

Location Based Application Hands-on Considerations

Jim DeLoach, Principal Engineer
QUALCOMM Engineering Services Group



Learning Objectives for this Presentation

- **Using the *iPosDet* interface:**
 - What are *iPosDet* modes?
 - When is each mode appropriate?
 - How do you optimize the Configuration Parameters?
 - What are the practical considerations for setting the parameters?
- **Controlling data usage with throttling algorithms**
- **Using uncertainty to optimize application behavior**

Using the BREW[®] Location API –*iPosDet*

The BREW Location API – *iPosDet*

- ***iPosDet* is the Application Programming Interface (API) in BREW for generating position fixes using QUALCOMM's gpsOne™ and QPoint™ technologies**

iPosDet Location Generation Method

- First some background – key to using *iPosDet* effectively is understanding the “location generation methods”
- Location generation methods supported:
 - **MS-Assisted** – the Location Server * (PDE) calculates the mobile’s position
 - **MS-Based** – the mobile station calculates it’s own position with limited periodic assistance from the server
 - **Standalone or Autonomous** – the mobile station calculates it’s own position with no server assistance
- *iPosDet* API parameters control how these methods are used

* The **Location Server** is the network element that provides assistance to mobiles in Assisted-GPS technology. Location servers go by many names, depending on the standards used and methods of operation of any given deployment. One common Location Server network element name is the **Position Determination Entity (PDE)**.

Brief Review of Location Generation Methods



MS-Assisted	MS-Based	Standalone
Assistance data sent to the mobile for every fix	Assistance data sent to mobile periodically and only as needed	NO assistance data at all
Mobile measures GPS pseudoranges, then returns them to server, where server makes final position calculation	Mobile measures GPS pseudoranges and makes position calculation	Mobile performs all operations with no assistance at all
<ul style="list-style-type: none"> • Has the greatest GPS sensitivity • Supports fall-back, backup position solution types, such as AFLT, thus maximum yield • A data call is needed for each fix 	<ul style="list-style-type: none"> • Uses only GPS satellites • Shortest latency between fixes enables rapid repetitive fixes for tracking and navigation • Only needs occasional data calls 	<ul style="list-style-type: none"> • Increased Time to Fix and battery consumption • Diminished yield and accuracy • No data calls needed
Fix available at the network	Fix available at the handset	Fix at handset only
<p>Best suited to applications requiring less frequent fixes, or maximum sensitivity:</p> <ul style="list-style-type: none"> • E-911 • Slower tracking applications • Fixes required in challenging locations 	<p>Best suited to handset-resident apps, particularly those requiring lots of fixes</p> <ul style="list-style-type: none"> • Turn-by-turn navigation • Fast tracking apps / geofencing 	<p>Best when out of wireless network coverage</p>

Components of *iPosDet*

- **Key Methods of *iPosDet*:**
 - *IPOSDET_SetGPSConfig* and *IPOSDET_GetGPSConfig* manage *iPosDet* configuration and thus how location generation occurs
 - *IPOSDET_GetGPSInfo* is an asynchronous interface which returns position information (lat/lon/altitude, velocity, uncertainty information, etc.)
- **This presentation focuses on how to get the most out of Assisted-GPS, rather than the software details**

IPosDet Parameters

- **Invoking a position fix via the *IPosDet* API is a 2-step process:**
 - Configuring the gpsOne engine
 - Requesting a positioning fix
- **Parameters which drive Assisted-GPS performance:**

Configuration Parameters: *

- Mode
- Optim
- nFixes
- nInterval
- QOS
- Server

Requesting Parameters: **

- req
- accuracy
- pGPSInfo
- pcb

* Members of the AEEGPSConfig data structure used in IPOSDET_SetGPSConfig() to configure Assisted GPS operation.

** Parameters of IPOSDET_GetGPSInfo used when requesting each location fix.

“Mode” Configuration Parameter

“Mode” Values *	Description
One_Shot	<ul style="list-style-type: none"> • Performs a single position fix (default) • “Optim” parameter controls whether MS-Assisted or MS-Based used
Track_Network	<ul style="list-style-type: none"> • Performs a series of MS-Assisted fixes
Track_Local	<ul style="list-style-type: none"> • Performs a series of MS-Based fixes
Track_Optimal	<ul style="list-style-type: none"> • The gpsOne engine will use either MS-Based or MS-Assisted method, depending a specified optimization criteria, set with the “Optim” parameter • Provision for a fall back to the other fix method in certain conditions • Good for users who do not want to have to worry about all the details
Track_Standalone	<ul style="list-style-type: none"> • Performs a series of stand-alone fixes **
DLoad_First	<ul style="list-style-type: none"> • Requests assistance information without causing a fix to occur • Useful, for example, if going out of network coverage, or to cause the download of assistance data in preparation for MS-Based operation while app. welcome and introductory screens are shown

* Mode values shown without the preceding “AEEGPS_Mode_” for clarity.

** Stand-alone capability is available on the MSM6500 beginning with release 6.x, the MSM6550 beginning with release 4.x, and with the MSM6800 and all subsequent MSM releases.

“Optim” Configuration Parameter

- **Used to optimize the gpsOne engine to one specific goal, when used with the “Track_Optimal” and “One_Shot” modes**
 - Allows an app developer to direct the gpsOne™ engine to optimize to one specific goal without worrying about the subtleties – you set it then forget it
 - App developers are encouraged to experiment with these settings in their specific usage conditions to determine if this “set and forget” approach works for them
 - How gpsOne™ engines optimize to each goal may change as product capabilities evolve

“Optim” Values *	Description
None or Default	<ul style="list-style-type: none"> • No optimization is performed (default)
Speed	<ul style="list-style-type: none"> • Optimize for speed • A favorite setting used with the Track_Optimal mode – will start with MS-Based fixes then fall back to MS-Assisted if MS-Based fixes fail
Accuracy	<ul style="list-style-type: none"> • Optimize for accuracy • Typically causes MS-Assisted fixes
Payload	<ul style="list-style-type: none"> • Optimize to minimize usage of network data transmission resources

* Optim values shown without the preceding “AEEGPS_OPT_” for clarity.

“nFixes”, “nInterval”, “Server” and “QOS” Configuration Parameters

Parameter	Description
nFixes	<ul style="list-style-type: none"> • Estimated number of fixes to perform
nInterval	<ul style="list-style-type: none"> • Estimated interval in seconds between fixes
Server	<ul style="list-style-type: none"> • Defines the IP address and port number of the location server (PDE) *
QOS	<ul style="list-style-type: none"> • Maximum number of seconds the gpsOne™ search engine will take to search for satellites • Longer search times can help find a sufficient number of satellites, particularly in more challenging conditions (example: indoors) • QOS values between 8 and 31 are typical

* In most operator networks, the app is not allowed to control the server’s address for security reasons. Thus the Server type parameter is usually set to “AEEGPS_SERVER_DEFAULT”.

Calling Parameters

- Parameters used when calling IPOSET_GetGPSInfo:

Parameter	Description
req	<ul style="list-style-type: none"> Specifies the Request Type: Location, Velocity, <i>and/or</i> Altitude
accuracy	<ul style="list-style-type: none"> Selects the level of accuracy for this request from: <ul style="list-style-type: none"> – AEEGPS_ACCRACY_LEVEL1 (lowest and default) to – AEEGPS_ACCRACY_LEVEL6 (highest) Used in “fall back” scenarios (example: “Track_Optimal” mode with “Speed” optimization) <ul style="list-style-type: none"> – The fall back is triggered when fix uncertainty falls below a threshold value, determined by this parameter Not used for any other mode
pGPSInfo	<ul style="list-style-type: none"> Pointer to structure for returned fix data
pcb	<ul style="list-style-type: none"> Callback function which gets called on completion of the position determination

Example Scenario #1 – Sequence of Fixes, Consumed at the Mobile

- **Example application:**
 - Turn-by-turn navigation application
- **MS-Based best fits the needs of this application because:**
 - A series of rapid fixes are needed
 - Location is consumed at the mobile
- **Desired iPosDet settings for this example:**

Configuration Parameters:

- **Mode = Track_Local**
- **Optim = N/A**
- **nFixes = ***
- **nInterval = 1**
- **QOS = 10**

Requests a series of MS-Based fixes spaced every ~1 sec

Sufficient for typically vehicle-based use

Requesting Parameters:

- **accuracy = N/A**
- **req = Location**

Uses default Location Server

* Predicted number of fixes needed, based on the expected duration of the MS-Based session and the fix interval.

Example Scenario #2 – single, isolated fix

- **Example application:**
 - A single fix to find a “friend”, or an “asset” being tracked
 - Assume GPS conditions could be marginal
- **MS-Assisted best fits the needs of this application because:**
 - Only require a single fix, which is typically consumed by an entity outside of the mobile
 - Maximum GPS sensitivity and availability of fall-back solution types is desirable
- **Desired iPosDet settings:**

Configuration Parameters:

- **Mode = Track_Network**
- **Optim = N/A**
- **nFixes = 1**
- **nInterval = N/A**
- **QOS = 31**

} Requests a single, MS-Assisted fix

← Give gpsOne™ engine more time to find satellites to maximize GPS yield

Requesting Parameters:

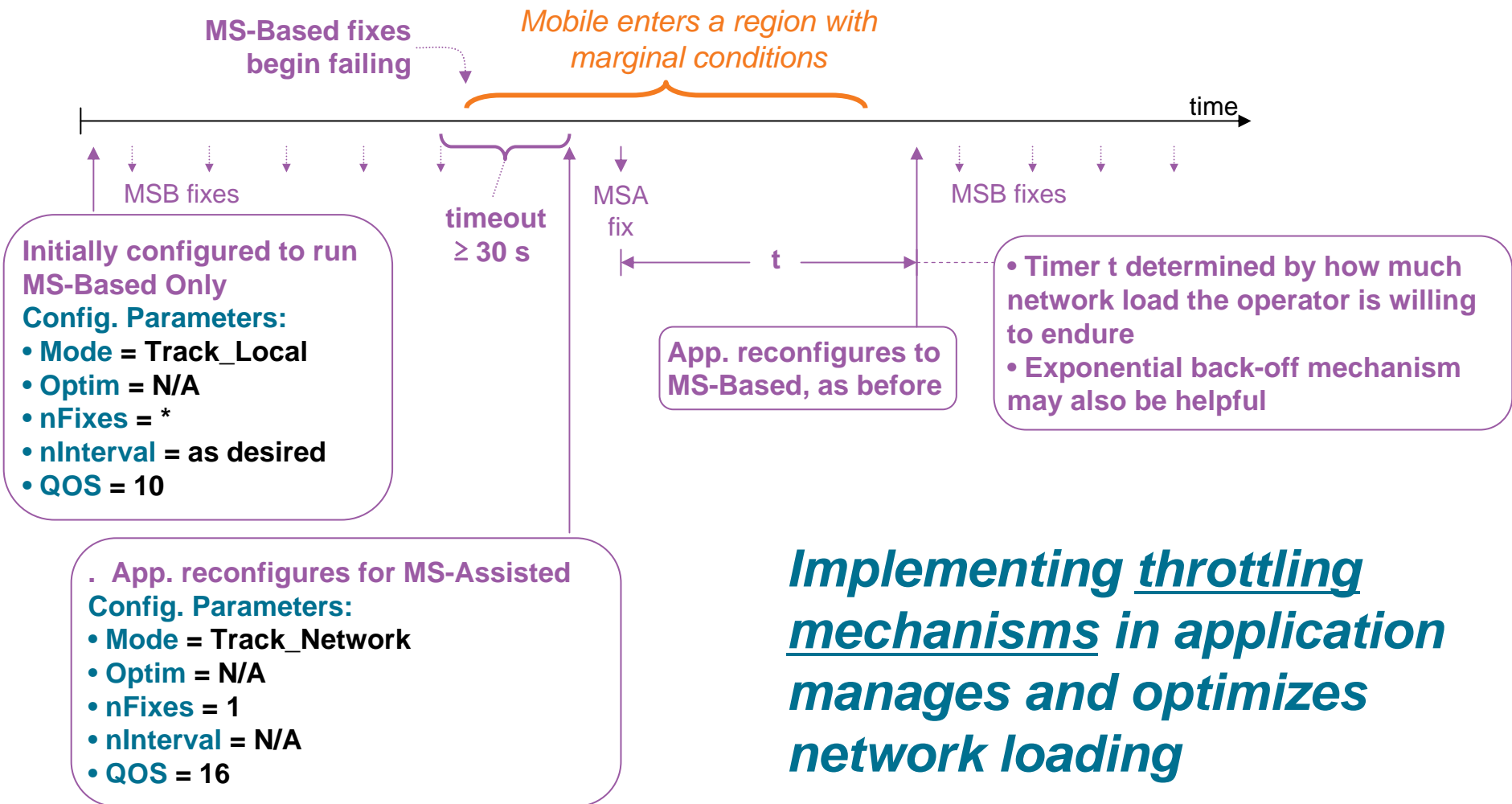
- **accuracy = N/A**
- **req = Location**

← Uses default Location Server

Example Scenario #3 – Application Controls Fallback Mechanisms to “Throttle” Data Usage

- **Example application:**
 - Geofencing application
 - Wireless Operator has requested special “**throttling algorithms**” to prevent excessive network utilization and prevent call originations beyond a desired rate
- **MS-Based, with controlled fallback to MS-Assisted, best fits the needs of this application because:**
 - A series of fixes are needed, even in challenging conditions
 - The customer is willing to sacrifice on fix update rate to minimize network load in challenging environments
 - Locations are processed at the mobile
- **Clearly, MS-Based is preferred where it works, but a carefully-controlled, customer-specified fall back to MS-Assisted is also required**
 - **Thus the application must implement the fall back mechanism manually**

Example Scenario #3 (continued)



* Predicted number of fixes needed, based on the expected duration of the MS-Based session and the fix interval.

Chronology of *iPosDet* Features

- **iPosDet has been continuously improved**
- **Use newer BREW versions to access all the features**

BREW version	gpsOne mode	Notes
BREW 1.x	Mobile-Assisted	Support limited to MS-Assisted gpsOne location data (latitude/longitude). <ul style="list-style-type: none"> • <i>IPOSEDET_GetSectorInfo()</i> • <i>IPOSEDET_GetPosition()</i>
BREW 2.0.x	Mobile-Assisted	Full support for MS-Assisted mode, plus a richer set of location data than that available in BREW 1.x (latitude, longitude, altitude, heading, horizontal velocity, vertical velocity, uncertainty). Added support for the following modes of operation <ul style="list-style-type: none"> • Single Shot, Track_Local, Track_Network, DLOAD_First
BREW 2.1.0 BREW 2.1.1	Mobile-Assisted	BREW 2.0 features plus added support for orientation (for handsets with integrated compass) and a query of modes supported on the handset. <ul style="list-style-type: none"> • Added <i>IPOSEDET_GetOrientation()</i>
BREW 2.1.2	Mobile-Assisted Mobile-Based	BREW 2.1.1 features plus MS-Based support and smart modes of operation which allow an application to optimize speed or accuracy when using gpsOne.
BREW 3.0	Mobile-Assisted Mobile-Based	BREW 2.1.2 features, plus a smart mode of operation which minimizes data exchange between a mobile and a PDE. <ul style="list-style-type: none"> • Added support for <i>IPOSEDET_ExtractPositionInfo()</i> • New mode : <i>Track_Optimal</i> • New optimization: <i>Optimal_Payload</i> • DLOAD_First enhancement to use nFixes*nInterval
BREW 3.1.2	Mobile-Assisted Mobile-Based Standalone	BREW 3.0 features, plus several new data elements to characterize returned location in more detail. Added Standalone mode of operation. <ul style="list-style-type: none"> • New mode : <i>Track_Standalone</i>
BREW 3.1.4	Mobile-Assisted Mobile-Based Standalone	BREW 3.1.2 plus addition of ability to read the E-911 Only / Location On Flag. (AEE_DeviceItem_PosDet_Emergency_Only Device Item using IShell_GetDeviceInfoEX)

iPosDet Outputs & Uncertainty

iPosDet Outputs

- iPosDet Outputs are generally well understood, with one exception: Uncertainty

Parameters related to Uncertainty

Members of AEEGPSInfo data structure

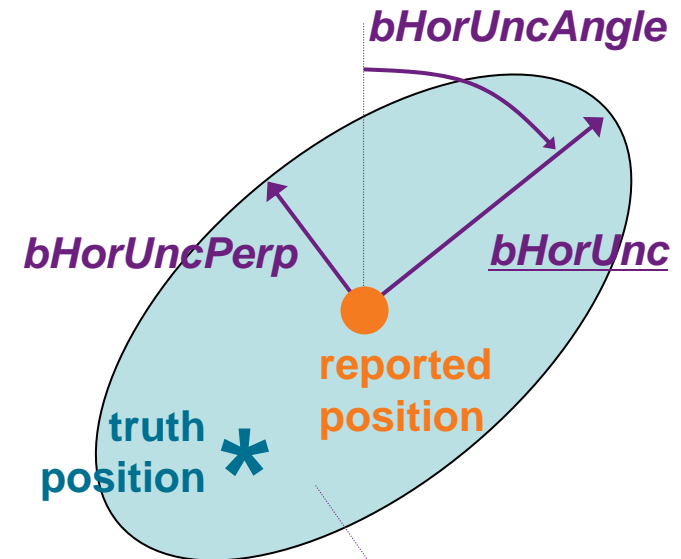
dwTimeStamp	Time of Fix
Status	Notification of any errors
dwLat	Latitude of fix
dwLon	Longitude of fix
wAltitude:	Altitude of fix
wHeading:	Heading of fix
wVelocityHor:	Horizontal Velocity
bVelocityVer:	Vertical Velocity
fValid:	Flags indicating valid fields in the structure
GPS_UTCOffset	Time difference between UTC and GPS time
method:	Fix method used
accuracy:	Not used
bHorUnc:	Semi-major of uncertainty ellipse
bHorUncAngle:	Angle of uncertainty ellipse
bHorUncPerp:	Semi-minor of uncertainty ellipse
bVerUnc:	Vertical uncertainty
LocProbability	Confidence that the true position falls within the uncertainty ellipse centered at the reported position

What is Uncertainty?

- **Generating location fixes is a *probabilistic* process, and the accuracy of each location fix *varies* from one fix to the next**
- **To help the recipients of location fixes – location based applications – understand how to interpret any given fix, all fix positions are delivered with an uncertainty**
- **This uncertainty helps the location based application know approximately how accurate the fix is believed to be, so it can interpret and utilize the fix properly**
- **Note that uncertainty is an estimate – a “guess”**
 - **Uncertainty is NOT the actual position error! To determine actual position error, it is necessary to know “ground truth”!**

Confidence and the Expression of Uncertainty

- For uncertainty to have meaning, it must be associated with a confidence factor (known as “*LocProbability*”)
- iPosDet expresses horizontal uncertainty as an ellipse
- Uncertainty and confidence factor are related and can be “traded off” one against the other
 - The smaller the confidence percentage, the shorter the uncertainty distance
 - The larger the confidence percentage, the longer the uncertainty distance
 - Mathematical relationships exist to translate an uncertainty with one confidence to a different confidence



Positioning system is “*LocProbability*”% confident that the true position falls within the ellipse

Recommendations for Use of Uncertainty

- Applications should **ALWAYS** determine if the intended operation makes sense given the reported uncertainty
- **Some Examples:**
 - Map scaling should not be inconsistent with the reported uncertainty – e.g. it makes no sense to zoom in on a position where the uncertainty is larger than the map scale
 - Apps should consider painting the actual uncertainty ellipse graphically when displaying maps
 - Avoid geocoding for larger uncertainties

**For More
Information**

For More Information

- **Take a class on Location Based Services from CDMA-University:**
 - <http://www.cdmauniversity.com/cdma/CourseDetails.aspx?CourseID=1146>
- **Information on gpsOne™ products and Qualcomm location technology:**
 - http://www.cdmatech.com/technologies/position_location.jsp
- **Information on QPoint™ location server products:**
 - <http://www.qualcomm.com/qis/qpoint/>
- **Information on BREW®:**
 - www.qualcomm.com/brew
- **Nice article on use of *iPosDet*:**
 - <http://www.devx.com/wireless/Article/29563>