

Symbian C++应用软件测试指南

版本 1.1; 2005 年 9 月 28 日

Testing

NOKIA

版权©诺基亚公司 20055。版权所有。

Nokia 和 Nokia Connecting People 是诺基亚公司的注册商标。Java 以及基于 Java 的商标是 Sun Microsystems 公司的注册商标。本文中提到的其它产品和公司名称可能是其相应公司的商标或商号。

否认声明:

本文内容按“现状”(as is)提供,即没有任何形式的保证,包括对产品可销售、适合特定目的以及其它由本文任何建议、规范和范例衍生出来的任何保证。另外,本文提供的信息是初级的,因此在最终版本确定之前其可能有很大改动。本文目的仅是提供信息参考。

诺基亚公司不承诺承担任何责任,包括对任何所有权的侵害责任,尽管这些所有权与实施本文给出的内容有关。诺基亚公司不保证或声称使用本文内容不会侵害上述所有权。

诺基亚保留对本文,在未经事先通知的情况下,随时进行变更的权力。

许可声明:

允许对本文进行仅用于个人使用目的的下载和打印。在此没有许可任何其它知识产权。

目录

1	简介	5
1.1	目的与范围	5
2	测试要求	6
2.1	对开发人员的基本硬件和软件要求	6
2.1.1	对Series 60 Platform 3 rd Edition的要求	6
2.2	区域测试	6
2.3	商用测试计划	6
3	基本要求	7
3.1	非技术问题	7
3.2	应用软件的性能和可靠性	7
3.3	应用软件的安全性	7
3.4	打包与安装	8
3.5	卸载与重新安装	9
4	功能测试	10
4.1	环境变化	10
4.2	网络连接	10
4.3	暂停应用软件	12
4.4	情景模式	12
5	用户界面测试	15
5.1	可用性基础	15
5.2	用户界面	16
5.2.1	菜单	16
5.2.2	按键	16
5.2.3	声音	17
5.2.4	主题	17
6	内存	18
7	本地化	19
8	Symbian C++应用软件中的典型错误	20
9	术语和缩略语	21
10	参考资料	22
11	文档评估	23

修订记录

2005 年 4 月 25 日	V.1.0	取代文档 <i>Series 60 Developer Platform: C++ 应用软件测试指南 v.1.0</i> (<i>Series 60 Developer Platform: Guidelines for Testing C++ Applications v.1.0</i>)
2005 年 9 月 28 日	V.1.1	修改文档名称。增加 2.1.1 节。

1 简介

1.1 目的与范围

本文提供了针对 **Series 60**、**Series 80**、诺基亚 **7710** 设备，设计和开发本地 **Symbian OS C++** 应用软件的指南，同时提供了一些信息，这些信息有助于开发人员满足测试人员的要求，使他们的应用软件符合诺基亚的建议和指南。本文还将帮助开发人员满足 **Symbian Signed** 标准。开发人员可以使用本文作为通过 **Symbian Signed** 测试的检查列表。

本文涉及 **C++** 应用软件的基本要求、功能性以及用户界面。同时包括内存和本地化问题。开发人员如果需要有关软件测试的其它一般信息，请参考市面上可以买到的书籍。

开发人员应该注意，对于某些开发平台的设备可能存在特殊问题。因此，开发人员应该仔细阅读已知问题或类似文档，并相应地测试特殊问题。

2 测试要求

下文描述了测试的基本硬件和软件要求。随着新设备的上市，硬件要求可能会有所变化。

2.1 对开发人员的基本硬件和软件要求

对所有开发人员的最低要求为：

- 两个用户连接（包括用户身份模块[SIM]卡）
- 两台设备
- PC 套件/Symbian OS 软件以及红外和/或蓝牙连接
- 作为可选项，一台具备红外和/或蓝牙功能的打印机

2.1.1 对Series 60 Platform 3rd Edition的要求

Symbian在v9版本中引入了一个新的安全模型。Series 60 Platform 3rd Edition使用Symbian v9.1，因此也受到该变化的影响。除了以上列出的要求之外，对于Series 60 Platform 3rd Edition的测试，还需要考虑新的内容。新的安全性包括能力模型，它将能力分为多个组。能力模型的目的是确保受信应用软件能够使用某些API和系统资源。有关能力及其划分的更多信息可以从<http://www.forum.nokia.com>上的“Series 60 Platform 3rd Edition: What's New for Developers”和其它相关文档中找到。

使用能力的应用软件要想能够在真正的设备上测试，需要一个开发者证书。开发者证书授予在特定设备上使用所需能力的权限。有关开发者证书的信息以及如何获得和使用它们的说明不久将可以从Symbian Signed网站(<http://www.symbiansigned.com>)上获得。

2.2 区域测试

开发人员测试全球各地的 GSM 频段或其它网络功能（General Packet Radio Service[GPRS]、Enhanced Data Rates for Global Evolution [EDGE]等）时通常会遇到困难。

语言版本的测试分为两类：

- 中文（台湾，香港和简体中文）
- 其它语言

两类测试语言版本是必需的，因为对于特定的语言，如中文、日文、韩文、希伯来文，都有其独特的字符，并且将其输入到应用软件时所用的方法不同，因此，其前端处理器也是不同的。

2.3 商用测试计划

2004年春季，Symbian发布了一个通用Symbian应用软件测试和签名计划，即Symbian Signed (www.symbiansigned.com)。该计划获得了业内的广泛支持，被全球多个销售渠道和运营商所采纳。

强烈建议使用Symbian Signed标准，并且实施该标准对应用软件的要求。要想获得有关Symbian Signed的更多信息，请参访问www.symbiansigned.com。

3 基本要求

3.1 非技术问题

应用软件测试不仅仅包含技术测试。所有非技术问题也需要处理得当。例如，一定不可侵犯知识产权（IPR），应用软件的形式和内容必须能被普遍接受，也就是说，不能冒犯任何团体。同时应确保许可协议有效。

开发人员需检查以下各项：

- 知识产权（IPR）情况令人满意，即开发人员拥有应用软件中所使用的 IPR。
- 应用软件内容可以接受。
- 开发人员有权使用应用软件中出现的所有商标。

3.2 应用软件的性能和可靠性

在开发应用软件时，开发人员一定要牢记，应用软件的运行速度不能牺牲应用软件的使用和用途。在应用软件设计的早期阶段，就必须考虑这个问题。根据应用软件的类型，开发人员也一定要牢记，终端用户可能同时使用其它应用软件。因此，应避免过多消耗设备的运行性能和内存。应用程序不应影响系统功能或其它应用软件的使用。

确保应用软件不会损害用户、系统的应用软件或存储在系统中的数据。有时有害的应用软件是意外产生的。例如，如果应用软件无谓地消耗过多设备的处理能力或占满设备的闪存，则该应用软件运行时是有害的。同时，一定要避免进行无目的的网络通信，因为这会导致用户支付不必要的费用。因此，通过找到并改正代码中的瓶颈、只把必要的数据保存到设备闪存中、并进行合理的设备通信，均可以优化应用软件。

应用软件必须能够处理异常、非法或错误的行为。应用软件一定不能导致移动设备崩溃或死机，必须能够从任何应用软件异常中合理地退出。例如，可以同时打开或使用多个应用软件。

开发人员需检查以下各项：

- 应用软件的运行速度没有牺牲应用软件的使用和用途。
- 应用软件没有过多地消耗设备的运行性能和内存。
- 应用软件没有影响系统功能或其它应用软件的使用。
- 应用软件没有损害用户、其它应用软件或数据。
- 代码中不存在瓶颈。
- 只把必要的数据保存到闪存中。
- 与其它设备或应用软件的通信保持在合理的限度内。
- 必须考虑偶然任务、异常任务（例如，突发情况）和处理错误的任务（例如，在使用应用软件时，由于使用网络连接中断而引起的任务），并合理地处理这些任务。
- 应用软件必须能够处理异常、非法或错误的行为。

3.3 应用软件的安全性

移动设备可被其所有者带到任何地方。由于这个原因，常常存在丢失设备的危险。因此，不要把未经过加密的任何敏感数据存储到移动设备中。例如，应用软件不可把信用卡号码保存到设备内存

中。但是，如果应用软件的特性要求存储敏感数据，那么数据必须要经过加密，并且要输入口令后才能查看。此外，应该明确地通知用户，设备内存中保存了敏感数据。

应用软件不应把敏感数据的输入显示出来，例如密码和口令。但是，被选择的字符可简要地显示出来，以便确认已经输入了某个字符；但是，之后一定要把字符掩盖起来。类CEikSecretEditor应该用于含字母和数字的输入，类CAknNumericSecretEditor应该用于数字输入。

首先应该在资源文件脚本中定义机密编辑器。

```
RESOURCE SECRETED r_alphanumeric_secret_editor
{
    // maximum number of letters accepted by the editor
    num_letters = 5;
}
```

然后在应用软件代码中创建编辑器：

```
void CSecretContainer::ConstructL(const TRect& aRect) {
    CreateWindowL();
    ...

    TResourceReader reader;
    iCoeEnv->CreateResourceReaderLC(reader, R_ALPHANUMERIC_SECRET_EDITOR);
    iSecretEditor = new (ELeave) CEikSecretEditor;
    iSecretEditor->SetContainerWindowL(*this);
    iSecretEditor->ConstructFromResourceL(reader);
    CleanupStack::PopAndDestroy(); // reader
    iSecretEditor->SetFocus(ETTrue);

    ...
    SetRect(aRect);
    ActivateL();
}
```

所输入的文本可以通过编辑器的 GetText 命令访问。

```
const TDesC& CSecretContainer::GetSecretText( void ) const {
    TBuf<KBufLength> text;
    iSecretEditor->GetText(text);
    return text;
}
```

除了资源结构名称为 NUMSECRETED 之外，CAknNumericSecretEditor 的使用方法完全一样。

开发人员需检查以下各项：

- 敏感数据在存到设备中之前已被加密。
- 访问敏感数据需要口令。
- 敏感数据存入设备内存时通知用户。

3.4 打包与安装

为了让用户有更好的体验，能够顺利容易地安装应用软件非常重要。如果安装期间应用软件要求输入注册码，最好显示如何获取注册码的说明。

应用软件应安装到预先指定的位置，并在合理的时间段（5 秒）内启动或提供相应的进度指示。安装时，除非用户选择其它驱动器，否则应用软件应该只在安装驱动器上创建文件。在安装期间，用户应该可以选择安装到 C:（设备内存）或 D:/E:（内存卡）驱动器。因此，*.pkg 文件中不要定义绝对文件路径。替代驱动器字母，"! "字符用于指示为用户提供驱动器选择，将在所选择的驱动器上安装文件。

开发人员应检查以下各项：

- 如果安装期间需要注册码，则显示如何获取注册码的说明。
- 应用软件创建并安装到预先指定的目的位置。
- 应用软件在合理的时间段（5 秒）内启动或提供相应的进度指示。应用软件可以从 **C:**或 **D:/E:**驱动器启动。
- *.pkg 文件中不定义绝对文件路径。

3.5 卸载与重新安装

卸载应用软件必须简单、直接。必须完全删除所有文件和已安装的数据，包括所有图标文件。

开发人员应检查以下各项：

- 可以卸载应用软件。卸载需要完全删除所有文件和已安装的数据（包括所有图标文件）。从 **C:**或 **D:/E:**驱动器删除应用软件时，卸载可以正常进行。
- 卸载后应用软件可以成功地重新安装（包括重新安装到其它内存设备上）。

4 功能测试

应用软件中的所有功能都应该按照文档（例如使用帮助或用户指南）中所描述的方式工作。即使在异常的事件和使用情况下，应用软件也应该正常工作；开发人员必须考虑到所有可能的用户场景。

4.1 环境变化

移动设备可以用于各种环境和情形下。由于这个原因，环境快速改变时，应用软件必须能正确无误地工作。然而，如果发生了错误，应用软件必须能合理地进行处理。

通常来说，要在代码中谨慎地处理所有异常情况，并且无论是否有异常都要保证应用软件的运行，这点非常重要。确保在异常情况下，用户能够及时得到通知。还要注意，应用软件中的任何错误消息都应该清楚明白。错误消息必须能清楚地解释问题本质，并指示需要采取的行动。

开发人员应检查以下各项：

- 即使环境变化，应用软件也能正常工作。
- 如果发生错误，能够合理地对其进行处理。
- 即使正在处理异常，应用软件也能同时运行。
- 在异常情况下，用户能够得到通知。
- 终端用户应该能够容易地理解错误消息。

4.2 网络连接

如果应用软件使用网络连接或需要涉及付费事件—例如，发送一条短信(SMS)—它应该向用户询问许可。在创建、编辑和删除因特网连接点(IAP)前应获得用户的许可。

另外，必须考虑到应用软件运行期间出现网络连接中断的情形，并对其进行合理的处理。应用软件应能处理可能的网络错误，并向用户通知错误。

可以通过与网络操作有关的活动对象检测网络错误，如连接建立，数据读取和写入。显式的网络操作失败通常意味着网络连接存在问题，应用软件必须合理处理这些问题。下面的代码范例说明了连接建立期间如何使用活动对象检测错误。

类 `CMyConnection` 是一个用于管理 `TCP socket` 连接的活动对象类。

```

void CMyConnection::OpenL( const TInetAddr ip_address ) {
    // Connect to system socket server.
    User::LeaveIfError( iSocketServer.Connect() );
    // Open TCP socket.
    User::LeaveIfError( iSocket.Open( iSocketServer, KAfInet,
                                    KSockStream, KProtocolInetTcp ) );
    // Connect to given remote host asynchronously.
    iSocket.Connect( ip_address, iStatus );
    SetActive();
}

void CMyConnection::RunL() {
    // Asynchronous operation has completed.
    if( iStatus != KErrNone ) {
        // Connection establishment failed. Forward error message.
        iMyEngine->HandleConnectionError( iStatus );
    } else {
        // Connection established successfully. Start using the socket.
        ...
    }
}

CMyConnection::~CMyConnection() {
    // Close the socket synchronously.
    iSocket.CancelAll();
    iSocket.Close();
    iSocketServer.Close();
}

```

类 `CMyEngine` 是一个使用 `CMyConnection` 对象的引擎类。它捕获 `OpenL` 方法调用返回的异常，为异步操作提供错误处理功能。

```

void CMyEngine::ConnectToRemoteHost() {
    // Construct remote host address.
    TInetAddr addr( INET_ADDR( 10, 10, 10 ), 1000 );
    // Try to open a connection.
    TRAPD( err, iMyConnection->OpenL( addr ) );
    if( err != KErrNone ) {
        HandleConnectionError( err );
    }
}

void CMyEngine::HandleConnectionError( const TInt aErrorCode ) {
    // Show informative message to user with, e.g., CAknNoteDialog.
    ...
    // Close the connection.
    delete iMyConnection;
    iMyConnection = NULL;
}

```

应用软件还必须处理不允许连接的情形，并在会话结束后关闭连接。

系统常常在应用软件将要结束时关闭网络连接。但是，建议在应用软件结束之前，由用户或通过编程关闭网络连接。

开发人员应检查以下各项：

- 向用户询问使用网络连接或其它收费事件的许可。
- 向用户询问创建、编辑或删除 **IAP** 的许可。
- 处理网络异常并向用户通知异常。
- 在不再需要网络连接或应用软件即将结束时，关闭网络连接。

4.3 暂停应用软件

在许多情形下，应用软件的自动暂停功能非常实用且必要的。例如，所有可能的系统通告能够在应用软件之上显示，从而把应用软件隐藏在后台。

在应用软件交互期间，如果发生下列事件之一，开发人员必须创建一种方法来暂停一个激活会话。

- 屏幕上显示一个系统通告（例如，电话呼入、接收到 vCard、时钟提示、定时器提示、日历提示、消息到达、红外或蓝牙传输，等等）。
- 用户按下红色拨号键、电源开关键、设备的应用键，或选择菜单中的暂停(Pause)。
- 应用软件的主屏幕被系统菜单隐藏，或者其它应用软件设置为前台运行。

实际上，当应用软件隐藏时，它总是处于暂停状态。这在游戏应用软件中尤其重要，因为，如果在游戏被隐藏时没有立刻暂停，游戏者可能会输掉游戏。应用软件在后台运行时，关闭声音和震动非常重要。

应用软件被送到后台或前台时，它会通过 `CCoeAppUi::HandleForegroundEventL` 接收到变化通知。由于应用软件的主要用户界面类以 `CCoeAppUi` 作为基类，因此开发人员只需要覆盖此成员函数：

```
void CMyAppUi::HandleForegroundEventL( TBool aForeground ) {
    CAknAppUi::HandleForegroundEventL( aForeground );
    if( aForeground ) {
        // Application is on the foreground
        ...
    } else {
        // Application is on the background
        // and it should be paused
        ...
    }
}
```

覆盖函数实现应调用相应的基类副本。直接基类的类型取决于平台和用户界面的体系结构（例如，如果未使用视图体系结构，Series 60 中的则调用 `CAknAppUi`）。

暂停之后，会话能够继续进行，故需要把 **Continue** 选项显示给用户。例如，**Continue** 选项可以是应用软件主菜单的一个条目。

开发人员应检查以下各项：

- 发生任何中断时，能够暂停应用软件。
- 如果应用软件被隐藏，它能自动暂停。
- 在应用软件被隐藏时，声音和震动会关闭。
- 在中断后，可以选择继续运行应用软件。

4.4 情景模式

应用软件应该能够读取并依从设备中已激活的情景模式。从这个角度来看，最重要的情景模式有 **Silent**、**Meeting** 和 **GSM Offline**。用户将情景模式设置为 **Silent** 或 **Meeting (beeping)** 时，应该缺省设置为关闭声音。然而，该实现取决于平台及其版本（一些版本组合根本不支持访问情景模式的公共接口）。

Series 60 Platform 2nd Edition及其后面的版本包含 `CSettingInfo` 机制，它支持两种操作模式：当前情景模式设置能够（同步）显式查询，或为所需设置注册观察器（`MSettingInfoObserver` 实例）。可访问设置的集合由枚举 `SettingInfo::TSettingID` 标识。代码范例说明了两种方法：当设置信息初始化时，以及当其后续活动情景模式或振铃类型每次发生变化时，

CProfileSample::AdaptToSettings将获得调用。其类声明如列表 1 所示。相关的成员函数实现如列表 2 所示。

```
class CProfileSample : public CBase, public MSettingInfoObserver {
public:
    static CProfileSample* NewL();
    ~CProfileSample();
    void HandleNotificationL( SettingInfo::TSettingID aId,
                             const TDesC& aNewValue );
private:
    CProfileSample();
    void ConstructL();
    void AdaptToSettings();

    CSettingInfo* iSettings;
    TInt iProfile; // Index of active profile (0=General, 1=Silent, ...)
    TInt iRingType; // Ringing type (0=normal,1=ascending, ...)
};
```

列表 1: 情景模式代码范例的类定义

实际的适配功能是与应用相关的。例如，在播放流行音乐前，应用软件应该检查当前情景模式。如果情景模式为 **Silent** 或 **beeping**，则不播放流行音乐。但是，如果用户打开声音，则应该播放声音，即使情景模式为 **Silent**。在任何情况下，一种好的办法是让用户直接控制应用软件的声音。仅依赖于情景模式设置可能导致不明确的情况（例如，用户可能已经将 **Silent** 情景模式设置成完全相反的内容）。

当用户将情景模式设置为 **GSM Offline** 时，应用软件应无法建立连接；该设置同样适用于蓝牙连接。应用软件应提示用户无法建立连接。

开发人员应检查以下各项：

- 应用软件读取并遵循设备中已激活的情景模式。
- 当情景模式为 **Silent** 或 **Meeting (beeping)** 时，缺省情况下声音被关闭。
- 当情景模式为 **GSM Offline** 时，应用软件无法建立连接。

```

void CProfileSample::ConstructL() {
    // this is observer. Argument 0 would mean no observing
    iSettings = CSettingInfo::NewL( this );
    // Listening to the desired settings
    User::LeaveIfError(iSettings->NotifyChanges( SettingInfo::EActiveProfile ));
    User::LeaveIfError(iSettings->NotifyChanges( SettingInfo::ERingingType ));
    // Initial values
    User::LeaveIfError(iSettings->Get( SettingInfo::EActiveProfile, iProfile ));
    User::LeaveIfError(iSettings->Get( SettingInfo::ERingingType, iRingType ));
    // First adaptation
    AdaptToSettings();
}
CProfileSample::~CProfileSample() {
    // Cancel observation
    iSettings->CancelNotifications( SettingInfo::EActiveProfile );
    iSettings->CancelNotifications( SettingInfo::ERingingType );
    delete iSettings;
}
void CProfileSample::AdaptToSettings() {
    // The settings (iProfile and iRingtype)
    // are either fetched for the first time
    // or they have changed.
    ...
}
void CProfileSample::HandleNotificationL( SettingInfo::TSettingID aId,
                                         const TDesC& aNewValue ) {
    TLex lex( aNewValue );
    switch( aId ) {
    case SettingInfo::EActiveProfile:
        User::LeaveIfError( lex.Val( iProfile ) );
        break;
    case SettingInfo::ERingingType:
        User::LeaveIfError( lex.Val( iRingType ) );
        break;
    }
    AdaptToSettings();
}

```

列表 2: 类实现的情景模式处理部分

5 用户界面测试

应用软件应使用与系统软件相同的 UI 风格和习惯。由于本文中无法列出所有 UI 指南和测试案例，因此建议参考诺基亚为 Series 60、Series 80 和诺基亚 7710 设备创建的 UI 风格指南。

- **Series 60 UI Style Guide**。此文概要描述了 Series 60 用户界面，描述了它的基本组成，给出了如何使用界面组件的范例。最新的 Series 60 设备与 Series 60 Developer Platform 2nd Edition 兼容。
- **Series 80 UI Style Guide**。此文描述了诺基亚 9300 和诺基亚 9500 通讯器中所有应用软件的 UI 规则和一致性，它们与 Series 80 Developer Platform 2.0 兼容。
- **Nokia 7710 UI Style Guide**。此文介绍了诺基亚 7710 用户界面。文中描述了应用软件界面的主要组件，以及诺基亚 7710 设备所提供的常用组件。本文还讨论了系统范围的图形和文本设计，以及对系统功能的支持。

5.1 可用性基础

本章节仅列出 UI 和可用性测试的基本要求。关于从用户体验的角度考虑的大部分基本测试案例列于 *User Experience Checklist For Symbian C++ Applications* 中。关于设计良好的可用性的更详细的说明，参见 *Series 60 Developer Platform 2.0: Usability Guidelines For Symbian C++ Games*，*Series 60 Developer Platform 2.0: Usability Guidelines For Enterprise Applications*，以及 *Series 80 Developer Platform 2.0: Usability Guidelines For Enterprise Applications*。这些文档以及其它与可用性有关的文档可以从 <http://www.forum.nokia.com/usability> 上获得。

良好的可用性是优秀的应用软件和满意的用户体验的基础。如果应用软件的可用性很差，一些用户可能会停止使用该应用软件。为了确保用户满意，应该安排单独的可用性测试，并在应用软件开发的早期阶段就开始测试应用软件。如果在早期阶段发现可用性问题，处理起来会更加简单。

为了让用户有好的体验，能通过主菜单轻松地访问应用软件的所有主要功能是非常重要的。确保每个功能都应该像在应用软件文档和说明中描述的那样正常工作，并且不存在隐藏功能的情形。

还要检查术语的正确性，并且保证没有语法错误。不要使用难以理解的术语或缩略语。

文本不应被截短或全部大写。不使用图像取代重要文本，这一点很重要。图像和图标通常比文字更难理解。但是，简单、一般化的图标，如箭头比文本更好。

确保应用软件向用户提供足够的反馈。通常情况下，如果用户选择一个功能后，显示屏没有改变，并且看起来受到阻塞，那么用户不会等待超过几秒钟的时间。此外，如果应用软件反复地向用户显示某些通告或问题，应该将这些通告或问题关闭。

因此，在选中应用软件的任何功能之后，它们都应该在五秒钟内开始执行。在一秒钟内，必须有视觉上的提示，表明该功能即将执行。举例来说，视觉提示可以是提示用户输入、使用 splash 屏幕或进度条、显示诸如 *Please wait...* 或 *Sending message....* 的文本。可以用声音增强提示效果。对于受到数字版权管理保护的应用软件，或者在启动时需要其它类型验证的应用软件，可能需要考虑较多事项。在这些情况下，等待时间会更长。

开发人员应检查以下各项：

- 可以通过主菜单轻松地访问应用软件的所有主要功能。
- 导航具有逻辑性且清晰明了。
- 每个功能均能够按照文档和应用软件的说明中描述的那样工作。
- 术语和语法正确，并且不使用缩略语。
- 文本没有被截短或全部大写。图形没有代替基本文本。

- 每个屏幕出现的时间应该足够长以使用户读取其信息。
- 在整个应用软件中，保持下列功能的一致性：术语、布局、颜色（或颜色反转）、功能键标签、振动和声音。
- 应用软件向用户提供足够的反馈。

5.2 用户界面

5.2.1 菜单

菜单的结构划分应该具有逻辑性，并且菜单结构不能太深。一个分层次的宽菜单要优于结构深的菜单。一定不要让用户在使用菜单时产生迷惑。在主菜单中必须具有 **Exit**。

一定要适当地定义菜单中的条目顺序。根据使用频率安排选项是明智的，也就是说，最常使用的条目应该位于菜单的上方，**Exit** 条目应该位于底部的最后一个。

菜单中的所有条目均显示为可选择条目，并且条目标签必须具有描述性。用户必须能容易地分辨出哪个屏幕条目可以被选中或被改变，尤其是低级 **UI** 组件用作菜单的情况。如果使用高级 **UI** 菜单，可选择性不是问题，因为高级 **UI** 菜单与系统菜单类似。

如果应用软件要求填写分成多页的表格，必须把当前屏幕的页码以及总页数显示给用户，例如，用以下格式：**Page 3/5**。表格的最后一页必须向用户提供某些信息反馈，例如，在按下提交按钮后出现 **Thank you** 屏幕。如需获得有关为 **Series 80** 应用软件设备设计表格的更多信息，开发人员可以查询 www.forum.nokia.com/usability 上的文档 *User-Friendly Forms In Series 80 Applications*。

在应用软件的主菜单中应该具有 **Help** 选项。**Help** 至少要包括按键的使用方法、应用软件的用途和规则。

应用软件还需要包含一个屏幕，用以给出关于开发人员的信息和关于应用软件的其它一般信息，其中至少要包括名称、版本号、厂商名和产品支持的联系方式。这个屏幕可以是主菜单中的 **About** 屏幕。

开发人员应检查以下各项：

- 动作顺序的划分应该有逻辑性。
- 菜单的结构划分应该具有逻辑性，并且菜单结构不能太深。
- 在主菜单中必须具有 **Exit**。
- 菜单顺序要合乎逻辑。
- 菜单条目的标记清晰。
- 应用软件具有 **Help/ Instructions** 部分，并且可以从主菜单中访问。
- 在 **About** 部分中给出应用软件名称、版本号、厂商名和产品支持的联系方式。

5.2.2 按键

功能键/命令按钮标签应反映设备的用户界面风格。例如，**Series 60** 设备中的功能键应按如下方式标示：

- 左功能键：正向效果（**Options**、**Select**、**Yes**、**OK**）
- 右功能键：负向效果（**Back**、**Cancel**、**No**、**Exit**）

左功能键应该用于打开主菜单。在游戏中，按下左功能键暂停游戏并打开主菜单或一个单独的 **Pause** 菜单。如果全屏模式中未使用标签，左功能键（或两个功能键）应该使用户进入游戏主菜单或 **Pause** 菜单。

此外，选择键应该用作缺省的动作按钮。

开发人员应检查以下各项：

- 功能键标签反映特定设备的用户界面风格。
- 与左功能键有关的标签标示 **Options**、**Select**、**Yes**、**OK** 或其它“正向功能。”
- 与右功能键有关的标签标示 **Back**、**Quit**、**Exit**、**Cancel**、**Clear** 或其它“负向/倒退功能。”
- 在游戏中，按下左功能键暂停游戏并打开主菜单或 **Pause** 菜单。
- 选择键用作缺省的动作按钮。

5.2.3 声音

确保应用软件中的每种声音（如果使用了声音）都是容易区别的，并且有特定的含义。例如：欢快的声音用于正向动作，而悲哀的声音用于负向动作。

保证声音设置的当前状态不会影响应用软件的可用性和可理解性（例如，声音是否激活不会有影响）。

总而言之，在加入声音时应该谨慎，因为过多的声音会使应用软件的使用者变得厌烦。因此，应该仅在声音能够向用户提供额外信息或者让用户有额外体验的地方和情况下加入声音。声音应该容易关闭。

开发人员应检查以下各项：

- 每种声音都有独特的含义。
- 没有声音，应用软件业也可以使用和理解。
- 不要在应用软件中过多地使用声音。

5.2.4 主题

Series 60 设备和诺基亚 **7710** 设备支持主题，主题定义了 **UI** 如何被表现。不同主题情况下，测试应用软件比较重要，因为即使用户改变主题，应用软件也应该可以使用。

开发人员应检查以下各项：

- 应用软件正确工作，且对于不同的主题应用软件仍然可用。

6 内存

设备中内存不足可能会在两次不同的时段影响应用软件：即启动应用软件和执行应用软件时。应用软件应该能够正确处理内存不足，这一点很重要。在内存不足的情况下，当应用软件启动时，应该通知用户内存不足。当 **RAM** 有限时，应用软件不应阻塞或崩溃；而是必须提示用户没有足够的内存执行该应用软件。此外，应用软件不应该占满整个存储空间。

开发人员应检查以下各项：

- 在内存不足的情况下，当应用软件启动时，应用软件通知用户内存不足。
- 当内存不足时，应用软件不会阻塞或崩溃，而是提示用户没有足够的内存。
- 应用软件不能占满驱动器的存储空间 (**C: / D:/E: 驱动器**)，而且要提示没有足够的内存。

开发人员应使用 `FFSSpaceBelowCriticalLevelL` 方法检查空闲闪存是否到达或低于临界水平。

```
// header file
void CheckMemoryL(TUint aBytes=0) const;

// cpp file
void CMyUtils::CheckMemoryL(TUint aBytes) const
{
    if( SysUtil::FFSSpaceBelowCriticalLevelL(NULLL,aBytes)) {
        User::Leave(KErrNoMemory);
    }
}
```

在应用软件启动时，`CheckMemoryL` 不带参数的调用即可用于检查当前空间是否已经低于临界水平。在分配大量内存前，可以使用 `TRAPD(err, iMyUtils->CheckMemoryL(sizeof(CMyBigClass)))` 查看分配后的当前空间是否将低于临界水平。

类似地，`DiskSpaceBelowCriticalLevelL` 方法可用于检查空闲驱动器存储空间是否已经到达或低于临界水平。

7 本地化

Series 60、Series 80 和诺基亚 7710 设备支持多种语言版本，因此测试需要考虑这些内容，以确保无论在哪里使用该功能时，应用软件操作均非常可靠。在启动期间，应用软件需要选择手机所使用的语言；如果应用软件没有该语言，它将使用缺省语言，缺省语言通常是英语。

不应该对 UI 组件采用硬编码。例如，下列各项应该位于代码的一个单独（资源）文件中。

- 用户界面中的所有文本（包括应用软件名）
- 字体名
- 文件名
- 控件和屏幕组件的大小和位置
- 声音
- 快捷键
- 图像

一定要根据目标国家来处理数据、事件和地址格式。验证日期、时间、时区、一周的开始日、数字分割符和货币的格式是否符合目标国家使用语言的习惯，以及是否在整个应用软件中支持这些格式。

数据录入字段必须能接受并正确地显示国家专有的字母，例如，ä、ü 和 ß 等。如果可能，测试所有数据录入字段能否接受并正确地显示所有特殊字符输入。此外，当英文单词被翻译成其它语言时，单词长度可能会发生变化，因此必须为不同语言的文本预留足够的空间，这一点非常重要。

开发人员应检查以下各项：

- 应用软件在启动时选择电话所使用的语言。
- UI 组件没有采用硬编码。
- 遵从目标国家的语言习惯。
- 能够接受并显示国家专有的字母。
- UI 为翻译预留了足够的空间。

要想进一步获得本地化指南，请参考www.forum.nokia.com/usability上的文档 *Terminology And Localization Checklist*。此外，SDK Help 也包括有关应用软件本地化的大量文档。

8 Symbian C++应用软件中的典型错误

本章列举了应用软件测试失败的大量常见原因。

- 系统无法终止应用软件。需要实现一个方法来处理 `EeikCmdExit` 命令。
- 在中断（如电话呼入）发生时，应用软件不能暂停。注：对所有类型的应用软件，这不一定是必要的。
- 当内存不足时出现的各种问题；通常在多个应用软件同时运行时发生。此情况通常导致崩溃、阻塞等。
- 内存不应该全部被耗完。`FFSSpaceBelowCriticalLevel()` 函数可以测试剩余内存。
- 应用软件无法选择安装的目的位置(驱动器 C:或 D:/E:)
- 应用软件的对话框、交互、控件、UI 布局等不一致或与有关 UI 风格指南不符。
- 两个应用软件的混在一起显示，或显示没有完全更新，使得一个应用软件在另一个应用软件已经启动后仍然部分可见。
- 应用软件没有 **Help**。
- 忽略用户指南，用户指南有缺陷或容易混淆。
- 应用软件仅仅进行了模拟器上的调试，没有在真正的诺基亚设备上测试。
- 启动时间太长。
- 快速的按键动作使应用软件出现问题并且导致其崩溃或阻塞。
- 中断后应用软件无法使用。
- 应用软件的名字和版本号不一致或有错误。
- 应用软件太大，导致安装极其耗费时间（许多分钟）或占用大部分内存。
- 应用软件的字符串比所分配的空间长，导致一部分对用户不可见。这在语言测试中相当重要。
- 错误提示不完整。

9 术语和缩略语

术语和缩略语	含义
DRM	数字版权管理
EDGE	Enhanced Data Rates for Global Evolution
GPRS	通用分组无线业务
IAP	因特网接入点
IPR	知识产权
OS	操作系统
SIM	用户识别模块
SMS	短消息服务
TCP	传输控制协议
UI	用户界面

10 参考资料

Designing C++ Applications For the Nokia 7710 Device, <http://www.forum.nokia.com/documents>

Nokia 7710 UI Style Guide, <http://www.forum.nokia.com/documents>

Series 60 Developer Platform: Avkon UI Resources - Editors v1.0,
<http://www.forum.nokia.com/documents>

Series 60 Developer Platform 2.0: Usability Guidelines For Enterprise Applications,
<http://www.forum.nokia.com/usability>

Series 60 Developer Platform 2.0: Usability Guidelines For Symbian C++ Games,
<http://www.forum.nokia.com/usability>

Series 80 Developer Platform 2.0: Usability Guidelines For Enterprise Applications,
<http://www.forum.nokia.com/usability>

Series 60 Platform 3rd Edition: What's New for Developers, <http://www.forum.nokia.com/documents>

Series 60 UI Style Guide, <http://www.forum.nokia.com/documents>

Series 80 UI Style Guide, <http://www.forum.nokia.com/documents>

Symbian Signed Test Criteria, <http://www.symbiansigned.com>

Terminology And Localization Checklist, <http://www.forum.nokia.com/usability>

Tip Of The Month: Use CSettingsInfo To Retrieve User- Settings, <http://www.forum.nokia.com/documents>

User Experience Checklist For Symbian C++ Applications, <http://www.forum.nokia.com/usability>

User-Friendly Form Design In Series 80 Applications, <http://www.forum.nokia.com/usability>

11 文档评估

请花费几分钟评估本文档，以便帮助我们提高文档质量以及选出您觉得最具价值的文档。