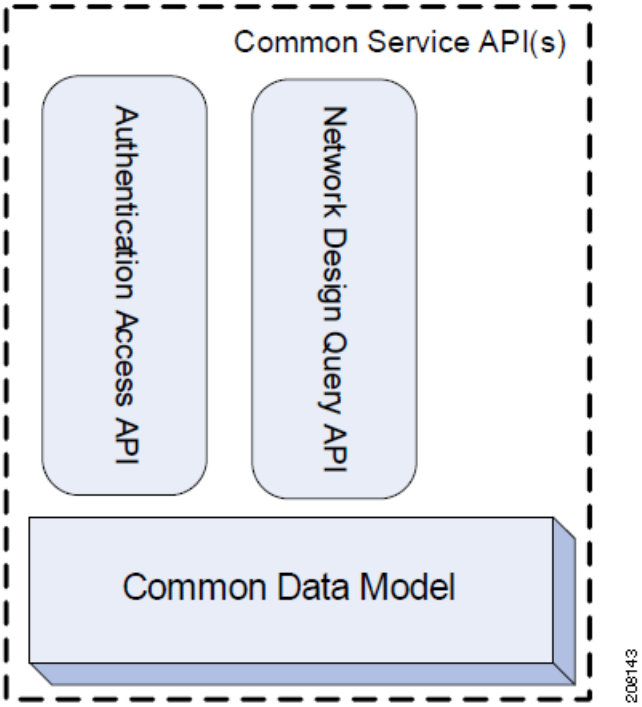


All of the APIs are defined as SOAP (Simple Object Access Protocol)/XML (Extensible Markup Language) interfaces accessible over HTTP(S).

In this document, “service user” represents any API user. The service user may be a server or application that uses one or more mobility services running on the MSE.

The Common Service API provides the methods and data structures that are used by all services running on the MSE (Figure 3-2).

Figure 3-2 Common Service API



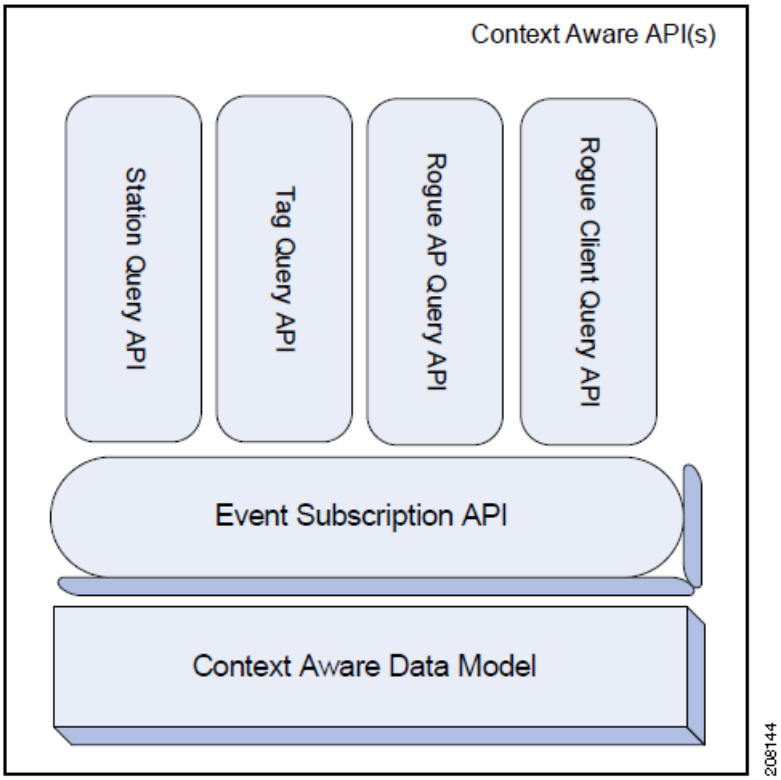
The common service API consists of two primary APIs and a common data model.

The APIs are:

- Authentication Access API - Used by the service user to authenticate against the MSE platform and obtain a valid business session for further use on the mobility services APIs.
- Network Design Query API - Used by the service user to read or query the physical environment structures including buildings, floors, and access point positions that are used by many mobility services to communicate consistently about the physical environment.

The Context-Aware Mobility API provides the methods and data structure that are specific to the context-aware service (Figure 3-3).

Figure 3-3 Context Aware Service API



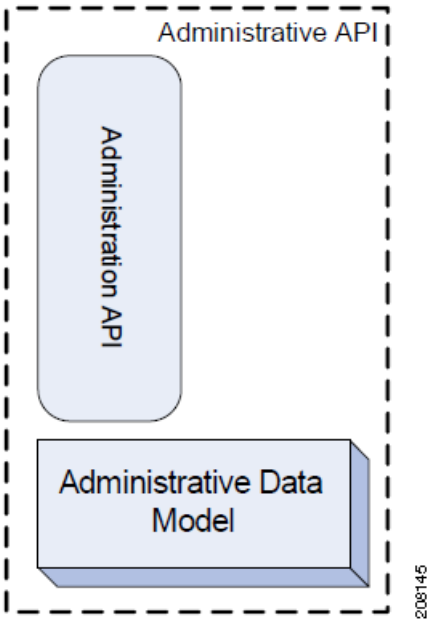
The context-aware mobility API consists of five primary APIs and a context-aware data model that is shared across the context-aware methods. The APIs include:

- Station Query API—Provides the methods to perform queries on 802.11 client devices including:
  - Information records such as MAC address, IP address, and 802.1X username
  - Current physical location
  - Historical location up to a configurable limit
  - Traffic statistics for the client
- RFID Tag Query API—Provides the methods to perform queries on 802.11 RFID tags including:
  - Information records such as MAC Address and battery life
  - Current physical location
  - Historical location up to a configurable limit
  - Traffic statistics for the tag
- Rogue AP Query API—Provides the methods to perform queries on 802.11 rogue access points including:
  - Information records such as MAC address and IP address
  - Current physical location
  - Historical location up to a configurable limit
- Rogue Client Query API—Provides the methods to perform queries on 802.11 rogue clients including:
  - Information records such as MAC address and IP address

  – Current physical location
  – Historical location up to a configurable limit
- Event Subscription API—Provides the methods to subscribe for events related to clients or tags for events:
  – In or out containment
  – Movement
  – Update
  – Absence
  – Battery life
  – Emergency

The Administrative API provides the methods and data structure that are specific to the administration of the MSE and its services (Figure 3-4).

Figure 3-4 Administrative API



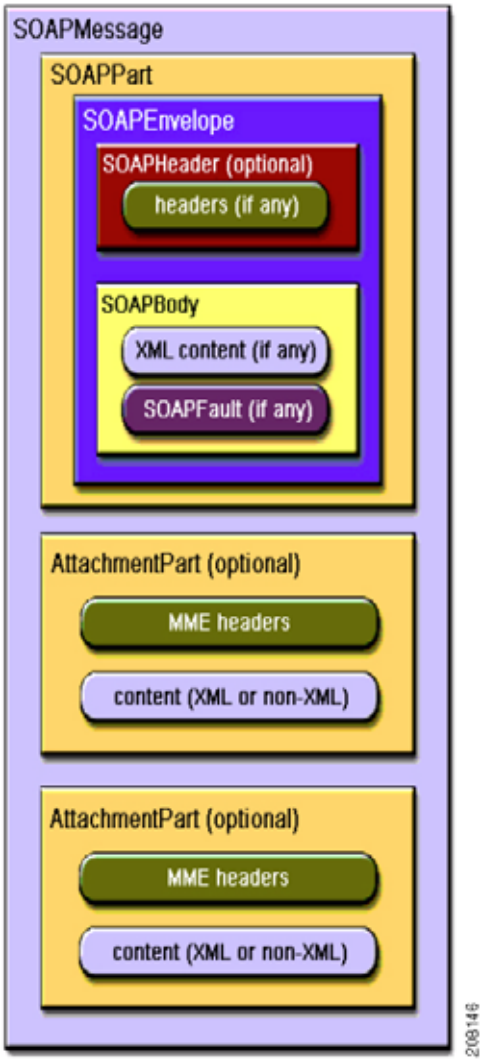
The administrative API consists of a set of methods and data model that is used by the administrator to manage the MSE and its services.

This paper focuses on the Context-Aware Mobility API. Contact Cisco for further details on the administrative API.

SOAP Background

The following section explains the structure of a SOAP message.

Figure 3-5 SOAP Message Structure



The MSE and Context-Aware Mobility API define the content of the messages within the SOAPBody as shown in Figure 3-5.

All request messages transmitted to the MSE include the following information:

- Request Name—Identifies the method being called including the namespace that identifies the service responsible for the method.
- Session—Identifies the authenticated session.
- List of request objects—A list of objects that are parameters to the request method.
- An optional list of attachments—Certain requests require the sender to attach content such as image files.

The following is an example of a request:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”>
<SOAP-ENV:Body>
<Ping xmlns="http://cisco.com/mse/location">
<AesBusinessSession id="7406"/>
<AesPing area="database"/>
</Ping>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

For responses, the MSE defines the message content as:

- Response—Identifies that this message is a response and the namespace for the response.
- List of response objects—A list of objects that are the result of the request.

The following is an example of a response:

```
<SOAP-ENV:Body xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<Response xmlns="http://cisco.com/mse/location">
<AesResult result="true"/>
</Response>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Common Services API Overview

Before using other methods defined for a mobility service, the service user must request a business session from the MSE platform. The MSE platform maintains all sessions for service users regardless of which other services that service user is using on the MSE.

The following is the common process used by an MSE API service user:Before using other methods defined for a mobility service, the service user must request a business session from the MSE platform. The MSE platform maintains all sessions for service users regardless of which other services that service user is using on the MSE.

The following is the common process used by an MSE API service user:

Step 1 The service user connects to the MSE API and provides credentials to the MSE for authentication.

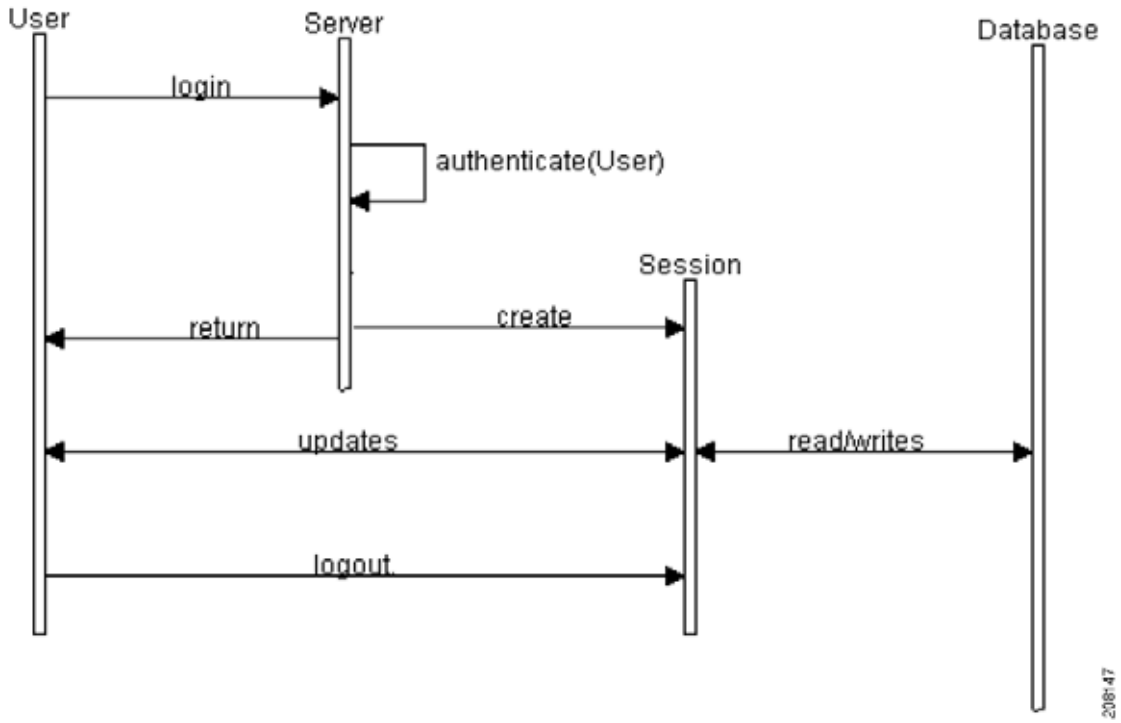
Step 2 The MSE authenticates the service user and, if allowed, provides a business session to the service user for use in all subsequent method calls to the MSE API.

Step 3 The service user calls a method or methods on mobility service(s) on the MSE. They exchange data with and receive results from the methods.

Step 4 The service user terminates MSE use by notifying the MSE that it no longer requires the business session.

This flow is depicted in Figure 3-6.

Figure 3-6 Common Service User API flow



The following methods are provided by the MSE API for session management:

- Login—Called by the service user when it requires access to other mobility service APIs.
  - The service user provides a username and password that is authenticated by the MSE upon method invocation.
- Logout—Called by the service user when it no longer requires access to other mobility service APIs.
  - The service user calls this method when it no longer requires active use of the MSE platform. This helps the MSE free up resources for other service users.
- Ping—Called by the service when it wishes to verify that the MSE platform is still active and able to perform its function.
  - The MSE discards business sessions that remain inactive for a configurable amount of time.
  - By calling this method more frequently than the stale session interval, the service user helps maintain a valid business session with the MSE when the service user is not actively calling other APIs on the MSE.

Physical Environment API Overview

All mobility services act on networks that are deployed in many different physical environments including office buildings, factories, hospitals, and city streets. To have a common understanding of the physical environment, the MSE platform defines a model that represents the physical environment and is shared by all mobility services running on the MSE. The Context-Aware Mobility Service provides data relating to that physical model which makes it critical that service users understand and can synchronize with the MSE's physical environment definition.

The primary objects in the physical environment data model include:

- Network Design—The top-level container used to represent any physical environment. It is the primary object to which service users refer when retrieving information about the physical environment.

Example: A network design that represents an enterprise campus.

- Group—A folder or organizational object that directly contains other groups or buildings.
- Building—This object class represents a building that may have one or more floors.
- Floor—This object class represents a single floor within a building. The floor is where network equipment and users of the network reside. It is the primary container for many of the object classes in the physical environment.
- Coverage Area—This object class represents an area within the floor that could represent all or part of the floor.
  - Examples include labs, offices, and conference rooms.

Each object class includes a name along with width, height, and length dimensions. These attributes are useful when synchronizing the data with a service user.

The primary methods to access the physical environment include:

- GetNetworkDesignList—Called by a service user to retrieve one or more network design objects and their associated attributes and child objects. This is typically done by a service user prior to using other mobility services that provide data related to one of the network designs.

  A service user can filter the results of the method by providing a query criteria for one or more specific attributes of the network design.

  Example: A service user wants to correlate the location information returned by the Context-Aware Mobility Service to a physical environment that the service user understands.
- GetNetworkDesign—Called by a service user to retrieve one specific network design object identified by name.

  This method is typically called by a service user that has knowledge about a specific network design on the MSE platform and wants to retrieve the additional information defined for the particular object.
- GetViewImage—Called by a service user to retrieve the graphic associated with a particular floor defined in the network design.

  This method is typically called by a service user that has a capability to show the graphics of a floor and relate that to other information it provides regarding that floor.

  Example: An asset management application intends to show the floor on which a device is located.

Context-Aware Mobility API Overview

The Context-Aware Mobility Service provides contextual information on many different classes of devices. The Context-Aware Mobility API provides two interfaces for service users to leverage the information:

- Query—Where the service user can query for a specific set of information related to one or more device in the network.
- Notification—Where the service user can register for notifications whenever specified events occur.

Query Methods

The query methods follow a simple naming technique that is used repeatedly in the API:

>> ***Get<DeviceType><Category><QueryType>***

where DeviceType, Category, and QueryType are replaced according to the following descriptions.

DeviceType

The device types defined in the API include:

- Station—Any 802.11 device.
- Tag—Any Active RFID device.
- RogueAP—Any 802.11 access point that is not part of the primary infrastructure network.
- RogueClient—Any 802.11 client device that is associated with a rogue access point.
- Interferer—Any interference source.

Category

The categories defined in the API include:

- Info—Network access information of the device including MAC address, IP address, 802.1X username, and LAN access security settings.
- Location—Physical location of the device type including building, floor, area, and position on the floor.
- Stats—Traffic statistics including octets and frames transmitted and received.
- History—The collection of info, location, and statistics as captured over time.

QueryType

The query types available in the API are as follows:

- Single Station Query Methods—These methods take a single MAC address as the key or search value and return a single object response.
  - The method names end with the category with no extension for query type.
- Multiple Station Query Methods—These methods take a query criteria to filter the results based on a set of attributes such as username, IP Address, network connection point, floor identifier.
  - A Query Criteria can provide simple or complex queries depending on the type of filtering the service user requires.
  - The method names end with the term “List” after the category.

Query Method Examples

GetStationLocationList

Called by a service user that is interested in one or more mobile station’s current physical location information. Typically a service user would provide a Query Criteria to filter the data based on a set of attributes such as username, IP address, and network connection point.

GetTagStats

Called by a service user that is interested in one specific tag's network traffic statistics identified by its MAC address. The statistical information includes both transmitted and received octets and frames allowing the service user to understand traffic flows related to the location of the tag.

GetRogueAPInfoList

Called by a service user that is interested in one or more rogue access point's network access information as filtered by a Query Criteria that filters the number of rogue access points by attributes such as MAC Address, and floor identifier.

Notification Methods

In addition to the data retrieval APIs, the Context-Aware Mobility Service API provides an asynchronous event interface that allows service users to register for notifications whenever an event occurs that matches the registration information known as "triggers."

Event Trigger Categories

Event triggers may be specified in the following situations:

- Containment—When a device moves in or out of a specified physical boundary such as a floor, building, coverage area, or network design.
- Movement—When a device moves beyond a distance greater than or less than a certain defined distance from a predefined marker point within a floor.
- Absence—When a device has not been seen on the network for a defined amount of time.
- Battery Life —When a tag sends a notification to the network indicating that its battery power may be low.
- Update—When a device's location information is updated and providing that location meets a configurable difference metric since the last update.
- Chokepoint—When a tag encounters a choke point.
- Emergency—When a tag notifies the MSE of an emergency using the CCX tag protocol.

Event Trigger Device Match

Event triggers specify whether they match a single device or a set of devices based on the following attributes:

- MAC address
- Asset Name
- Asset Group
- Asset Category

Event Trigger Combinations

Event triggers may be combined using logical operators to form more complex or specific event triggers. For example, a trigger that matches on containment may define the notification being sent when a device moves into the 1st floor or 3rd floor of a building. In this case the service user would define one trigger and logically combine containment for the 1st and 3rd floors using the OR operator.

Notification Delivery Protocols

Whenever an event trigger matches, a notification is generated. For any notification sent by the Context-Aware Mobility Service, the service user may choose to receive the notification using one or more of the following protocols:

- SMTP
- SNMP Trap
- SOAP/XML
- Syslog

For each transport choice there are additional attributes, such as TCP destination port number, that the service user must specify in order for the notification to be sent correctly. Additional information may be found in the API reference documentation.

Notification Content

Notifications are sent when one or more triggers are matched. Once a notification has been sent for a particular event trigger, the Context-Aware Mobility Service tracks the state of that event. If a particular condition clears, a notification is sent to clear that state. For example, a container event fires when a device moves into an area and the container event clears when the device moves outside of the area.

The common fields that are included in most notifications are:

- State—Defines whether this is a matched or cleared trigger.
- Track Definition name—The name of the trigger that caused the notification to fire. This name is specified by the service user during notification registration.
- Entity Type—The device type that identifies the type of related event. This helps service users distinguish events that are registered as wild cards and therefore could apply to any number of device types such as stations or tags.
- Entity ID—The device’s key identifier such as MAC Address.

Most event triggers contain additional fields that are specific to the notification being sent and the details of those fields may be found in the API reference documentation.

Example #1: Containment Event

<ContainmentTrackEvent state=”set” in=”true” trackDefn="update2" floor=”wcsArea,4th Floor,Bldg-14,WNBU_Group,WNBU,“ entityType="Mobile Station" entityID="00:0c:41:15:99:92"/>

CHAPTER 4

Benefits

The MSE and Context-Aware Mobility API provide many benefits to application and system integrators looking to build products that are enhanced or enabled by the Context-Aware Mobility service. These benefits include:

- Flexibility

  – Service users can retrieve data using a query (‘pull’) model or a subscription (‘push’) model.

  – Service users have a choice of individual models or a combination of models.

  – The API is compatible with any programming model.

- Network Isolation

  – The Context-Aware Mobility API provides contextual network data without requiring the service user to understand how that information is retrieved or is represented in the infrastructure across different device types from which the MSE gathers its data.

  – By using the Context-Aware Mobility API, the service user is isolated from those details and can focus on the primary task of delivering the business application using the contextual data provided.

  – The MSE platform acts as a secure aggregation point for all network infrastructure information. Configuration with secure access credentials for network devices is not necessary for the service user which improves the isolation between network and business applications.

- Location regardless of detection technologies

  – The Context-Aware Mobility API provides location information using the physical environment representation. It removes the underlying location detection technologies that were used to calculate the location result.

  – This removes the need for service users to understand these underlying detection techniques. They can focus on the location results and how the results fit with their business applications.

  – This provides future compatibility to applications that use location information from the API. No changes are required to the location-enabled applications when the underlying location determination technology changes.

- Improved scaling

  – In environments where multiple service users consume contextual information including location, it is potentially harmful to the network infrastructure if those service users access this information directly from the network infrastructure at the same time.

  – By having the MSE for the Context-Aware Mobility Service gather information for all of the service users in the network, the network infrastructure only provides the information once rather than for each service user.

– The Context-Aware Mobility Service can distribute the information to service users in a more scalable approach.

- Standard Interface Language and Rapid Integration

  – The Context-Aware Mobility API is primarily a SOAP/XML interface that is accessible using common web technologies such as WSDL (http://www.w3.org/TR/wsdl). This allows many web based service users defined in .NET, C++, Java, and so forth to efficiently use the capabilities of the API.

  – The Context-Aware Mobility API also provides notifications in many native protocols that service users may already support such as SNMP, SMTP, and syslog.

- API Consistency

  – The Context-Aware Mobility API provides contextual data for many categories of devices (including stations, tags, rogue access points) using a common API form and expression. This makes it easy for the service user programmer to become familiar with the API for any category of device once they know the API for a single category.

- Contextual combination

  – The Context-Aware Mobility API provides the combination of many sources of data including network access information, location information, and statistical information.

  – By combining this information in a single API, service users can build products without the need to gather information from multiple alternate sources.

CHAPTER 5

Examples

The following examples highlight the flexibility of the Context-Aware Mobility API to provide custom information based on the service user's requirements.

Example 1: Time based query

Description

A service user may want a graphical display that only updates location information for mobile stations that have moved in the last five minutes. This reduces the amount of information retrieved over the network in a short period of time and decreases the amount of time that it takes to display changes in the service user.

Details

The GetStationLocationList method is used with the additional Query Criteria that filters the query results based on the "changedOn" data attribute of the location object.

Example

```
<GetStationLocationList xmlns=" http://cisco.com/mse/location">
<AesBusinessSession id="18740" mask="-1"/>
<AesQueryCriteria numOfResults="-1">
<and-list>
<AesMatchMbrCriteria matchClass="AesMobileStationLocation" matchMbr="changedOn"
matchValue="300000" operator="1" valueClass="java.lang.Long"/>
</and-list>
</AesQueryCriteria>
</GetStationLocationList>
</SOAP-ENV:Body>
```

Example 2: Floor and Time based query

Description

A service user may want to show the mobile station location information on a display for a particular floor known to the service user and only show the information that has changed in the last five minutes. By filtering the data to a specific floor the service user avoids the costly retrieval of all location information for many buildings or floors that the Context-Aware Mobility service is tracking.

Details

The GetStationLocationList method is used with the additional Query Criteria that filters the query results based on the “changedOn” data attribute of the location object and the container that specifies the floor with a fully qualified name as “Campus1”->”Building1”->”Floor1.”

Example

```
<SOAP-ENV:Body xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<GetStationLocationList xmlns=” http://cisco.com/mse/location”>
<AesBusinessSession id="18740" mask="-1"/>
<AesQueryCriteria numOfResults="-1" options="2">
<and-list>
<AesMatchMbrCriteria matchClass="AesMobileStationLocation" matchMbr="changedOn"
matchValue="5" operator="4" valueClass="java.lang.Integer"/>
</and-list>
<containList>
<containEl containCl="AesFloor" size="3">
<hierEl val="floor1"/>
<hierEl val="building1"/>
<hierEl val="campus1"/>
</containEl>
</containList>
</AesQueryCriteria>
</GetStationLocationList>
</SOAP-ENV:Body>
```

Example 3: User based query

Description

A service user may want to show the mobile station location information on a display for a particular username known to the service user.

Details

The GetStationLocationList method returns the list of location objects that match the specified username. However, the username attribute is defined on the Info object and is not defined on the location object. Therefore the query must specify the username attribute and a join to return the correct location object. The result object is a location object and this is achieved by joining the location and information objects based on the MAC address key.

Example

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<GetStationLocationList xmlns=” http://cisco.com/mse/location”>
<AesBusinessSession id="21854" mask="-1"/>
<AesQueryCriteria>
<and-list>
<AesMatchColumnCriteria fieldName1="macAddress" fieldName2="macAddress"
tableName1="AesMobileStationLocation" tableName2="AesMobileStation"/>
<AesMatchMbrCriteria matchClass="AesMobileStation" matchMbr="username" matchValue="allant"
valueClass="java.lang.String"/>
</and-list>
</AesQueryCriteria>
</GetStationLocationList>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Example 4: Movement based notification

Description

A service user may want to receive notifications whenever a particular device moves beyond a defined point in the physical environment such as a doorway.

Details

The AddTrackGroup method is used to register the service user at IP address 10.77.152.14 to receive a SNMP trap whenever the device identified by MAC address “00:05:4e:4b:ae:7a” moves 20 feet from the doorway marked as “Door1.”

Example

```
<AddTrackGroup>
<AesBusinessSession id="2397" mask="-1"/>
<AesTrackGroup name="AbsenceTest">
<defnList>
<AesTrackDefn name="absenceTest">
<destDefnList>
<AesDestDefn destIPList="10.77.152.14,">
<transport>
<AesTransportSNMP community="public" portNumber="162"/>
</transport>
</AesDestDefn>
</destDefnList>
<trigger>
<AesTriggerMovement distance="20.0" param="00:05:4e:4b:ae:7a" refParam="Door1"/>
</trigger>
</AesTrackDefn>
</defnList>
</AesTrackGroup>
</AddTrackGroup>
```

Example 5: Containment based notification

Description

A service user may want to receive notifications whenever a particular device moves beyond a certain distance from a defined point in the physical environment such as a doorway.

Details

The AddTrackGroup method is used to register the service user at IP address 10.77.152.14 to receive a SNMP trap whenever any device with the MAC address prefix “00” moves into the floor identified by HCL-2->hclbuild1->floor1

Example

```
<AddTrackGroup>
<AesBusinessSession id="2397" mask="-1"/>
<AesTrackGroup name="SEETHATEST">
<defnList>
```

```
<AesTrackDefn name="SEETHA">
<destDefnList>
<AesDestDefn destIPList="10.77.152.14,">
<transport>
<AesTransportSNMP community="public" portNumber="162"/>
</transport>
</AesDestDefn>
</destDefnList>
<trigger>
<AesTriggerContainment containerSize="4" isIn="true" operator="1" param="00:">
<container element="HCL-2"/>
<container element="hclbuild1"/>
<container element="floor1"/>
</AesTriggerContainment>
</trigger>
</AesTrackDefn>
</defnList>
</AesTrackGroup>
</AddTrackGroup>
```

CHAPTER 6

Conclusion

The comprehensive interface of the Context-Aware Mobility API allows application and system integrators to rapidly develop context-aware business applications without binding them to the underlying network technologies.

Since the MSE acts as the unified interface, service users do not need to understand the interfaces and data structures feeding data into the MSE.

Because the Context-Aware API provides both push and pull programming models, service users can choose the API that best fits with their programmatic and business needs. They are not forced to use an API in a way that does not align easily with their current product.