Qt default user agent gets rejected (Known Issue)

Overview

An online source for image elements cannot be retrieved from certain 3G network providers. This issue first occurred with SFR France.

Description

For example, a basic Qt Quick application loading an image from the network systematically receives the following message:

```
>>Error downloading http:/xxxxxxxx.jpg - server replied:Forbidden >>
```

Steps to reproduce

1. Create a Qt Quick application.
2. Set the Image element's source to an external image:

```
    Image{
         id:newsImg
         width:100
         height:80
         source:"http://qt.nokia.com/logo.png"
      }
```

3. Install an SFR SIM card in your device.
4. Deactivate all Wi-Fi access points on your device to make sure you'll use SFR IAP.
5. Build and run the application on a Symbian^3 device.

Expected result

The image is displayed.

Actual result

The image is **not** displayed.

Solution

The origin of the problem seems to be related to user agent filtering on operator network side: the default user agent used by the Qt HTTP stack is rejected by the network. The workaround solution consists of setting an explicit user agent (for instance the same user agent as the one used by the Qt webkit). Here is how to fix it :

1. Define a helper class which inherits **QWebPage** as follows:

```
class UserAgentProvider : public QWebPage {
    Q_OBJECT
public:
    explicit UserAgentProvider(QWidget *parent = 0);
    QString getUserAgent()
    {
        // set your UA string here, or return the one defined by Qt Webkit
        // recommended to have minimum "Mozilla/5.0 (Nokia; Qt; Symbian)"
        return userAgentForUrl(QUrl(""));
    }
};
```


(C) Copyright Nokia 2014. All rights reserved.

2. Define a custom factory class which inherits **QDeclarativeNetworkAccessManagerFactory** as follows:

```
class NetworkAccessManagerFactory : public QDeclarativeNetworkAccessManagerFactory
{
public:
   explicit NetworkAccessManagerFactory(QString p_userAgent = "");
   QNetworkAccessManager* create(QObject* parent)
   {
       CustomNetworkAccessManager* manager = new
CustomNetworkAccessManager(__userAgent, parent);
       return manager;
   }
private:
   QString __userAgent;
};
```

3. Define a custom network class which inherits **QNetworkAccessManager** as follows:

```
class CustomNetworkAccessManager : public QNetworkAccessManager {
   Q_OBJECT
public:
   explicit CustomNetworkAccessManager(QString p_userAgent = "", QObject *parent = 0);
protected:
   QNetworkReply *createRequest( Operation op, const QNetworkRequest &req, QIODevice *
outgoingData=0 )
   {

       QNetworkRequest new_req(req);
       new_req.setRawHeader("User-Agent", __userAgent.toAscii());
       QNetworkReply *reply = QNetworkAccessManager::createRequest( op, new_req,
outgoingData );
       return reply;
   }
private:
   QString __userAgent;
};
```

4. Modify **main.cpp** as follows:

```
// Set custom NetworkAccessManagerFactory object in QDeclarative context
UserAgentProvider p;
QString userAgent = p.getUserAgent();

NetworkAccessManagerFactory factory(userAgent);
viewer.engine()->setNetworkAccessManagerFactory(&factory);
```

Attached is a sample Qt project (NetworkOverload) which does the job. Note : The problem also occurs with Qt C++ applications (not using QML). Here the fix is slightly simpler since subclassing **QDeclarativeNetworkAccessManagerFactory** is not required (steps 2 and 4 can be removed).

Media: NetworkOverload.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.