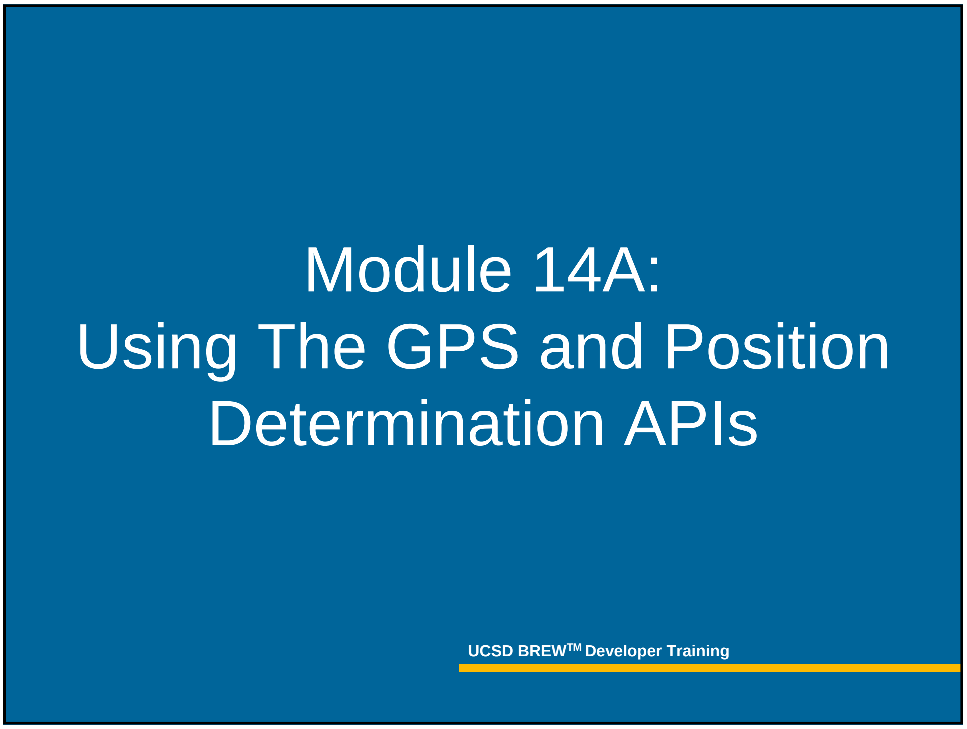
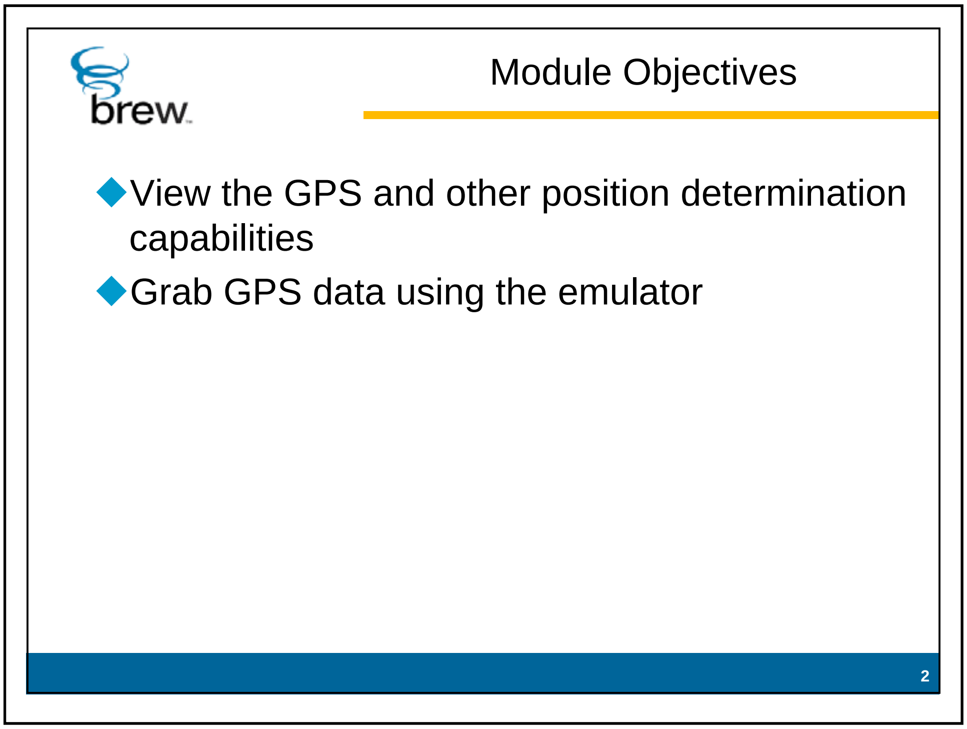


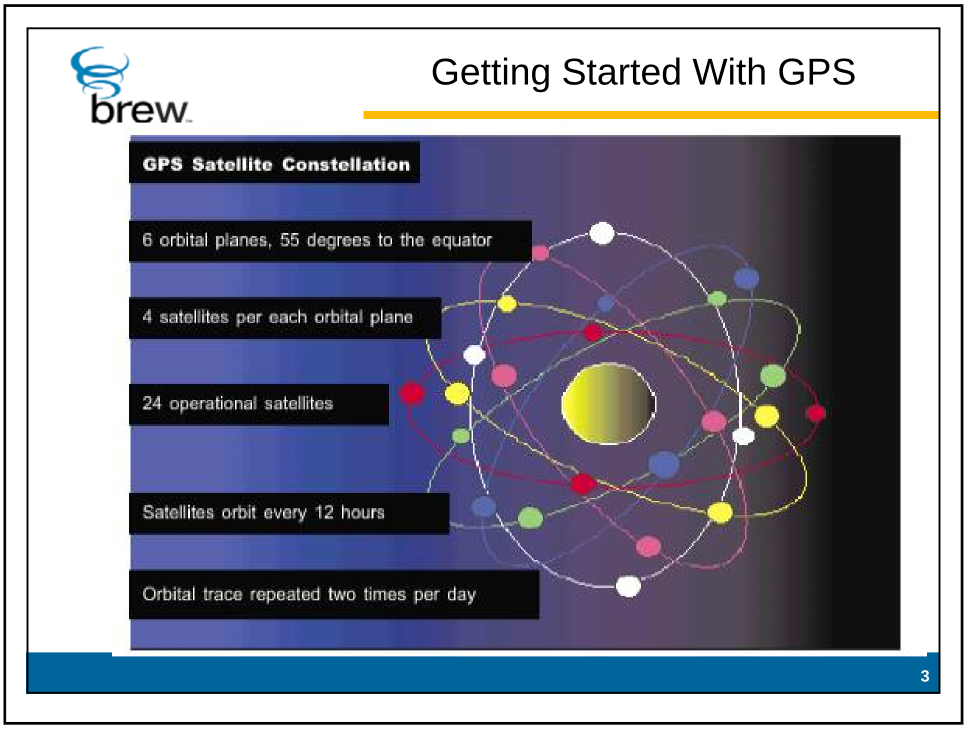
Module 14A: Using The GPS and Position Determination APIs

UCSD BREW^{TM} Developer Training



Module Objectives

- View the GPS and other position determination capabilities
- Grab GPS data using the emulator



Getting Started With GPS

GPS Satellite Constellation

6 orbital planes, 55 degrees to the equator

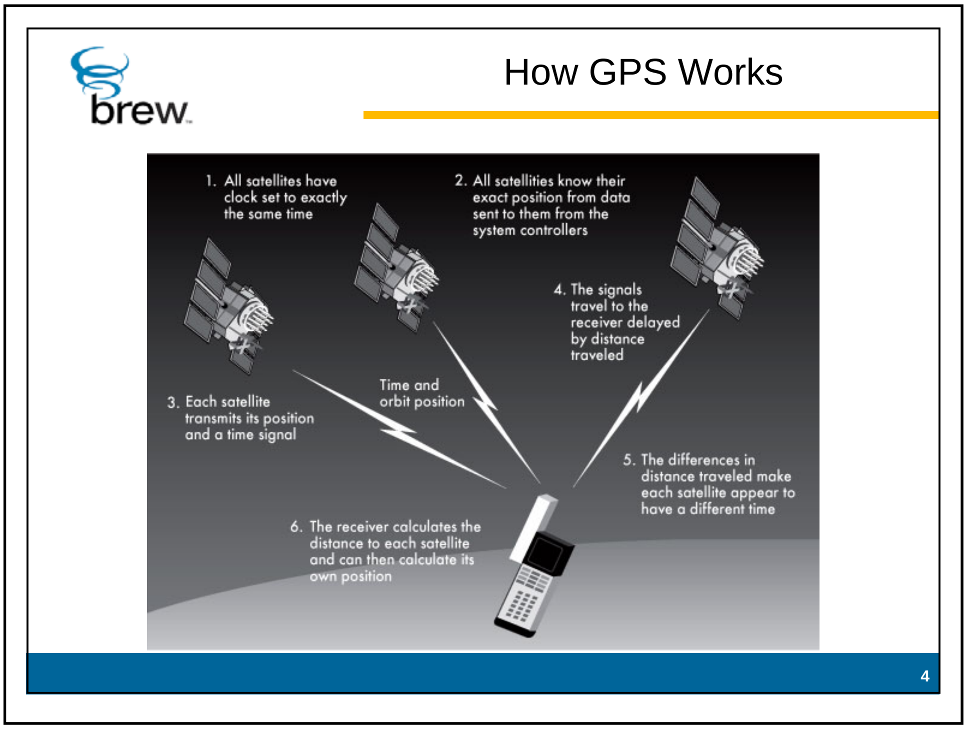
4 satellites per each orbital plane

24 operational satellites

Satellites orbit every 12 hours

Orbital trace repeated two times per day

3

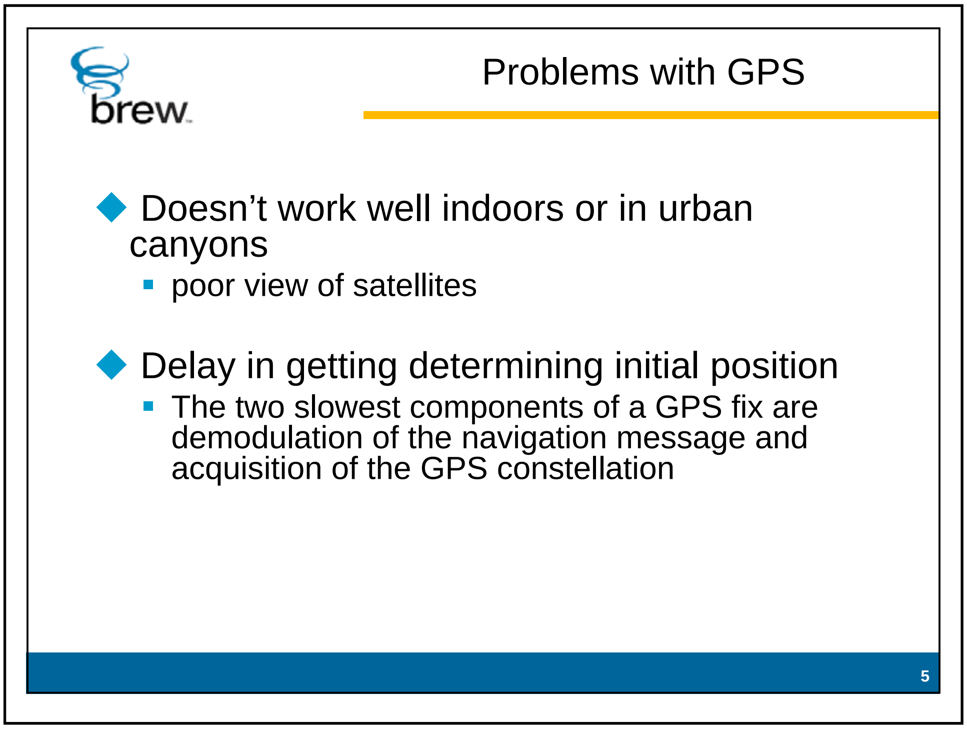


How GPS Works

1. All satellites have clock set to exactly the same time
2. All satellites know their exact position from data sent to them from the system controllers
3. Each satellite transmits its position and a time signal
4. The signals travel to the receiver delayed by distance traveled
5. The differences in distance traveled make each satellite appear to have a different time
6. The receiver calculates the distance to each satellite and can then calculate its own position

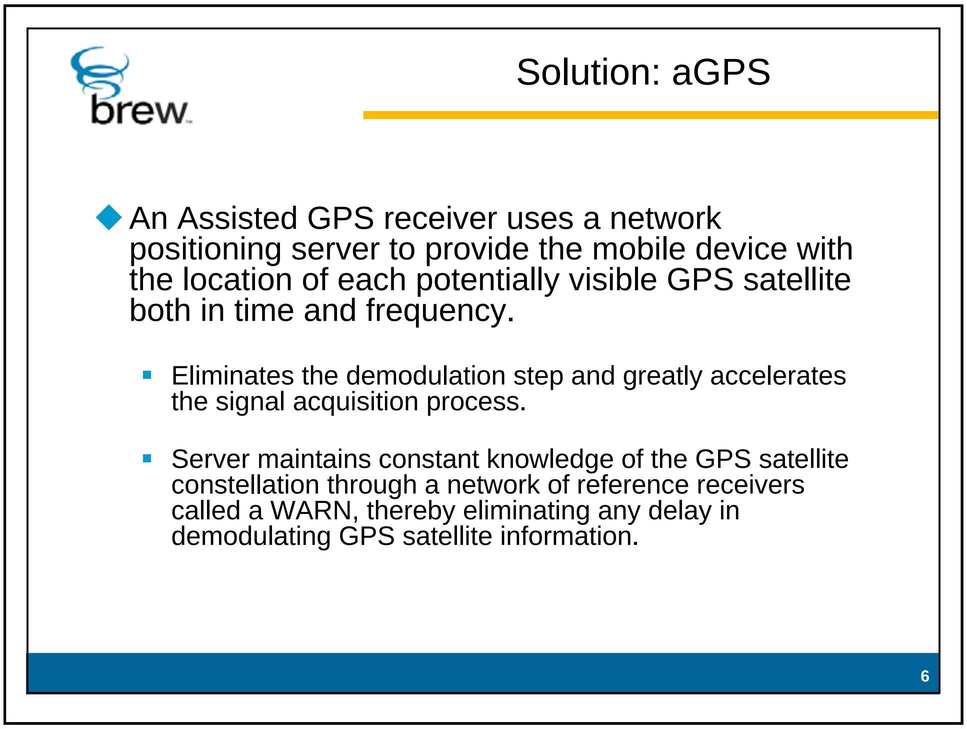
Time and orbit position

4



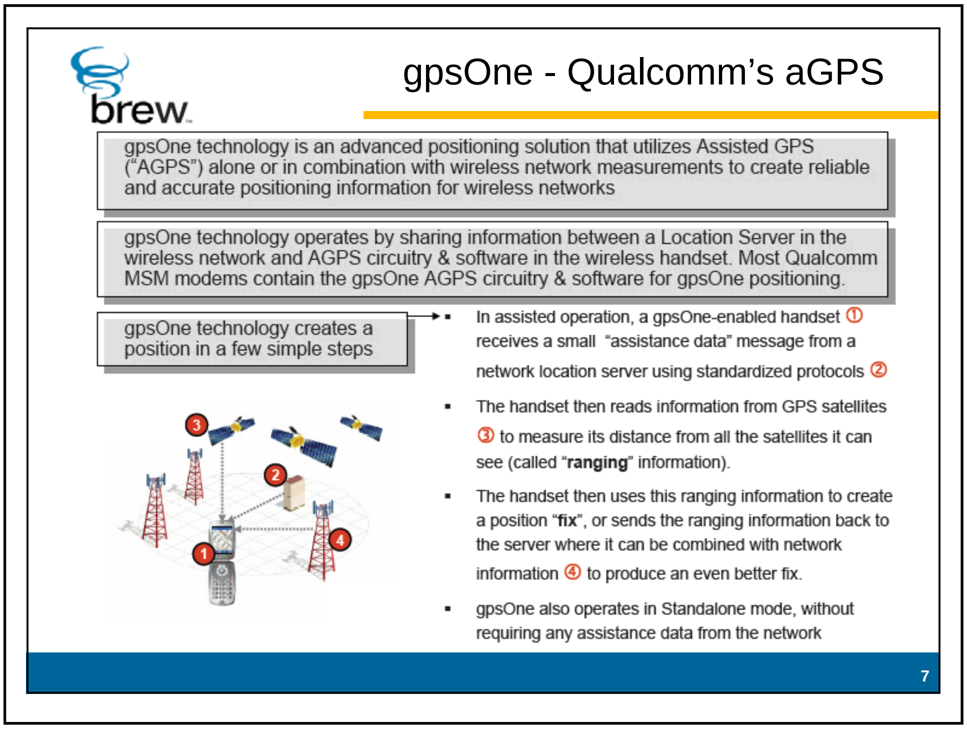
Problems with GPS

- Doesn’t work well indoors or in urban canyons
  - poor view of satellites
- Delay in getting determining initial position
  - The two slowest components of a GPS fix are demodulation of the navigation message and acquisition of the GPS constellation



Solution: aGPS

- An Assisted GPS receiver uses a network positioning server to provide the mobile device with the location of each potentially visible GPS satellite both in time and frequency.
  - Eliminates the demodulation step and greatly accelerates the signal acquisition process.
  - Server maintains constant knowledge of the GPS satellite constellation through a network of reference receivers called a WARN, thereby eliminating any delay in demodulating GPS satellite information.



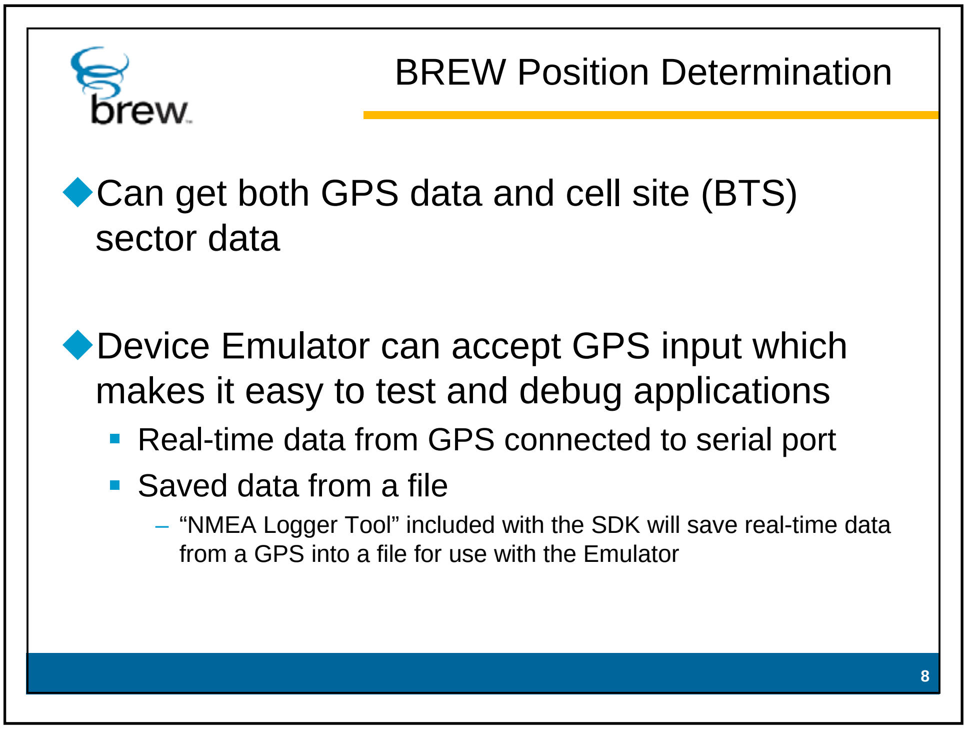
gpsOne - Qualcomm’s aGPS

gpsOne technology is an advanced positioning solution that utilizes Assisted GPS (“AGPS”) alone or in combination with wireless network measurements to create reliable and accurate positioning information for wireless networks

gpsOne technology operates by sharing information between a Location Server in the wireless network and AGPS circuitry & software in the wireless handset. Most Qualcomm MSM modems contain the gpsOne AGPS circuitry & software for gpsOne positioning.

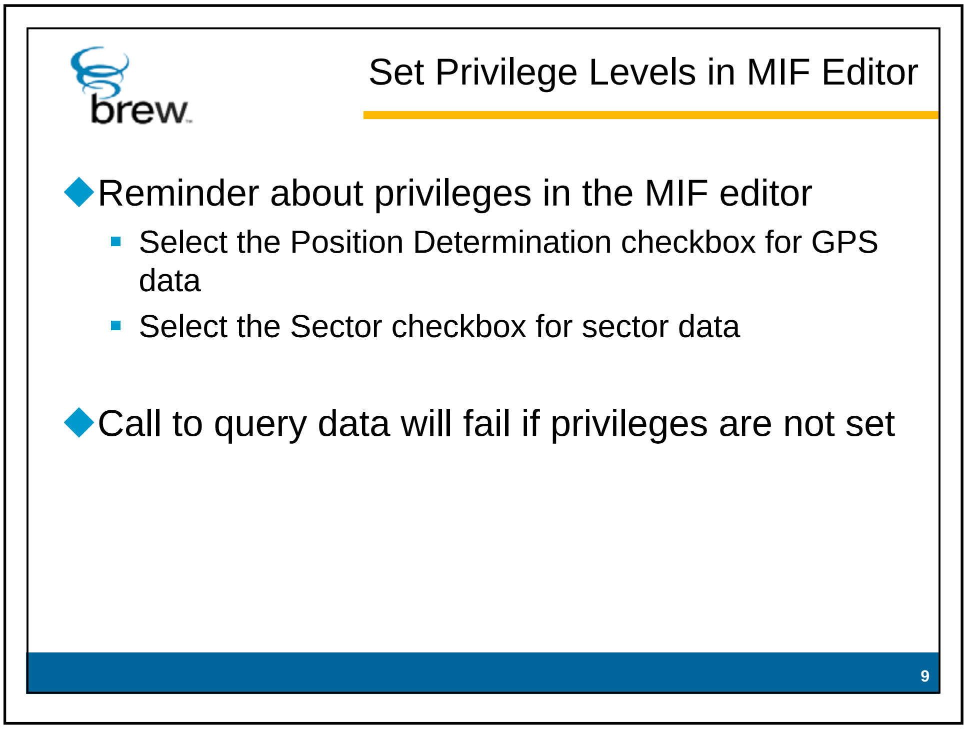
gpsOne technology creates a position in a few simple steps

- In assisted operation, a gpsOne-enabled handset ① receives a small “assistance data” message from a network location server using standardized protocols ②
- The handset then reads information from GPS satellites ③ to measure its distance from all the satellites it can see (called “**ranging**” information).
- The handset then uses this ranging information to create a position “**fix**”, or sends the ranging information back to the server where it can be combined with network information ④ to produce an even better fix.
- gpsOne also operates in Standalone mode, without requiring any assistance data from the network



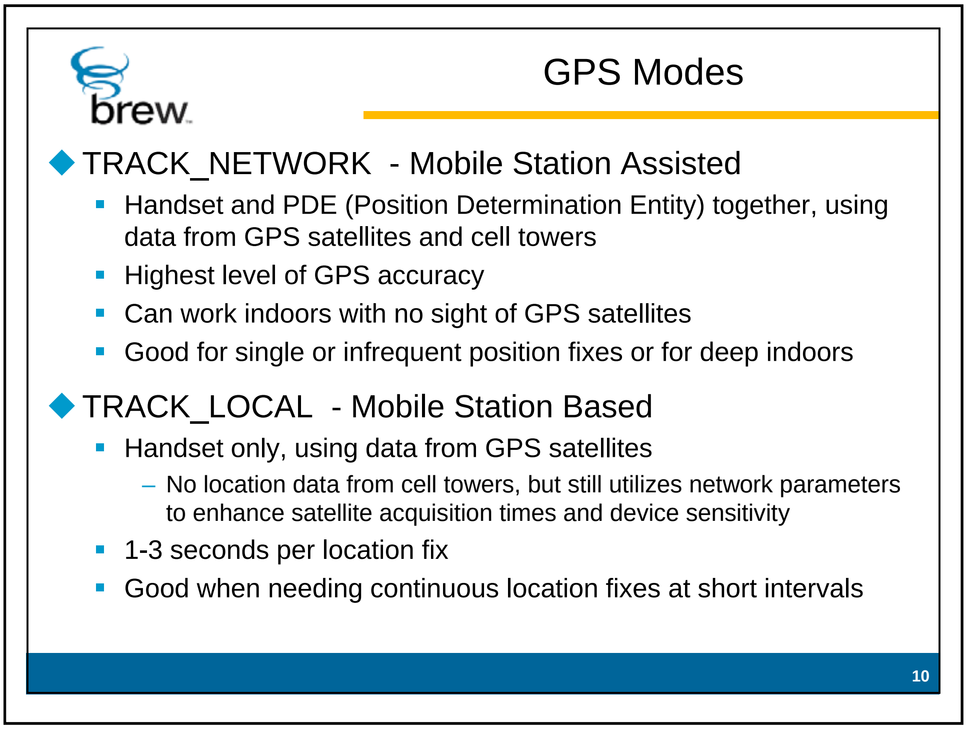
BREW Position Determination

- Can get both GPS data and cell site (BTS) sector data
- Device Emulator can accept GPS input which makes it easy to test and debug applications
  - Real-time data from GPS connected to serial port
  - Saved data from a file
    - “NMEA Logger Tool” included with the SDK will save real-time data from a GPS into a file for use with the Emulator



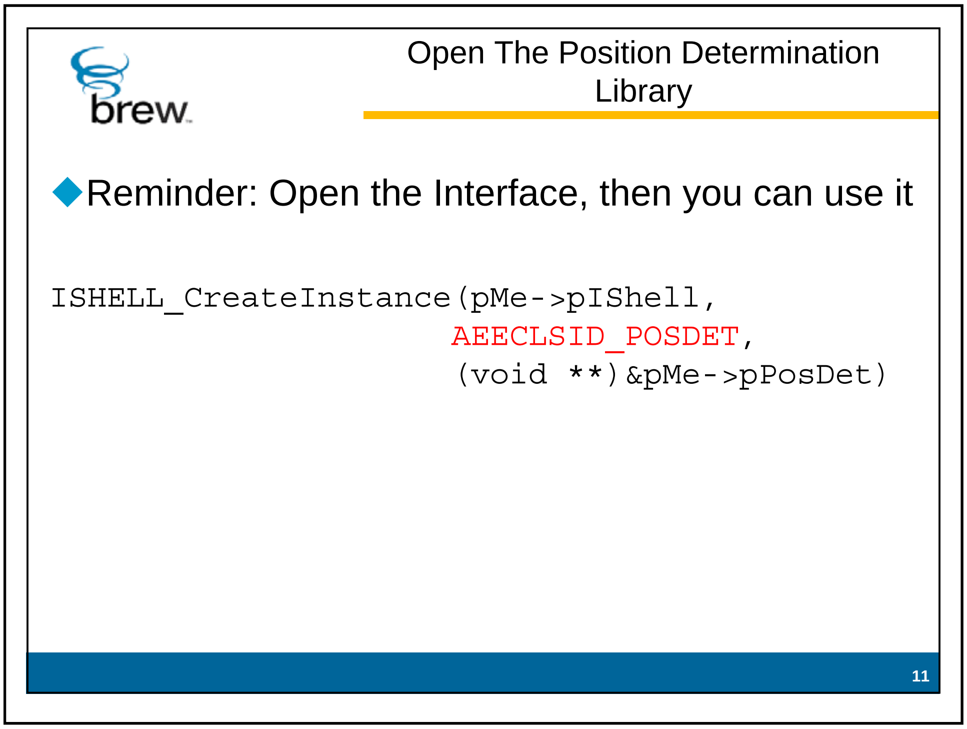
Set Privilege Levels in MIF Editor

- Reminder about privileges in the MIF editor
  - Select the Position Determination checkbox for GPS data
  - Select the Sector checkbox for sector data
- Call to query data will fail if privileges are not set



GPS Modes

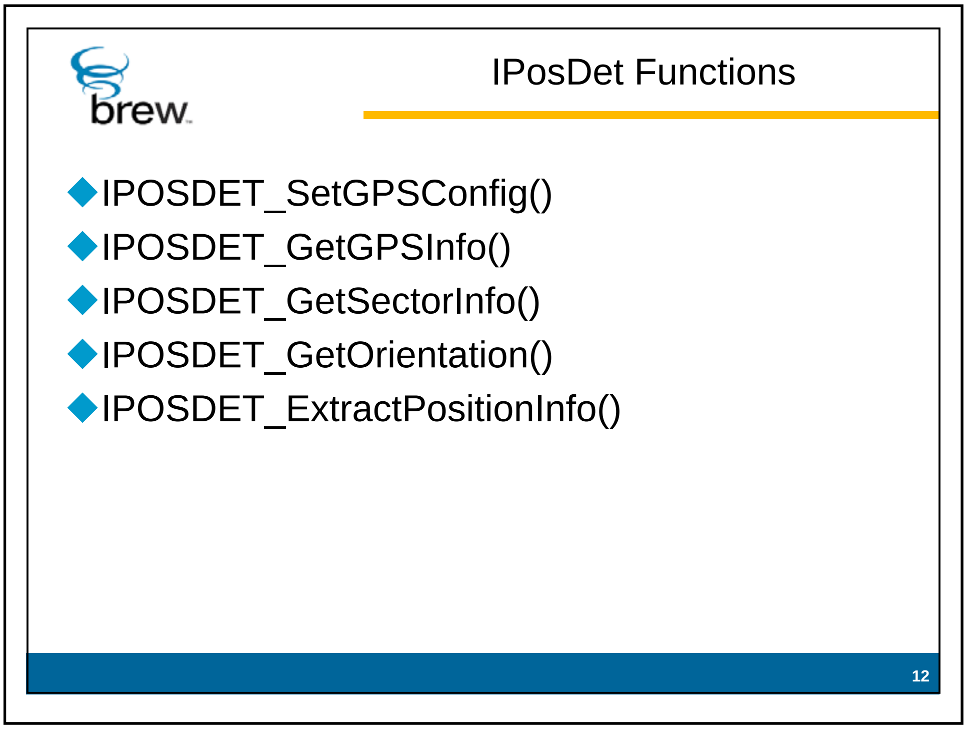
- TRACK_NETWORK - Mobile Station Assisted
  - Handset and PDE (Position Determination Entity) together, using data from GPS satellites and cell towers
  - Highest level of GPS accuracy
  - Can work indoors with no sight of GPS satellites
  - Good for single or infrequent position fixes or for deep indoors
- TRACK_LOCAL - Mobile Station Based
  - Handset only, using data from GPS satellites
    - No location data from cell towers, but still utilizes network parameters to enhance satellite acquisition times and device sensitivity
  - 1-3 seconds per location fix
  - Good when needing continuous location fixes at short intervals



Open The Position Determination Library

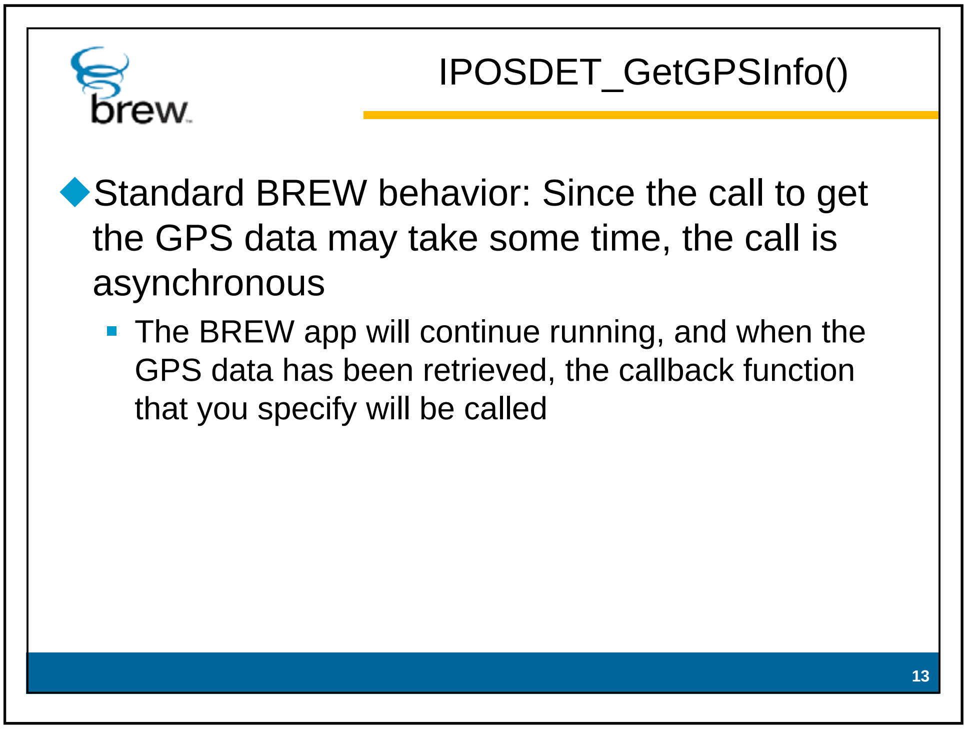
- Reminder: Open the Interface, then you can use it

```
ISHELL_CreateInstance(pMe->pIShell,
                      AEECLSID_POSDET,
                      (void **)&pMe->pPosDet)
```



IPosDet Functions

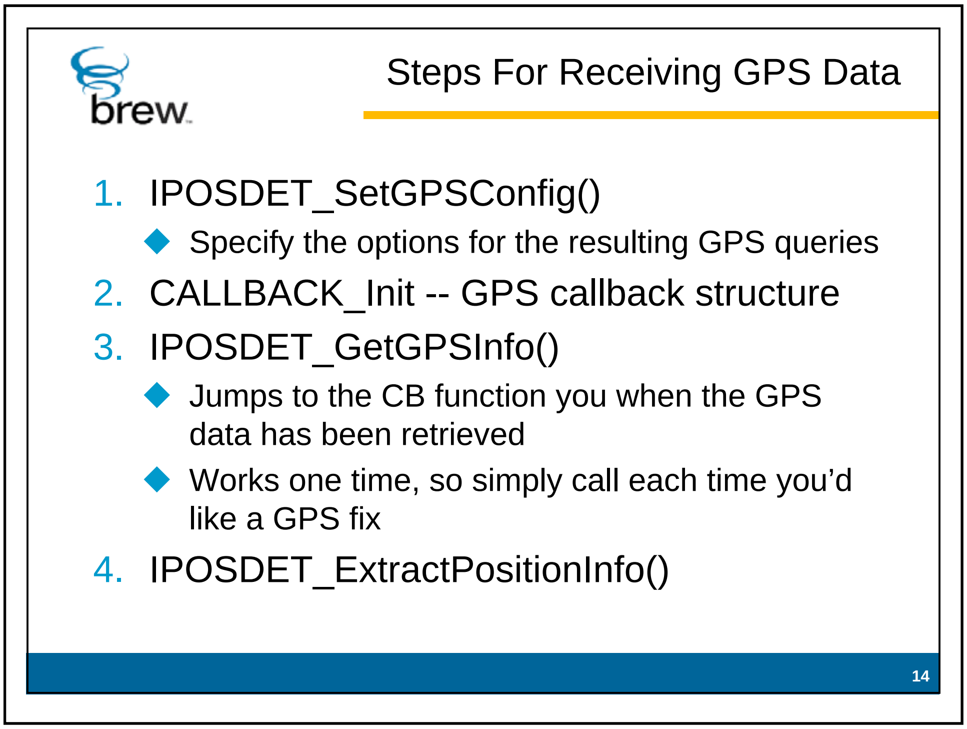
- IPOSDET_SetGPSConfig()
- IPOSDET_GetGPSInfo()
- IPOSDET_GetSectorInfo()
- IPOSDET_GetOrientation()
- IPOSDET_ExtractPositionInfo()



IPOSDET_GetGPSInfo()

- Standard BREW behavior: Since the call to get the GPS data may take some time, the call is asynchronous
  - The BREW app will continue running, and when the GPS data has been retrieved, the callback function that you specify will be called

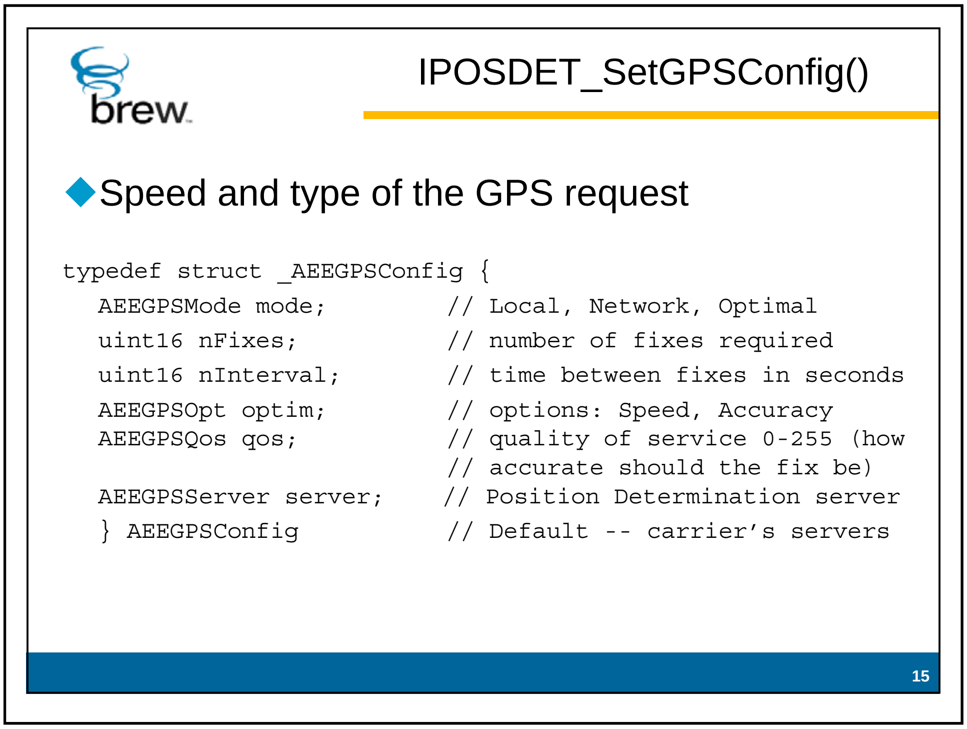
13



Steps For Receiving GPS Data

1. IPOSDET_SetGPSConfig()
   - Specify the options for the resulting GPS queries
2. CALLBACK_Init -- GPS callback structure
3. IPOSDET_GetGPSInfo()
   - Jumps to the CB function you when the GPS data has been retrieved
   - Works one time, so simply call each time you’d like a GPS fix
4. IPOSDET_ExtractPositionInfo()

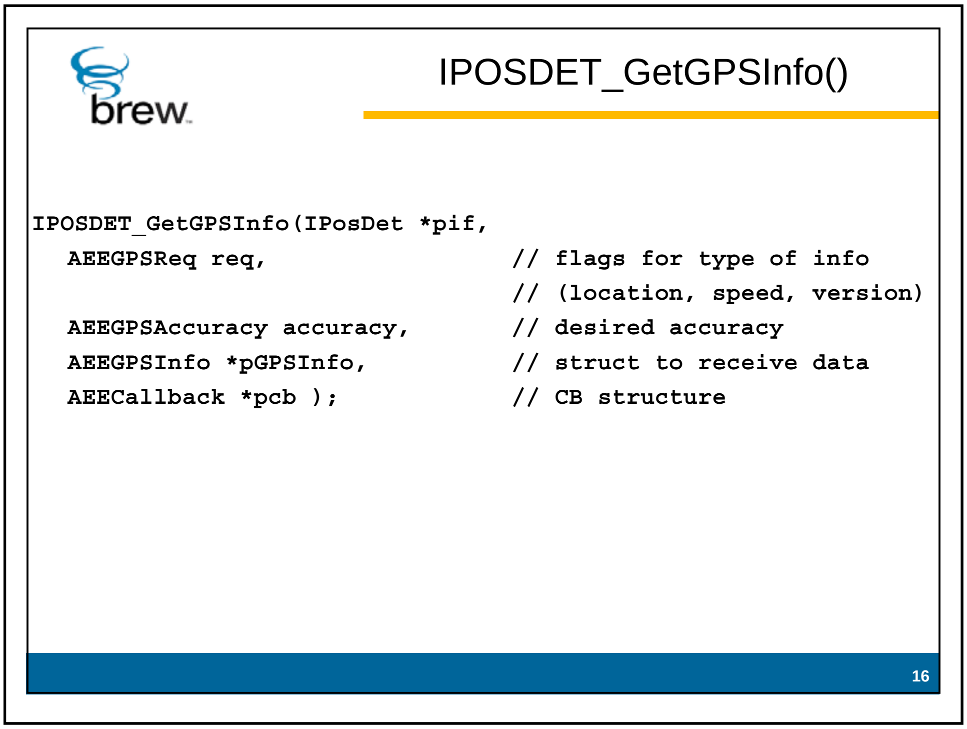
14



IPOSDET_SetGPSConfig()

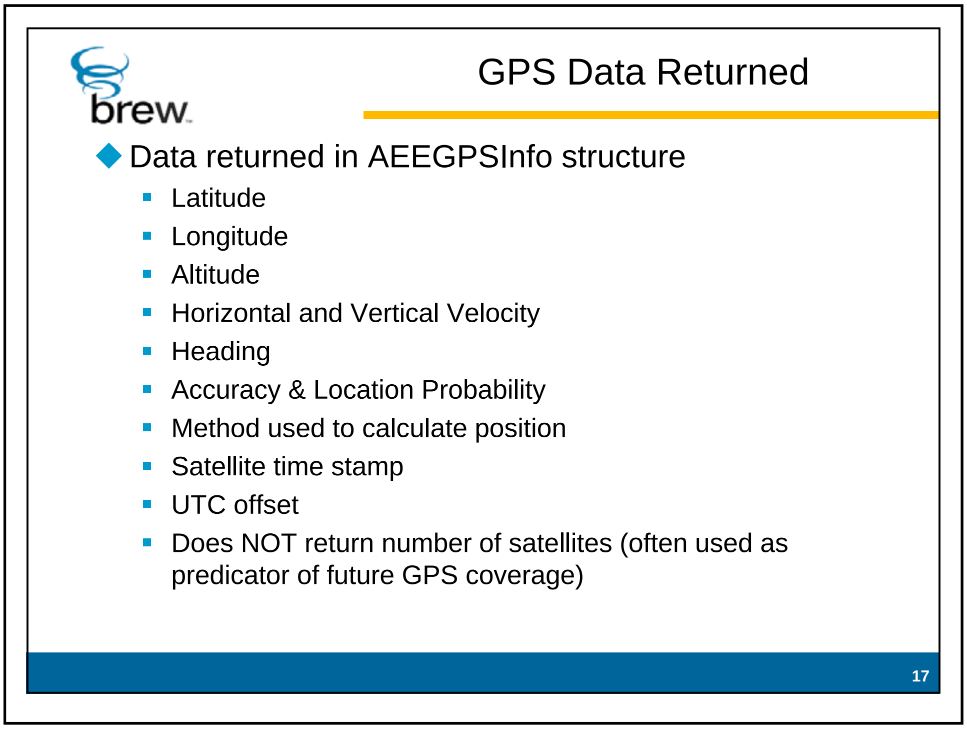
- Speed and type of the GPS request

```
typedef struct _AEEGPSConfig {
   AEEGPSMode mode;          // Local, Network, Optimal
   uint16 nFixes;            // number of fixes required
   uint16 nInterval;         // time between fixes in seconds
   AEEGPSOpt optim;          // options: Speed, Accuracy
   AEEGPSQos qos;            // quality of service 0-255 (how
                             // accurate should the fix be)
   AEEGPSServer server;     // Position Determination server
   } AEEGPSConfig            // Default -- carrier's servers
```



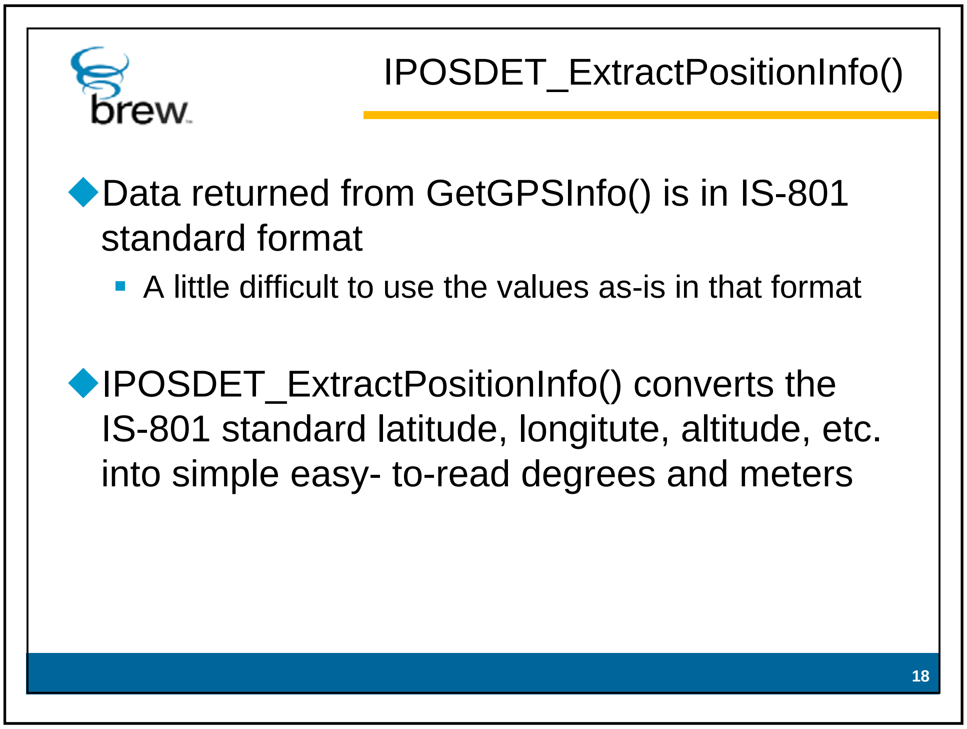
IPOSDET_GetGPSInfo()

```
IPOSDET_GetGPSInfo(IPosDet *pif,
   AEEGPSReq req,                   // flags for type of info
                                    // (location, speed, version)
   AEEGPSAccuracy accuracy,         // desired accuracy
   AEEGPSInfo *pGPSInfo,            // struct to receive data
   AEECallback *pcb );              // CB structure
```



GPS Data Returned

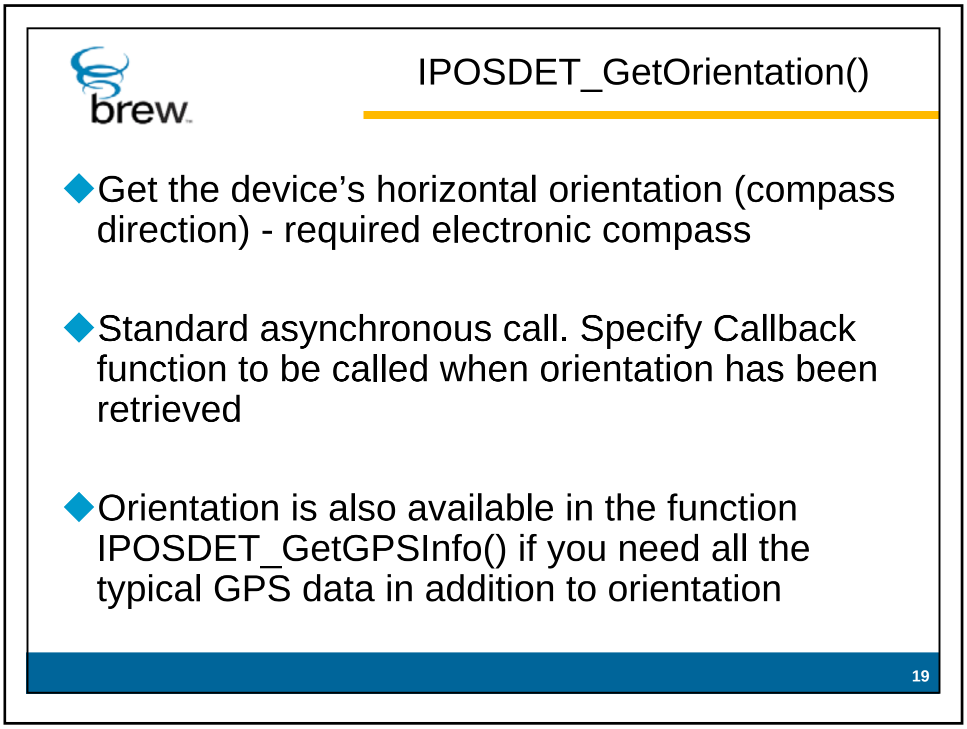
- Data returned in AEEGPSInfo structure
  - Latitude
  - Longitude
  - Altitude
  - Horizontal and Vertical Velocity
  - Heading
  - Accuracy & Location Probability
  - Method used to calculate position
  - Satellite time stamp
  - UTC offset
  - Does NOT return number of satellites (often used as predicator of future GPS coverage)



IPOSDET_ExtractPositionInfo()

- Data returned from GetGPSInfo() is in IS-801 standard format
  - A little difficult to use the values as-is in that format

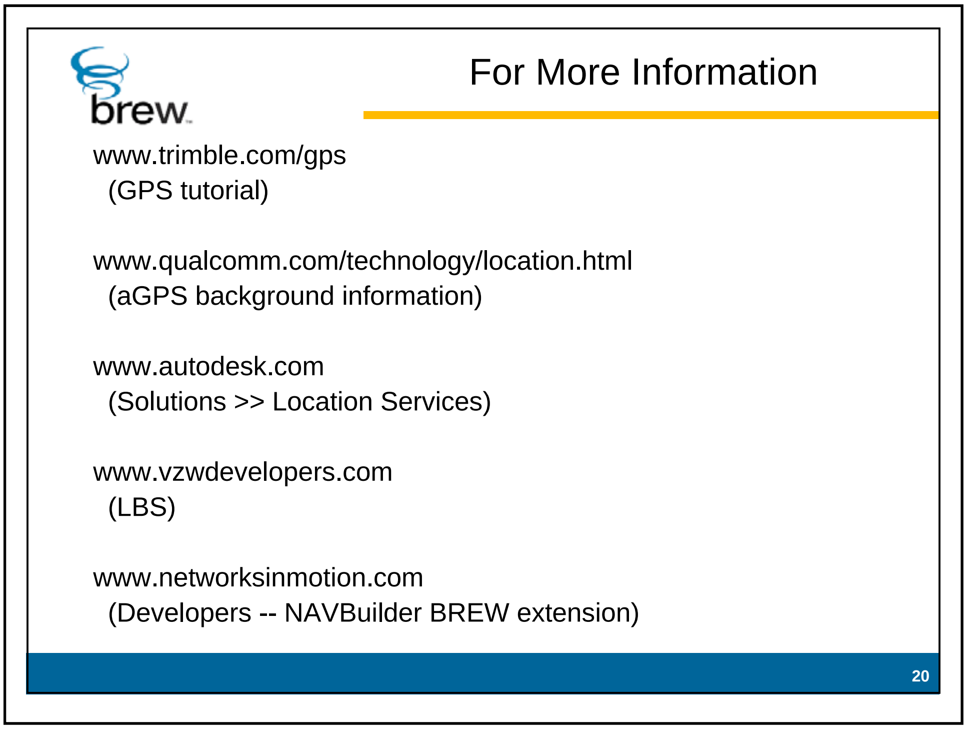
- IPOSDET_ExtractPositionInfo() converts the IS-801 standard latitude, longitute, altitude, etc. into simple easy- to-read degrees and meters



brew

IPOSDET_GetOrientation()

- Get the device’s horizontal orientation (compass direction) - required electronic compass
- Standard asynchronous call. Specify Callback function to be called when orientation has been retrieved
- Orientation is also available in the function IPOSDET_GetGPSInfo() if you need all the typical GPS data in addition to orientation



brew

For More Information

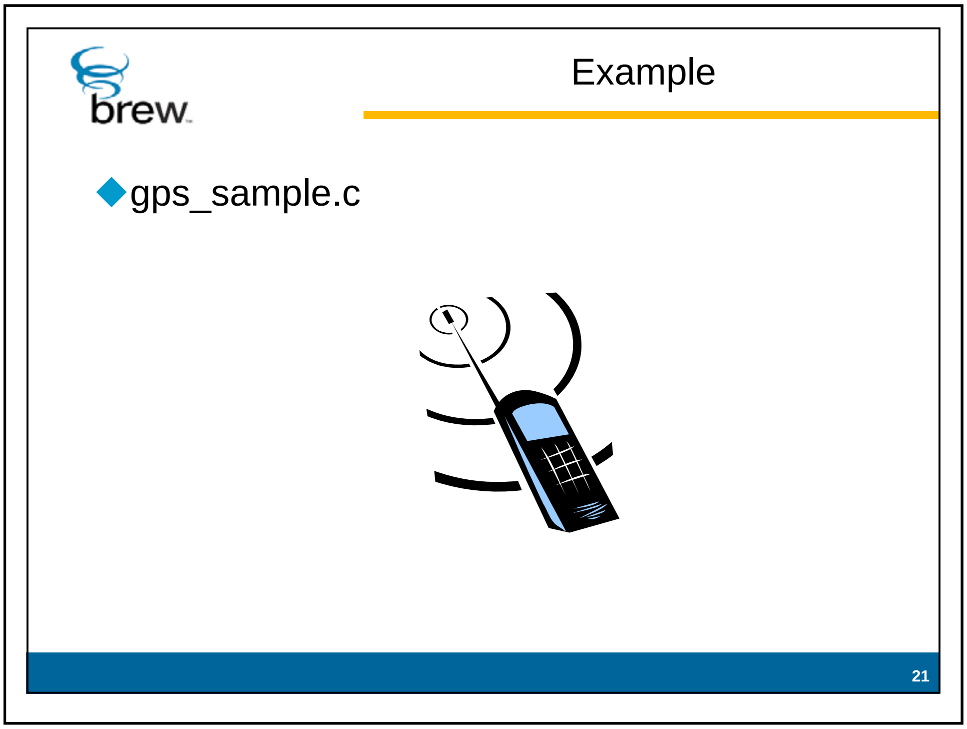
www.trimble.com/gps
(GPS tutorial)

www.qualcomm.com/technology/location.html
(aGPS background information)

www.autodesk.com
(Solutions >> Location Services)

www.vzwdevelopers.com
(LBS)

www.networksinmotion.com
(Developers -- NAVBuilder BREW extension)



Example

- gps_sample.c