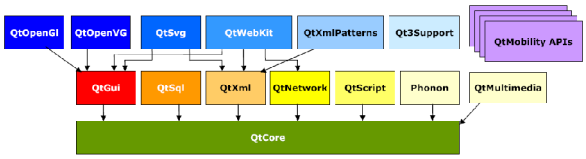
Qt Mobility

Qt Mobility

- New APIs added to allow access to the unique features of mobile devices
- First preview released 1st of December 2009
- First stable (1.0.0) released 27th of April, 2010
- Shipped in Nokia Qt SDK
- Current newest 1.2



Installing Qt Mobility on a Device

- N900
  - Qt Mobility API binaries are installed automatically when installing packages depending on them. One such package is the **qt-mobility-examples** package.
    - Install it though the application manager
  - Alternatively you may use apt-get
  - Current version from the app. manager 1.0.2
    - Use the *extras-devel* repository to get the 1.1 or 1.2 (but unless you specifically need something there stick to the current version from the app. manager)

Qt Mobility APIs

Qt Mobility 1.0.2

- **System Information**
  - Discover system related information and capabilities
- **Service Framework**
  - Discover and instantiate arbitrary services
- **Publish & Subscribe**
  - Read item values, navigate through and subscribe to change notifications
- **Messaging**
  - Messaging services, including SMS and email
- **Bearer Management** (has been integrated into QtNetwork in Qt 4.7)
  - Controlling the system's connectivity state
- **Contacts**
  - Enabling clients to request contact data from local and remote backends
- **Location**
  - Receiving location data using arbitrary data sources
- **Multimedia**
  - Play and record media, and manage a collection of media content
- **Sensor**
  - Accessing the acceleration, xyz-rotation and orientation of the device
- **Versit API**
  - API to exchange vCard data

Qt Mobility APIs

Qt Mobility 1.2

- **Document Gallery**
  - API to navigate and query documents using their meta-data
- **Feedback**
  - API enabling clients to control e.g. the vibration of the device
- **Organizer**
  - Access to calendar, schedule etc.
- **Camera**
  - Control and access to camera
- **Landmarks**
  - Creation and updating landmark information
- **Connectivity**
  - NFC and Bluetooth APIs

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

System Information API

- QSystemDeviceInfo
  - Device information (battery, power state, input method type, IMEI, manufacturer, profile status etc.)
- QSystemDisplayInfo
  - Display information (color depth, brightness)
- QSystemInfo
  - Various generation information (language, SW versions, etc.)
- QSystemNetworkInfo
  - Network information (network name, mode, signal strength, etc.)
- QSystemScreenSaver
  - Access to screen saver
- QSystemStorageInfo
  - Memory and disk information (disk types, free space, etc.)

Example using Mobility APIs

```
#include <QtGui/QApplication>
#include <QtGui/QLabel>
#include <QSystemInfo>

using namespace QtMobility;
int main( int argc, char *argv[] )
{
  QApplication app( argc, argv );
  QSystemInfo s;
  QLabel *label = new QLabel( "Current language is "+ s.currentLanguage() +
  " and you're using Qt " + s.version(QSystemInfo::QtCore) );
  label->show();
  return app.exec();
}
```

The QSystemInfo is defined in the #include <QSystemInfo> header

Example using Mobility APIs

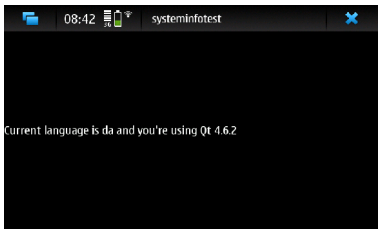
```
#include <QtGui/QApplication>
#include <QtGui/QLabel>
#include <QSystemInfo>

using namespace QtMobility;
int main( int argc, char *argv[] )
{
  QApplication app( argc, argv );
  QSystemInfo s;
  QLabel *label = new QLabel( "Current language is "+ s.currentLanguage() +
  " and you're using Qt " + s.version(QSystemInfo::QtCore) );
  label->show();
  return app.exec();
}
```

- The mobility APIs are defined in the QtMobility namespace. The using QtMobility makes the visible.
- Also the macro USE_QTM_NAMESPACE can be used

Updating the .pro file

```
CONFIG += mobility
MOBILITY += systeminfo
```



Mobility Modules

Each QtMobility API has its corresponding value which has to be added to **MOBILITY**.

The table lists the APIs and the corresponding values that can be assigned to **MOBILITY**.

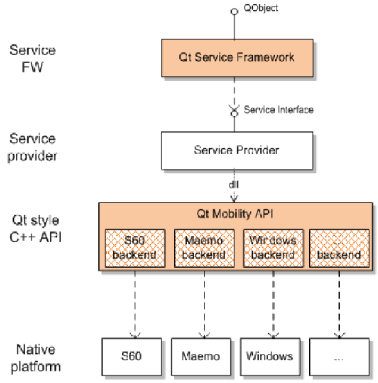
Domain	Value
Bearer Management	bearer
Contacts	contacts
Location	location
Multimedia	multimedia
Messaging	messaging
Publish And Subscribe	publishsubscribe
Service Framework	serviceframework
Sensors	sensors
System Information	systeminfo
Versit	versit
Document Gallery	gallery
Telephony Events	telephony
Organizer	organizer
Tactile Feedback	feedback

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Service Framework

- Uniform service / **plug-in handling** across multiple platforms
- Allows functionality reuse between application.
- Platform independent method for finding, using and implementing services



From Qt Mobility Whitepaper 1.0.1

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

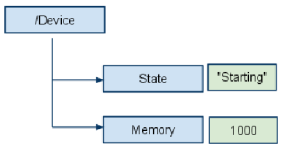
Publish & Subscribe

- Easy to use IPC (Inter Process Communication) mechanism.
- A *publisher* can use the API to make certain values available or to notify *subscribers* about changes.

- Main classes involved
  - QValueSpacePublisher
  - QValueSpaceSubscriber

Publish & Subscribe

Values and key are arranged in a directory like structure



Example:
- An incoming phone call
- IM messaging updates
- Battery status

Great concept for seperating responsability in our applications.
- Engine / UI
- MVC pattern

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Messaging

- Access to SMS, MMS, Email, instant messaging capabilities

- Composition and manipulation of messages:
  - QMessage
  - QMessageAddress
- Accessing message accounts
  - QMessageAccount
  - QMessageFolder
- Sorting and filtering
  - QMessageStore
  - QMessageFilter
- Accessing message services
  - QMessageService

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Bearer Management

In Qt 4.7

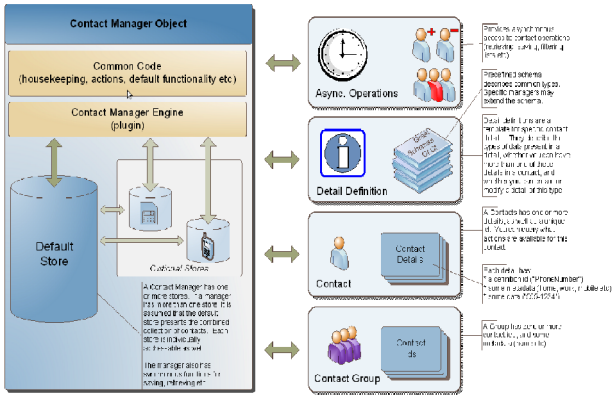
- Manages the connectivity state to the network
- Allows the user to start or stop network interfaces
- Info on if the device is online and how many available interfaces there are
- Can support automatic roaming between cellular and WLAN networks

- QNetworkConfigurationManager
  - Access configuration and monitor state
- QNetworkConfiguration
  - Represents a specific network configuration for a specific network interface. (Note several configurations may exist for a single interface).
- QNetworkSession
  - Control over system's access points. Start and stop access points based on a specific configuration.

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Contacts



From Qt Mobility Whitepaper 1.0.1

Contacts

Get phone number:

```
QContactManager cm; // instantiate the default manager
QList<QContact> allContacts = cm.contacts();
QContact firstContact = allContacts.first();
qDebug() << "The first contact has a phone number:" << firstContact.detail<QContactPhoneNumber>().number();
```

Save detail:

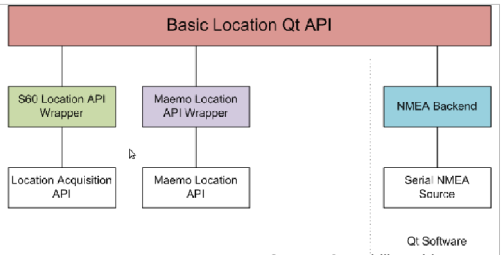
```
QContactPhoneNumber newPhoneNumber; // create the detail to add
newPhoneNumber.setNumber("12345"); // set the value(s) to save
firstContact.saveDetail(&newPhoneNumber); // save the detail in the contact
cm.saveContact(&firstContact); // save the contact in the manager
cm.removeContact(firstContact.localId()); // remove the contact from the manager
```

Source and more examples: http://doc.qt.nokia.com/qtmobility-1.2/samplephonebook.html

Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Location



Source: Qt Mobility White Paper

- The classes in the API consist of containers for the positional data and classes that manage the sources of the data

Location

Source: Forum Nokia Wiki

```
#include <QGeoPositionInfo>
#include <QGeoPositionInfoSource>

// Neccessary for Qt Mobility API usage
QTM_USE_NAMESPACE

class LocationInfo : public QObject
{
Q_OBJECT

public:

LocationInfo(QObject* parent = 0) : QObject(parent)
{
QGeoPositionInfoSource* src = QGeoPositionInfoSource::createDefaultSource(this);
if (src)
{
connect(src, SIGNAL(positionUpdated(QGeoPositionInfo)), this,
SLOT(updatePosition(QGeoPositionInfo)));
connect(src, SIGNAL(updateTimeout()), this, SLOT(updateTimeout()));
src->requestUpdate(5000); // Start request for actual position
}
}

private slots:

void updatePosition(const QGeoPositionInfo& info)
{
qDebug() << "Current position: " << info;
}

void updateTimeout()
{
// Current location could not be retrieved within the specified timeout of 5 seconds.
qWarning("Failed to retrieve current position.");
}
};
```


Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Multimedia

- Playing audio & video of various formats
- Recording audio
- Playing and managing of an FM radio
- QtMultimedia will eventually replace Phonon API
- Access of multimedia services with minimal code and maximal flexibility

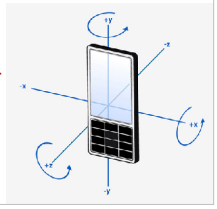
Qt Mobility APIs

- System Information
  - Discover system related information and capabilities
- Service Framework
  - Discover and instantiate arbitrary services
- Publish & Subscribe
  - Read item values, navigate through and subscribe to change notifications
- Messaging
  - Messaging services, including SMS and email
- Bearer Management
  - Controlling the system's connectivity state
- Contacts
  - Enabling clients to request contact data from local and remote backends
- Location
  - Receiving location data using arbitrary data sources
- Multimedia
  - Play and record media, and manage a collection of media content
- Sensor
  - Accessing the acceleration, xyz-rotation and orientation of the device

Sensor API

- The API can be used to poll sensors for data, or for the sensors to push data as they arrive
- QSensor derived classes provide access to input from various sensor:

QAmbientLightSensor
QAccelerometer
QCompass
QMagnetometer
QOrientationSensor
QProximitySensor
QRotationSensor
QTapSensor



Sensor API

3 steps to start using it

```
m_accelerometer = new QAccelerometer(this);
connect(m_accelerometer, SIGNAL(readingChanged()), this, SLOT(readingChanged()));
```

```
m_accelerometer->start();
```

```
QAccelerometerReading *r = m_accelerometer->reading();

qreal x = r->x();
qreal y = r->y();
qreal z = r->z();

ui->xvalue->setText(tr("%1").arg(x));
ui->yvalue->setText(tr("%1").arg(y));
ui->zvalue->setText(tr("%1").arg(z));
```

Platform Compatibility

Color	Explanation
	A functional backend for the API on the platform is complete.
	A functional backend for the API on the platform is being worked on, however it is not functionally complete.
	A functional backend for the API on the platform is being worked on. At this point it is far from functionally complete or there is no platform specific code inside QDF source code.
	A functional backend for the API on the platform is not being worked on. It is possible for others to implement and integrate support. Alternatively, the platform does not support the feature.

	API Maturity Level	Tier 1 Platforms							Tier 2 Platforms		
		S60 3rd Edition, feature Pack 1	S60 3rd Edition, feature Pack 2	S60 5th Edition	Symbian	Maemo 5	Harmattan	MeeGo	Windows XP/Vista	Linux	Mac OS X
Service Framework	FINAL										
Messaging	FINAL										
Device Management	FINAL							1)			
Publish and Subscribe	FINAL										
Contacts	FINAL										
Location	FINAL										
Multimedia	FINAL										
System Information	FINAL										
Sensors	FINAL										
Versit(vCard)	FINAL										
Versit(Organizer)	FINAL										
Camera	FINAL										
Organizer	FINAL										
Landmarks	FINAL										
Document Gallery	FINAL										
Maps/Navigation	FINAL										
Feedback	FINAL	1)	1)	2)							
Connectivity (Bluetooth)	BETA										
Connectivity (NFC)	BETA										

Source: http://doc.qt.nokia.com/qtmobility-1.2

Document Gallery

- Access to files on the device
  - E.g. images, audio, video and documents
  - Read/update meta-data for the found files e.g. reading embedded thumbnails
- Uses the platforms file-indexing service

Feedback

- Tactile and audio feedback on certain events
  - Theme defined feedback or custom defined feedback

```
QFeedbackEffect::playThemeEffect(QFeedbackEffect::ThemeBasicButton);
```


Organizer

The Organizer API enables a client to request calendar, schedule and personal data from local or remote backends.

- Add, remove update todos , events etc.