# Nokia Mobile Internet Toolkit

**Version 4.1**

# User's Guide

**May 2004**

# Contents

# Introduction

This guide describes how to use the Nokia Mobile Internet Toolkit 4.1.

## Audience

Read this guide if you are a developer of Push, MMS, and DRM messages or of web content for cell phones.

## Standards Support

Nokia Mobile Internet Toolkit 4.1, supports WAP and other standards maintained by the Open Mobile Alliance (OMA), as well as other standards maintained by other organizations.

## Typographical Conventions

The following typographical conventions are used in this guide:

| Notation | Explanation |
|---|---|
| `Courier` | Text that you enter (as opposed to system prompts and responses)<br>• File paths<br>• Commands<br>• Program code |
| *Italic* | • Names of books, documents, and new terminology |
| **Bold** | Names of Windows menus, commands, buttons, and icons |
| URL link | Active link |

# Related Documents

Specifications, that are created and maintained by the Open Mobile Alliance and are freely available at http://openmobilealliance.org.

Assorted documents aboutmessaging technologies are freely available from http://www.forum.nokia.com/.

Numerous specifications of interest to mobile content developers, notably those relating to XHTML Mobile Profile and CSS, are published by the World Wide Web Consortium (W3C). You may access these at http://www.w3.org/

# Getting Started

Nokia Mobile Internet Toolkit (NMIT) consists of a set of editors that you can use to learn how tto create various types of mobile Internet content. NMIT lets you display this content on multiple phone SDKs.

Phone SDKs are installed separately. NMIT detects installed, supported phone SDKs at startup and lists these in its **SDK Control Panel**. You can display content you author on any supported phone SDK by simply clicking a **Show** button within an editor.

Many NMIT editors are used for creating XML-based content types defined by Document Type Defintions (DTDs). These editors employ content validation to check content against a DTD, and they provide features for easily selecting elements and attributions for insertion based on current cursor position. In addition, NMIT provides a DTD Manager through which you can import new DTDs for use by NMIT editors.

This chapter discusses the following topics:

# Features

Major features of NMIT include:

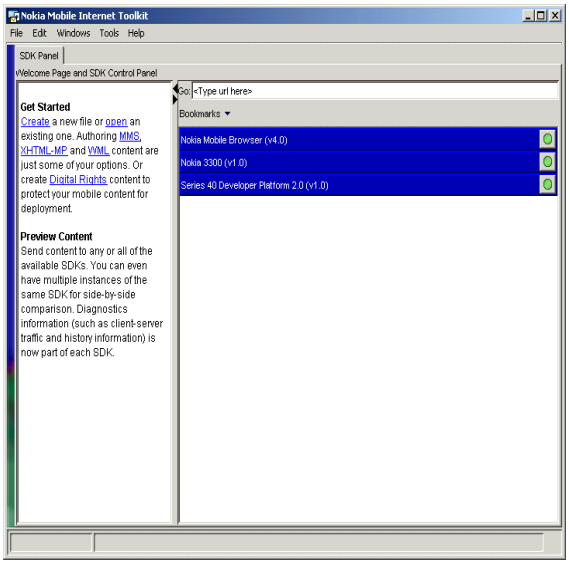
- A set of editors for creating and editing mobile Internet content. These are all accessible using the **File>New** or **File>Open** command. Most editors enable the immediate processing and display of content on a phone SDK by pressing a button (**Show** or **Push**) within the editor. The following table briefly describes these editors.

**NMIT Editors**

| Browsing Editors | Description |
|---|---|
| WML 1.3 Deck | Create a Wireless Markup Language (WML) document. Supports the WML 1.3 DTD, which conforms to the June 2000 WAP specification. Creation of WML 1.2 and 1.1 documents is also supported. |
| WML Script | Create WMLScript content. WMLScript is a derivative of ECMA Script and is used to add programmatic logic to a WML deck, for example, calculations. |
| WBMP Image | Create a Wireless Bitmap (WBMP) image. Similar to most image editors, the WBMP editor enables the creation and editing of WBMP images, as well as conversion of existing images from GIF and JPEG formats to WBMP format. |
| XHTML-MP | Create an XHTML document based on the XHTML Mobile Profile DTD. |
| XHTML-MP+CHTML | Create an XHTML document based on the XHTML Mobile Profile DTD with additional element and attributes taken from Compact HTML. |
| CSS | Create a Cascading Style Sheet (CSS). The CSS contains formatting style definitions that will be applied to the elements specified in an XHTML document. |
| Select DTD From List | Create a document based on a selected system DTD, that is, a DTD from a built-in list or one you created yourself. |
| **Push Content Editors** | **Description** |
| Service Indication (SI) Editor | Create a Service Indication Push message, which is sent to notify a user of the availability of new content. |
| Service Loading (SL) Editor | Create a Service Loading Push message, which is sent to force a user agent running on the client device to download new content (without notifying the user). |
| Cache Operation (CO) Editor | Create a Cache Operation Push message, which is sent to invalidate specific content in the user agent cache (thus forcing a reload of content the next time the user requests that content). |

| NMIT Editors | |
|---|---|
| Multipart Message Editor | Create a multipart message, which is a kind of Push message consisting of more than one part, each of which is separately processed by the user agent. The editor enables the incorporation and ordering of already existing parts (files) into a single multipart message. |
| **Messaging Editors** | **Description** |
| MMS Wizard | Create a Multimedia Messaging file consisting of one or more parts, each part consisting of text, image, or audio content. |
| MMS Message Editor | Create or edit an MMS Message. Major features enable you to add, delete, or rearrange media parts; edit MMS headers and individual Part headers; and push the message to selected phone SDKs. |
| SMIL Editor | Create a Synchronized Multimedia Integration Language file (SMIL and SMIL 3GPP), for specifying the presentational options of an MMS message. |
| **Deployment Editors** | **Description** |
| DRM Editors | Create a DRM message with content and rights to it. |
| Rights Editor | Edit DRM rights in an XML editor. |
| Download Descriptor Editor | Create a Download Descriptor, which describes the content so a phone user can determine whether the content can be loaded on the phone. |

- An **SDK Control Panel** from which you can select one or more installed phone SDKs upon which to display content from an editor or from the Internet. The panel is a tab in the main window and is shown below. See the chapter titled Working With SDKs for a complete description.



- Several sample applications you can use as aids in developing your own content. These include the content types WML, XHTML, Push, and MMS and are readily available through the **File>Open** menu, through the **Bookmarks** button (see above figure), or in the directory: `<NMIT-installation-directory>\samples`.
- Ease of Use Features, described in the table below:

| Ease of Use Feature | Description |
|---|---|
| Pop-Up Menus | Many NMIT commands and functions are accessible by clicking the right mouse button.<br>Right-clicking within a window area but not on an individual item will generally display a popup menu of window-level commands, such as **Close** or **Tear off**.<br>Right-clicking on an individual item displays commands relevant to that item. For example, right-clicking on an XHTML document displays commands for revalidating the document and for showing the document on phone SDKs. If there are no commands particularly relevant to an individual item, NMIT usually displays window-level commands in the popup menu. |
| Tool Tips | Tool tips are pop-up text describing the function of an active control. Throughout the NMIT interface, resting the cursor on an active control such as a button or drop-down lists displays a tool tip. |
| Documentation | The User's Guide (this manual) is provided in PDF format for viewing using Adobe Acrobat. |
| Keyboard Shortcuts | NMIT makes navigation and selection functions available through keyboard and keypad presses for those who prefer the keyboard to the mouse. These are described in the section titled Navigate Using Keyboard Shortcuts. |
| Independent Windows | You can keep multiple documents open and visible simultaneously by "tearing off" a document, that is, detaching it from the main NMIT window so it is an independent window. As an independent window it can be sized, moved, and minimized independently.<br>To tear off a document, right-click in the document window and choose the **Tear Off** pop-up menu option.<br>To put a torn off document back into the NMIT main window as a tab page, right-click in the document window and choose the **Put Back** pop-up menu option. |
| Editor Preferences | Before using an NMIT editor, you may wish to set the available editor preferences such as line numbering, auto-validation, and show warnings. |

# Launching NMIT

You can launch NMIT from the Windows **Start** menu or from the command line.

## Launching NMIT From the Windows Start Menu

To launch NMIT from the Windows **Start** menu, choose:

**Start>Programs>Nokia Developer Tools>Nokia Mobile Internet Toolkit>NMIT4.1**

## Launching NMIT From a Command Line

You can launch NMIT by directly executing a `.exe` or `.jar` file. Launching from a command line is useful if you wish to launch NMIT programmatically, for example, from within another application.

Launching from the command line is also useful for specifying that one or more files be opened upon launch. These files will be opened in the appropriate NMIT editor, based on file type. Additionally, for all text files listed on the command line, you can specify that they be opened using any of the supported character encodings. In this case, the files will be read using the specified encoding, unless encoding information specified in the content itself (XML-based content only) overrides the encoding you specify.

If you have accepted the NMIT default installation location, the location of the NMIT executable is: `C:\Nokia\Tools\Nokia_Mobile_Internet_Toolkit\bin\Toolkit.exe`. In the following syntax statement, substitute the full path name, such as the above default path, for `Toolkit.exe`.

```
Toolkit.exe [-argname argvalue ...] [file ...]
```

An argument is identified by a required hyphen (-) preceding the argument name. For arguments that accept a value, the value follows the name and is separated by a space. If multiple arguments are specified, a space ( ) is required between each, and between an argument and any specified file parameters. The following are the permitted arguments:

`-enc` This argument specifies that any text files listed on the command line be imported into the appropriate NMIT editor (based on file type) using the specified character encoding.

This argument must be specified on the command line prior to any files. For example, the following command launches NMIT, opens `file1` in the WML editor and `file2` in the XHTML editor, and reads both files using the `utf-16` character encoding:

```
Toolkit.exe -enc utf-16 file1.wml file2.xhtml
```

Character encodings supported by NMIT are listed in the **Default Input Text Encoding** list box within the **Toolkit Preferences** dialog (**Edit>Preferences**). You may enter any of these encodings as a command line argument.

When you use character encoding as an argument value, the following rules apply:

- The encoding in the file takes precedence over the encoding in Preferences or the command line.
- If the file does not specify a character encoding, the encoding in the command line argument takes precedence over the encoding specified in Preferences.
- If no encoding is specified in the file or on the command line, the Default Input encoding value in Preferences is used.

You can also launch NMIT from a command line by executing its `.jar` file. Assuming you accepted the default NMIT installation directory, do the following:

1 Navigate to `C:\Nokia\Nokia_Mobile_Internet_Toolkit`

2 Enter the following MSDOS command:

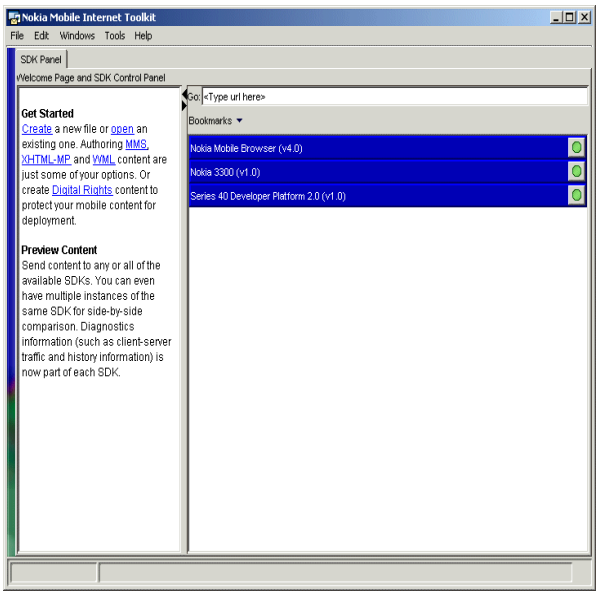
```
PATH=C:\Nokia\Nokia_Mobile_Internet_Toolkit\bin;%PATH%
```

3 Enter the following MSDOS command:

```
java -jar lib\toolkit.jar
```

If NMIT is not currently running, it will be started. If it is already running, the content is displayed in the running NMIT rather than in another instance of NMIT.

Upon launch, the NMIT **Welcome** tab is displayed along with the **SDK Control Panel**:




The **Welcome** tab provides an overview of NMIT functions. You can drag the bar so that the **Welcome** section does not display.

# Menus

This chapter describes the following six NMIT main menu commands and their submenus.

| | | |
|---|---|---|
| File Menu | Edit Menu | Windows Menu |
| Tools Menu | Help Menu | |

# File Menu

The **File** menu provides these options:

**File Menu Options**

| | |
|---|---|
| **New** | Displays the **Available Content Types** dialog from which you can choose a content type for the file you wish to create.<br>After you select the content type and choose **OK**, NMIT opens the appropriate editor, displaying a default template for the content type. |
| **Open** | Displays the **Open file for editing** dialog from which you can navigate to an existing file that you wish to edit.<br>NMIT opens the appropriate editor for the selected file based on the file type. |
| **Close** | Closes the current document and displays the next open document in the NMIT tab order (if there are other open documents). |
| **Save** | In an NMIT Editor window, saves an already named file. If the file is untitled, you are prompted for a file name. |
| **Save As** | In an NMIT Editor window, opens the **Save as** dialog into which you can type a file name for the file you wish to save. By default, NMIT supplies a file extension for the file based on the content type. |
| **Save as Template...** | In an NMIT Editor window, lets you save the current content so that the editor is initialized with that content when you next create a new file of the same Content Type.<br>To restore a default template, quit NMIT and then delete the template you saved from: `C:\Nokia\Tools\Nokia_Mobile_Internet_Toolkit\Templates`. Template names are the icon label prepended by `New` and appended with the Content Type.<br>When you restart NMIT and create new content with the editor, the editor then opens with the default template. |
| **Print** | (Text editors only) Opens the **Print** dialog from which you can print the current document. |
| **Print to HTML...** | Opens the **Save As** dialog which enables you to save the current document in publishable HTML format (text editor only). When such a document is subsequently opened in a browser, it is displayed as it is displayed within an NMIT editor, showing elements and attributes, proper syntax, and color coding.<br>This format is useful for publishing code for review by other content developers or for creating online documentation for programming code. |
| <various file names> | Re-opens any of these most recently opened file names. Up to four files are listed. |
| **Exit** | Closes NMIT and all the SDKs started from NMIT |

## Edit Menu

Edit commands are selectively available or unavailable depending on the context. **Find, Cut, Copy**, and **Paste** are generally available in Editor windows. Others, such as **Undo** and **Redo** are unavailable if you have as yet taken no action. Unavailable commands are grayed out.

**Edit Menu Options**

| | |
|---|---|
| **Undo** | Undo the previous action.<br>Note that NMIT dynamically renames this menu item to reflect what its action will be. For example, if you have performed a paste operation, the menu item is renamed **Undo addition**. This item is grayed out if there is no previous action. |
| **Redo** | Do the action that had previously been undone; that is, undo the last undo.<br>Note that NMIT dynamically renames this menu item to reflect what its action will be. This item is grayed out if there is no previous action. |
| **Cut** | Delete the selected text and copy it to the Clipboard. |
| **Copy** | Copy the selected text to the Clipboard. |
| **Paste** | Paste the contents of the Clipboard at the cursor insertion point. |
| **Find** | Opens the **Find/Replace Text** dialog in which you can specify text to find (and optionally replace) in the current editor window. |

## Windows Menu

The **Windows** menu enables you to view all open documents and provides options for selecting which window is the active window and for tearing off or putting back the active window.

**Windows Menu Options**

| | |
|---|---|
| <open documents> | Displays a list of all open documents. Select an open document to bring that document to the foreground for editing. |
| Tear Off <current> | Tears off the active document from the NMIT main window, creating an independent window (with a complete menu bar) that can be sized, moved, and minimized independently. This function enables you to view multiple documents simultaneously.<br>To put a torn off document back into the main NMIT window, choose the **Windows>Put Back** menu item from the torn off window's menu bar.<br>Both the tearing off and the putting back of document windows can also be accomplished by right-clicking the mouse in a document window and then choosing the desired option from the popup menu. |

## Tools Menu

The **Tools** menu provides these options:

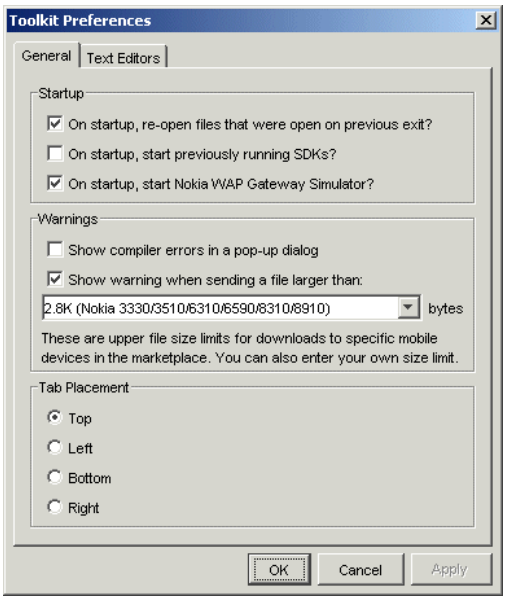
**Tools Menu Options**

| | |
|---|---|
| **SDK Control Panel** | Opens the SDK Control Panel tab from which you can choose among available phone SDKs to which you wish to send a document for display. The use of this control panel is described in the chapter titled Working With SDKs. |
| **DTD Manager** | Opens the DTD Manager, an NMIT component that enables you to register additional DTDs for use with NMIT content editors (see DTD Manager). |
| **WAP Gateway** | Launches Nokia WAP Gateway Simulator (NWGS) if installed on your computer. NWGS is a single-user WAP Gateway based on the multi-user Nokia Activ Server.<br>In order to browse the Internet using a phone SDK, the phone SDK must be configured to use NWGS or an external WAP gateway. If the former, NWGS must be running. This menu option is provided for convenience in launching NWGS. |
| **Preferences** | Opens the Preferences dialog, described below. |

## Preferences

Choose **Tools>Preferences** to open the Toolkit Preferences dialog, which contains the two tab pages **General** and **Text Editors**.

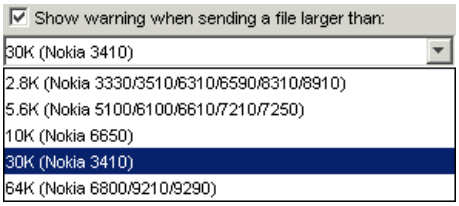
The **General** tab provides settings arranged in the three regions: **Startup**, **Warnings**, and **Tab Placement**, as shown below.




### General Tab Options

| **Startup** | |
|---|---|
| **On Startup, re-open files that were open on previous exit** | When checked, NMIT reopens files that were open when you last exited from NMIT. |
| **On Startup, start previously running SDKs** | When checked, NMIT starts all the SDKs that were running when you last exited from MIT. |
| **On Startup, /Start Nokia WAP Gateway Server** | When checked, NMIT starts the Nokia WAP Gateway Simulator every time you start NMIT. |
| **Warnings** | |
|  | If checked, NMIT displays compilation errors in a pop-up dialog box, as well as in the message area below the editing window.<br>Unchecking this box disables pop-up display of compilation error messages. |

### General Tab Options




If checked, NMIT displays a warning under these conditions:

- The current WML or XHTML document you show on a phone SDK exceeds the size you specified in the bytes text entry field.
- The size of the compiled content of a WML file saved in binary format exceeds the size you specified in the bytes text entry field.

If unchecked, NMIT displays no warnings.

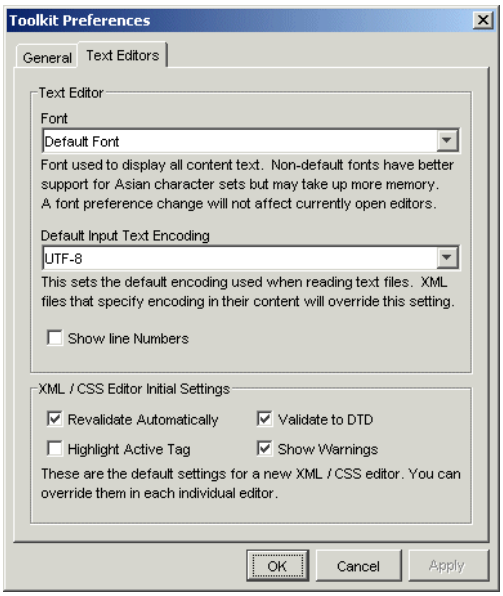
This field uses a drop-down list box displaying several existing mobile devices and the size of largest file that can be downloaded to the device.

You can choose from the list, or you can enter your own value by selecting the existing value and then typing in a new value.

MMS ignores this setting and uses size limits defined for each SDK.

| Tab Placement | |
|---|---|
| **Top** | Positions document tabs along the top of the display window. |
| **Right** | Positions document tabs to the right of the display window. |
| **Bottom** | Positions document tabs along the bottom of the display window. |
| **Left** | Positions document tabs to the left of the display window. |

The **Text Editor** tab provides settings that apply to all text editors in NMIT, including the font and encoding to be used in the display of documents open in a text editor, and initial settings for XML/CSS editors:




## Text Editor

**Font**

Sets the font to be used in the display of text within the content editing region of all text editors. NMIT offers the two font options **Default Font** and **Arial Unicode MS**, though the latter is offered only if this font is already installed on your computer. Some differences between these are given below; however, you should choose whichever you prefer.

The **Default Font** is a monospaced font: each character is of the same fixed width. This is the NMIT default font, a sample of which is shown below:

```
<body>
    <hl> Heading </hl>
```

The **Arial Unicode MS** font is a "richer" font than the NMIT default font and contains support for many languages; however, it is not a monospace font. It is recommended to choose this font if your content contains characters from any Asian language. The following code snippet uses the **Arial Unicode MS** font:

```
<body>
  <h1> Heading </h1>
```

When you change a font, the change is seen in documents you open thereafter; documents already open are unchanged.

**Default Input Text Encoding**

Use this drop-down list to choose a text encoding that an NMIT text editor will use as the default encoding when reading in a document that you have chosen to open. Note the following:

- Whenever an XML-based document contains a statement specifying its own encoding (the `charset` attribute in the XML prologue on the first line), NMIT reads in the document using that encoding rather than the encoding specified in the **Default Input Text Encoding** list.
- Some encodings have an identifying signature in the first few bytes. In this case, NMIT may detect that the document uses a different encoding and will then use the encoding it discovers rather than the one you specified in the list.
- If the document was specified as a command line argument with the `-enc` encoding, NMIT uses that encoding.
- If the Default Input Text Encoding does not match the encoding used at the time the document was created, the document may appear garbled or unreadable. In this case, close the document and re-open it using a different input encoding. (See the section Preferences for details on how to do this.
- NMIT uses the UTF-8 encoding by default.

For additional information, see the section Displaying a Non-UTF-Encoded Document on Phone SDKs.


If checked, line numbers are displayed along the left border in the text editing areas of all the text-based editors.

If unchecked, no line numbers are displayed.

Line numbers in compile error reporting will be displayed even if this box is unchecked.

## XML/CSS Editors


If checked, NMIT's text-based editors revalidate document content automatically at five-second intervals after editing activity stops. This saves you from having to revalidate manually using the **Revalidate** button in the editor window. This is the default setting. Open files are automatically revalidated, even if this box is unchecked.

If unchecked, automatic revalidation is disabled.


If checked, NMIT's text-based editors validate the document content against the DTD specified in the `DOCTYPE` declaration found within the document content. In this case, the parser checks that all elements and attributes are used in accordance with the rules specified in the DTD as well as in accordance with the XML rules for creating "well formed' content. This is the default setting.

If unchecked, NMIT's validating parser checks only for "well-formed" XML-based content.


If checked, the element in which the cursor is currently positioned is highlighted in a light background color.

If unchecked, no highlighting is used to identify current cursor position. This is the default setting.


If checked, non-fatal parser warnings are displayed whenever NMIT's parser revalidates the document.

If unchecked (the default), no warnings are shown.

Errors are always shown no matter what the setting.

## Help Menu

The **Help** menu provides options for accessing NMIT user guide information and release notes:

**Help Menu Options**

| | |
|---|---|
| **User's Guide (PDF)** | Opens the *Nokia Mobile Internet Toolkit 4.1 User's Guide* (this manual) in Adobe Acrobat Reader. |
| **Nokia Update Manager** | Launches Nokia Update Manager, a small application that enables you to easily download and install Forum Nokia software. An Internet connection is required to use Update Manager. |
| **Release Notes** | Displays the release notes accompanying this version of NMIT. These notes include version information, changed features, and unresolved issues. |
| **About Mobile Internet Toolkit** | Opens the **About Mobile Internet Toolkit** dialog, which provides NMIT version information. |
| **Feedback** | Lets you send any comments you have about this product to Nokia Corporation if you have an active Internet connection. |

# Navigate Using Keyboard Shortcuts

NMIT provides keyboard shortcuts you can use to navigate and perform operations. These are key combinations that can be used in place of normal mouse operations and are particularly useful if you prefer using the keyboard to using the mouse.

- **NMIT Windows shortcuts.** Use the Windows shortcut **ALT+TAB** to move input focus from one window to another, for example, to move from the NMIT main window to a "torn off" NMIT editor window. Within the Multipart Editor, use **TAB** to move input focus from one header field to another. Use **CTRL+TAB** moves the input focus between window elements such as controls or buttons that you may wish to select.
- You can tell that buttons have the input focus by noting an indented rectangular border; text-entry fields are highlighted.

- **Menu item shortcuts.** To choose a menu item, press the **ALT** key while simultaneously pressing the letter that is underlined in the menu item name. Once a menu is opened, you can select another item within that menu by pressing the **ALT** key in combination with the **UP** or **DOWN** arrows and then choosing **ENTER**.

A key combination is executed by holding down the key preceding the plus sign (+) and then pressing the key following the plus sign. In effect, pressing two keys simultaneously.

**Keyboard Shortcuts for NMIT Windows**

| Key Combination | Description |
|---|---|
| ALT+TAB | On Windows operating systems, moves focus from one Windows process to another. In NMIT, moves focus from the main NMIT window to a phone SDK or to a torn-off NMIT window. |
| ALT+x | On the menu bar, opens the menu whose name contains an underlined x (x).<br>Within a menu, chooses the menu item whose name contains an underlined x (x). |
| TAB | Within a window, navigates forward from one control or field to the next, eventually looping back to the starting point. The current position (or "focus") is indicated by a dotted rectangle.<br>In an editing area, inserts a tab character. |
| CTRL+TAB | Within a window, moves focus from inside an editor window to the control buttons and fields on that window but outside the editing area. |
| SHIFT+TAB | Within a window, navigates backward from one control or field to the next, eventually looping back to the starting point. The current position (or "focus") is indicated by a dotted rectangle. |
| ENTER | On a button, executes either the default button function or the function having the active focus.<br>In an editing area, enters a carriage return character. |
| ALT+ENTER | Within a window, displays the application menu allowing you to move, size, resize, minimize, maximize, or close the window. |
| <space bar> | On a button, executes the button function.<br>On a check box, pressing <space bar> checks the button if it is currently unchecked or unchecks it if it is currently checked.<br>When focus is in an editing area, enters a space character. |
| <Left arrow> | When a main menu item is open, moves the focus to the main menu item to the left. |
| <Right arrow> | When a main menu item is open, moves the focus to the main menu item to the right. |
| <Up arrow> | On a menu or one of its menu items, moves the focus to the previous item on that menu. |

**Keyboard Shortcuts for NMIT Windows**

| Key Combination | Description |
|---|---|
| <Down arrow> | On a menu or one of its menu items, moves the focus to the next item menu on that menu. |
| PgUp | In an editor window, scrolls one page up. |
| PgDn | In an editor window, scrolls one page down. |

# Sending comments to Nokia Corporation

You can send any comments you have about this product to Nokia Corporation if you have an active Internet connection by selecting **Help>Feedback**. Selecting this option automatically opens http://www.forum.nokia.com/feedback where you can enter your comments.

# Browsing Editors

This manual defines a *browsing* editor as an editor used to create content for display by the browser application of a phone SDK. Note that phone SDKs also have inbox-type applications for handling Push messages and MMS messages. You do not "browse" the Push and MMS content types; these are delivered to you.

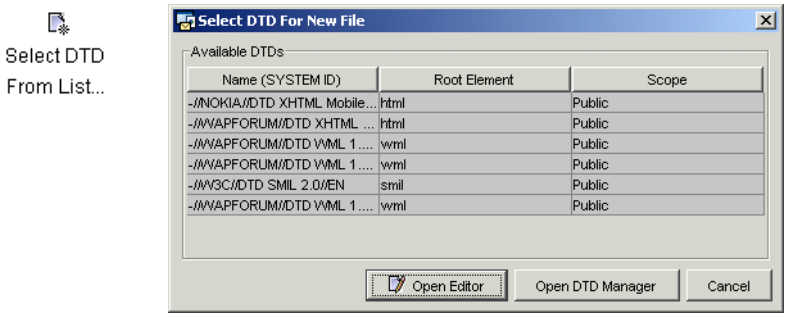
Browsers on many phone SDKs can display both WML and XHTML text documents, as well as other content types that are referenced by or invoked within these documents, such as WBMP or GIF images, or, in the case of WML documents, WML Scripts. All of these content types, then, are destined for display or handling by the browser application of a phone SDK, and this manual groups the editors for these content types together under the category "browsing editors."

Within the category of browsing editors may be found a set of editors that employ a validating parser to check that text content is valid. These *validating editor*s are designed for content that (1) is XML-based and uses a DTD or (2) employs hierarchical text elements that can be validated for syntax (such as CSS). The two NMIT browsing editors that are not validating editors are the WML Script and WBMP Image editors. The following table summarizes these distinctions among NMIT's browsing editors.

| Validating Editor | Validatable Content | Uses DTD |
|---|---|---|
| WML 1.3 Deck | Yes | Yes |
| XHTML-MP | Yes | Yes |
| XHTML-MP+CHTML | Yes | Yes |
| Select DTD From List... | Yes | Yes |
| CSS | Yes | No |
| WML Script | Yes | No |
| WBMP Image | No | No |

You can use any of the following methods to open a validating editor, once you have in mind the content type you wish to create:

- Choose **File>New** and then select one of the following icons in the **Available Content Types** dialog.
- Choose **File>New**, choose the **Select DTD From List** icon (in the left column below), select the desired DTD from the list showing all currently installed DTDs, and then choose the **Open Editor** button:




- Choose **Tools>DTD Manager**, select a DTD from the list (same list as in (2) above) in the **Available DTDs** region, and choose **Open Editor**.
- Choose **File>Open** and then choose an existing file with one of the following types: `wml`, `wmls`, `xhtml`, `html`, `wbmp`, `gif`, or `css`.

Editors for structured content types are functionally similar in the editing facilities they provide. For this reason, they are described together in the section titled Browsing Editors for Validatable Content. The WBMP Image and WML Script Editors are described separately in the section titled Other Browsing Editors.

## Browsing Editors for Validatable Content

Recall that a distinction is made between validatable content that is based on a Document Type Definition (DTD) and validatable content which is not. (CSS is the single content type that does not use a DTD.)

This section discusses first the editing features common to all validating editors; secondly, those features used only when editing DTD-based content types; and then concludes with separate discussions of each validating editor.

Validating editors are editors that operate on any of the following structured content types:

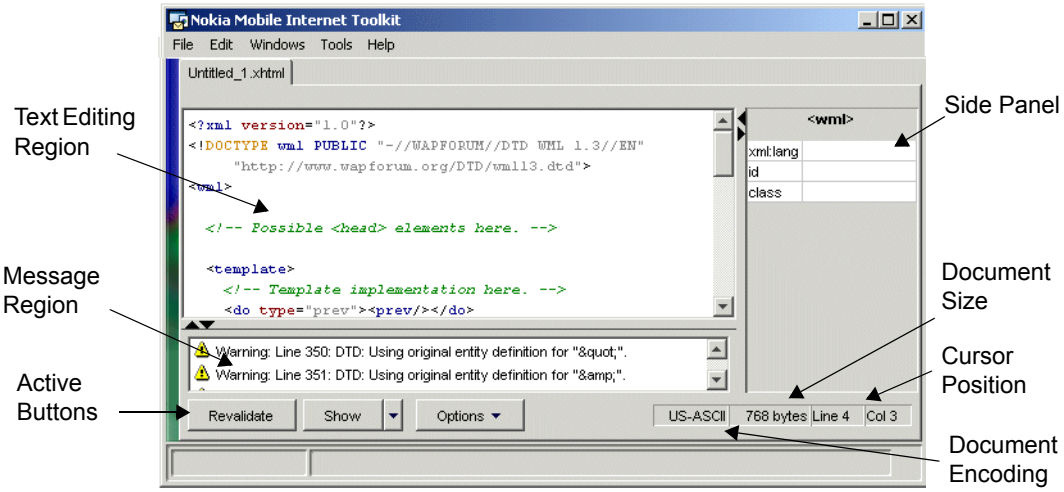
- WML 1.3 Deck
- XHTML-MP
- XHTML-MP+CHTML
- CSS
- SMIL Editor (The SMIL content type is DTD-based, and SMIL editor features are identical to those of the browsing editors for text content except for the lack of a **Show** button to display content on a phone SDK. It is discussed in the Messaging Editors chapter instead of here because it is not used to create browsable content.)
- Any content type opened using the **Select DTD From List** menu option (**File>New** and then the **Browsing** tab).

## Editing Features For Validatable Text Content

This section describes editing features for validating editor.

### Editor Window Layout

The following figure shows the major regions in a validating editor:




Note the following with respect to the above figure:

- Enter text in the text editing region. The NMIT editor displays a template for the content type you have chosen when you create a new file.
- Use the side panel to insert valid attributes for the element highlighted by the cursor in the text editing region and to modify attribute values.
- The message region displays validation messages when you save, **Show**, or **Validate** a document.
- Use the active buttons to validate content, show the document on phone SDKs, or set editing options.

- The document summary displays details about the current document.

## Initial Content of the Document Editing Region

Each of NMIT's supported XML-based content types has an associated template. Therefore, when you open an XML-based content type, the editor is initialized with the associated document template. It is within this template that you begin entering your content. All required elements are already inserted in the editor display, and comments suggest where you might wish to add various new elements.

For an NMIT editor to open an XML-based content type, a unique template for that content type must be installed. NMIT does provide a template file for the pre-installed DTD-based content types. If you wish to install your own DTD, you must provide an associated template file or have NMIT auto-generate a template file. These operations are performed using NMIT's DTD Manager.

Note that when you open the CSS text editor, which is not XML-based, the text editing region of the editor window displays a sample CSS style sheet. This is helpful in that it shows proper syntax and it may already include tags that you had planned to use.

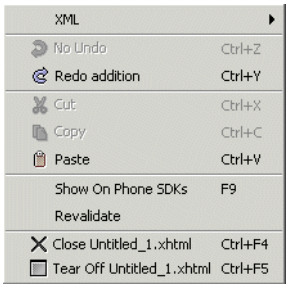
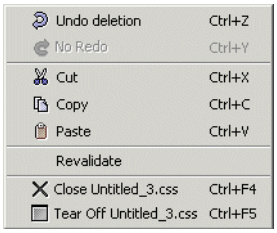
## Color Coding

A validating editor automatically differentiates among recognized data entered. For example, element names are displayed in blue; attribute names, in red; attribute values, in gray; and so on. This makes it easy to visually "parse" the displayed content. Note that colors might vary depending on the configuration of your monitor.

## Mouse Operations

The following table describes the possible mouse operations on text in the text editing region:

| | |
|---|---|
| Single click | • Sets the insertion point.<br>• Upon an element name, displays permissible attributes in the side panel.<br>• Upon an attribute name, highlights that attribute in the side panel. |
| Double click | • Selects a word, where a word consists of the characters bounded by space or special characters such as the "<" and ">".<br>• Upon element or attribute names, also highlights these in the side panel. |
| Triple click | Selects an entire line. |

| | |
|---|---|
| Right click | Displays a pop-up menu, whose options vary depending upon whether the document content type is XML-based or CSS. For example:<br>**RIGHT-CLICK ON DTD-BASED CONTENT**<br>  <br><br>**Right-Click on CSS Content**<br>  <br> |
| **XML** | Displays permissible attributes that can be inserted at the cursor position. See the next section titled XML Element and Attribute Insertion Features. |
| **Undo** | Undoes the previous text editing operation. |
| **Redo** | Performs again the last text editing operation. |
| **Cut** | Cuts the selected text to the Clipboard. |
| **Copy** | Copies the selected text to the Clipboard. |
| **Paste** | Pastes the contents of the Clipboard at the insertion point. |
| **Show on Phone SDKs** | For XML-based documents only, displays the current document on the currently running phone SDKs. Only phone SDKs you have launched using the SDK Control Panel will receive the document content.<br>Because CSS documents cannot be independently displayed, this option is unavailable for CSS documents. |
| **Revalidate** | For an XML-based document, checks that the document is well-formed (correct syntax) and strictly conforming to the DTD.<br>For a CSS document, checks that the document is well-formed (correct syntax). |
| **Close** | Closes the document. You will be prompted to save as yet unsaved documents. |
| **Tear Off** | Removes the document from the NMIT main window and places it in a separate, independent window. |

## XML Element and Attribute Insertion Features

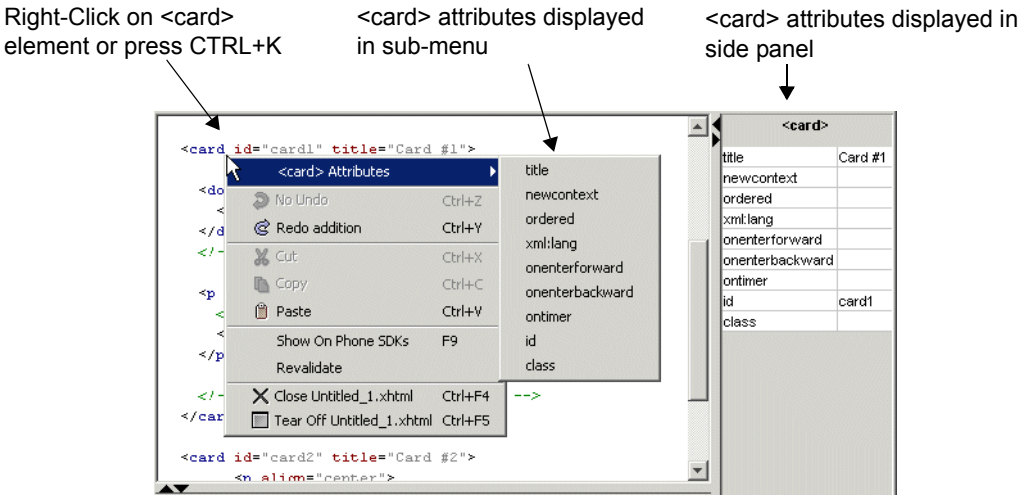
When editing an XML-based document, you need to know which elements, if any, are valid child elements of any given element. As well, you need to know which are the valid attributes that may be inserted within an element. Then, once you have determined which element or attribute you would like to insert, it is helpful to have these inserted for you automatically, using correct syntax and spacing. NMIT provides the following features in support of this:

- XML popup menu option
- CRL+K key combination

- Side panel

All of these features are available for XML-based documents; however, the XML popup menu is not available for CSS documents. XML-based documents must be valid, however, because menu options displayed are derived from the rules inherent in the DTD. You must successfully revalidate your document before using these automatic insertion features because revalidation maps the cursor text position (where the item gets inserted) to the XML logical structure of the document.

To use any of the above features, first position the cursor by clicking within the current document. For example, in the following figure, right-clicking on the `<card>` element in the document displays a popup menu within which the top **XML** menu option changes to **<card> Attributes** when selected. At the same time, legal attributes of the `<card>` element are displayed both in a submenu and in the side panel at the far right:

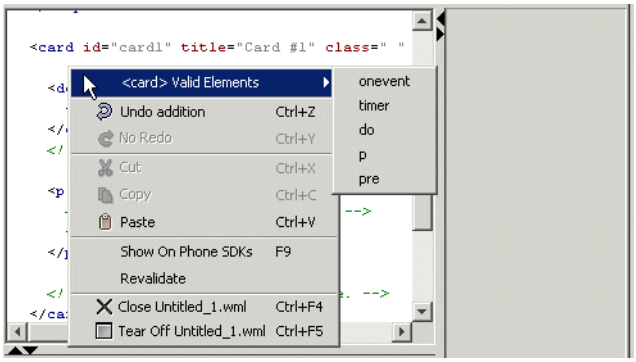



Note that pressing **CTRL+K** within the `<card>` element results in the same popup menu as does right-clicking and choosing the **XML** submenu option.

To insert an attribute: (1) simply choose it from the submenu using the mouse or (2) use the Up and Down arrow navigation keys to highlight it and then choose **Enter**. So, in the example shown in the figure above, choosing the `class` attribute inserts the string `class=" "` and the insertion occurs within the opening and closing brackets of the `<card>` element.

Both the XML menu and CTRL+K functions operate with sensitivity to cursor position. In the following example, the **XML** menu option is selected with the cursor position on the blank line below the line containing the `<card>` element shown in the earlier figure. In this case, the cursor position is outside of the opening and closing brackets of the `<card>` element (where attributes would be appropriate) but is inside of the opening `<card>` and closing `</card>` tags where child

elements would be appropriate. Consequently, valid child elements rather than attributes are displayed in the submenu:



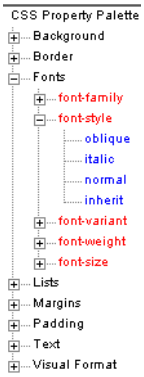

Note that in the above figure, the side panel to the right is blank, as no attributes may be validly inserted at the insertion point.

The following are guidelines for determining whether attributes or elements are displayed in a popup menu, as well as whether the side panel is active:

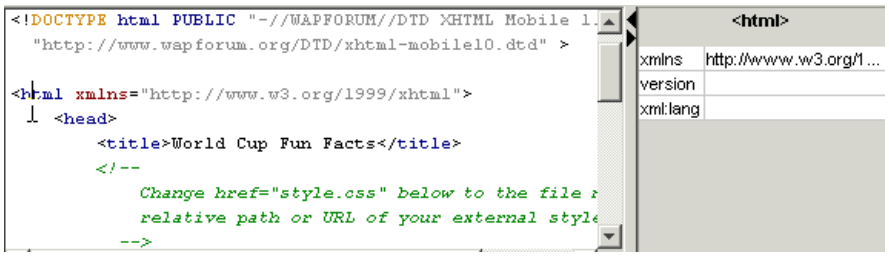
- Elements are displayed if the cursor is positioned anywhere between the opening and closing tags of an element which has valid child elements (between the `<card>` and `</card>` tags, for example).
- Attributes are displayed if the cursor is positioned within the delimiters of an opening element, whether or not that element has valid child elements (for example, within the `<p>` or `<head>` tags. (but not within the brackets of the closing `</p>` tag).

For XML-based documents, the side panel is displayed whenever the cursor is positioned such that attributes, and not elements, can be validly inserted (see the list of rules above). When attributes cannot be legally inserted at the cursor position, the side panel is blank. For CSS documents, the side panel is always available and displays the list of valid CSS properties available for mobile devices.

Attributes and attribute values are displayed in the side panel in hierarchical fashion. Attributes are listed in eight categories, each of which when expanded displays the attribute names within that category. When an attribute name is expanded, its valid attribute values are displayed. The following figures depict this hierarchical structure:

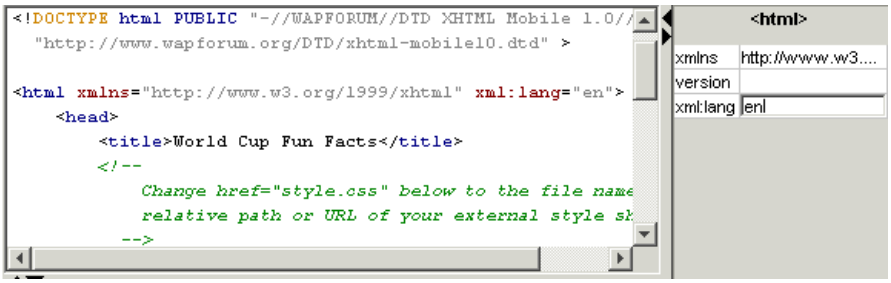
When the side panel is active, the element name of the current element (the element in which the cursor is positioned) is displayed at the top. For example, the `<html>` element is the current element in the figure displayed below:




The side panel is displayed as a two-column table where the first column displays valid attribute names and the second column displays the values currently assigned to the attributes. An empty cell in the right column indicates that no value is currently assigned to that attribute, that is, that the attribute is absent from the document.

If you click in a cell in the second column, it becomes an active text-entry box into which you can enter a value for the attribute in the adjacent column. For example, clicking in the column adjacent to the `xml:lang` allows us to enter the value “en” as displayed in the figure below.

To insert into the document the attribute with the newly entered value, choose **Enter** or click anywhere in the first column. This is shown in the figure below:




You can navigate up or down the list of entries in the side panel by using the Up and Down arrow keys.

If you change your mind and do not wish to insert any attributes, click anywhere in the text editing region to clear the side panel.

## Active Buttons

This section describes the active buttons along the bottom of the text editing window. Which buttons are displayed depends upon the content type of the current document. For example, when a CSS document is the current document, the **Show** button is not displayed because it is not possible to show this document on a phone SDK. Subsequent sections describe each button.

### *Revalidate*

Choose the Revalidate button to validate the current document. Validation is done by NMIT’s validating parser. Using the **Validate to DTD** check box menu option in the **Options** button, you can set the parser to validate against a DTD or not. The following is a brief discussion of validation, what it involves, and how it can be applied to documents of various content types.

A *well-formed* document is a document that conforms to certain syntactical rules and therefore can be processed unambiguously by the software, whether or not the document contains a DTD. Both CSS and some HTML documents can be validated for well-formedness, though neither has a DTD. Likewise, documents that do contain DTDs, such as the WML, XHTML, and SMIL content types, can also be tested only for well-formedness. You could, for example, create a brand new element, insert it into a WML document, and, so long as it was specified with correct syntax, validate it successfully for well-formedness despite the fact that it was not defined within a DTD.

A *strictly conforming* document is one that is (1) well-formed (as described above), (2) contains only elements, attributes, and attribute values as specified in the DTD, and (3) contains these only in the correct locations within the document. The parser may fail a document in a test for validity against the strictly conforming standard if, for example, text was entered at an improper location, an attribute name not specified in the DTD was used, or a child element was specified outside of its parent element. To test that a document is strictly conforming, ensure that the **Validate to DTD** check box is checked, and then choose the **Revalidate** button.

To test a document for well-formedness, ensure that the **Validate to DTD** check box (**Options** button) is unchecked, then choose the **Revalidate** button. Testing for well-formedness is important for CSS documents and for HTML documents that you might wish to convert to XHTML. When converting to XHTML, for example, the first step might be to ensure that the document is well-formed, and after that to begin making it strictly conforming by replacing nonstandard elements and attributes with conforming XHTML elements and attributes.

### *Show*


Causes the current document to be displayed on the phone SDKs currently checked in the SDK Control Panel. Note that you must first launch a phone SDK before sending a document to it.

Also note that, for WML documents, the **Show** button has no drop-down arrow because it cannot be displayed in a browser (as can XHTML documents) but rather only in a phone SDK.


Clicking the narrow rectangle to the right of the Show button displays a drop-down target selector offering two choices:

- **Show on Phone SDKs**. Sends an XHTML document to all selected phone SDKs; this is the same as simply choosing the **Show** button.
- **Show in Browser**. Sends an XHTML document to the default web browser for your computer.

### *Options*


Choosing the **Options** button displays the default settings in effect for all XML-based and CSS editors (WML, XHTML, SMIL, CSS).

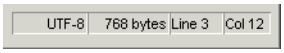
These default options are specified in the **Editor Preferences** tab (**Edit>Preferences**).

The purpose of the **Options** button is to enable you to override these default options for the current document. Individual options are described below.

| | |
|---|---|
| **Output Encoding** | Choose the character encoding to be used the next time that NMIT saves the current document to disk. A document is saved to disk whenever you explicitly save it or when you choose **Show** to display the document on a phone SDK.<br><br>You must be sure that a character encoding that you choose here is identical to the character encoding specified within the document itself, if one is specified in the document. Within an XHTML document, for example, character encoding may be specified as the value of the `charset` attribute within the XML prologue on the first line.<br><br>Typically, the document character set for XML and WML is the Universal Character Set of ISO/IEC-10646. This character set is usually identical to Unicode 2.0. However, any character encoding ("charset") that contains a proper subset of the logical characters in Unicode may be used. The list of NMIT-supported encodings is shown in the radio button list (see left).<br><br>By default, NMIT editors save the document in the same encoding used to read in the document (if you opened the document) or in UTF-8 if you created a new document and you make no selection in this radio button list.<br><br>For additional information, see the section Displaying a Non-UTF-Encoded Document on Phone SDKs. |
| **Highlight Active Tag** | Check this box if you prefer the editor to highlight the element in which the cursor is positioned; the highlight is a light background color. |
| **Revalidate Automatically** | Check this box if you wish the current document to be checked for validity automatically, roughly at intervals of five seconds after text entry has stopped. If checked, you do not need to regularly use the **Revalidate** button to validate the document you are editing.<br><br>If unchecked, no automatic validation occurs. Note that highlighting of document content depends on validation. |
| **Validate to DTD** | If checked, the document is validated against the DTD that it references.<br><br>If unchecked, the document is validated for well-formedness only, and not against the DTD.<br><br>The section describing the Revalidate button provides a full discussion of validation. |
| **Show Warnings** | If checked, all three categories of messages (informational, warning, and error) are displayed in the message region.<br><br>If unchecked, only error messages are displayed.<br><br>See the section Message Region for additional details. |

## Document Summary Bar

The document summary bar to the right of the active buttons along the bottom of the editing window displays summary details about the current document, as shown below:



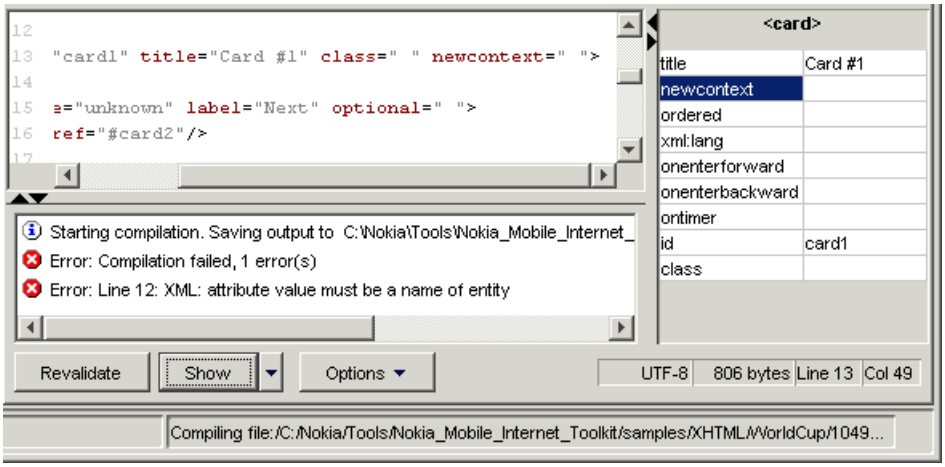

The summary details provided for the current document are as follows:

- The character encoding, UTF-8 in the above figure.
- The size of the document in bytes. For encoded content types, such as WML, the encoded content size is given. For unengaged content types, such as XHTML and CSS, the size is given for the text length.
- The line and column number in which the cursor is presently positioned.

## Message Region

The message region is an adjustable screen region below the text editing region wherein messages relating to the validation and compilation of the document are displayed. Messages are displayed whenever you choose the **Revalidate** or **Show** buttons or whenever automatic validation occurs.

In the following figure, a portion of a WML document is displayed, the **Show** button has been chosen, and three messages relating to the compilation of the document are displayed:




Messages may have any of the following three severity levels:

- Informational. These can be turned off using the **Options** button.
- Warning. These can be turned off using the **Options** button.
- Error. These are always displayed regardless of the options settings.

Double-click an error message to bring the cursor to the line within the document where the error was encountered; the line will also be highlighted, whether or not you have selected the **Highlight Active Tag** option in the **Options** menu.

Note that messages relating to an error within the document always indicate the line number on which the offending item was encountered. (The second error message in the figure above refers to

line 12.) Because of this, you may wish to specify that the document itself display line numbers (**Edit>Preferences**).

## Validating the Same Content Against Different DTDs

You can validate a document against other DTDs that are based on the same root element. For example, a `wml` file may be checked for validity against any of the three installed DTDs that specify the root element to be `<wml>`.

Assume you are working on a WML document that references the WML 1.3 DTD. To check whether the document is also valid against a WML 1.2 DTD, follow this procedure:

1 Place the cursor anywhere within the `DOCTYPE` declaration of the current document. In the side panel, the DTD drop-down list is displayed.

2 Select the desired DTD from the DTD drop-down list. That DTD is substituted for the existing one within the document.

3 Choose the Revalidate button. If there are no errors, the document is valid for the substituted DTD.

The following figure shows a WML document with the DTD drop-down list:


Note that nothing in the document is modified except the `DOCTYPE` declaration. Also, you should revalidate the document whenever switching from one DTD to another.

## Displaying a Non-UTF-Encoded Document on Phone SDKs

The following procedure describes how to open and display an XML-based document that is encoded in a format other than UTF-8 or UTF-16.

1 Within NMIT, choose **Edit>Preferences** to display the **General** tab in the Preferences dialog. In the **Default Input Text Encoding** list, select the encoding that corresponds to that used in the document you wish to open and display.

2 Choose **File>Open** to open the document in an NMIT editor.

3 Choose the **Options** button and then choose the **UTF-8** radio button as the value for the **Output Encoding**.

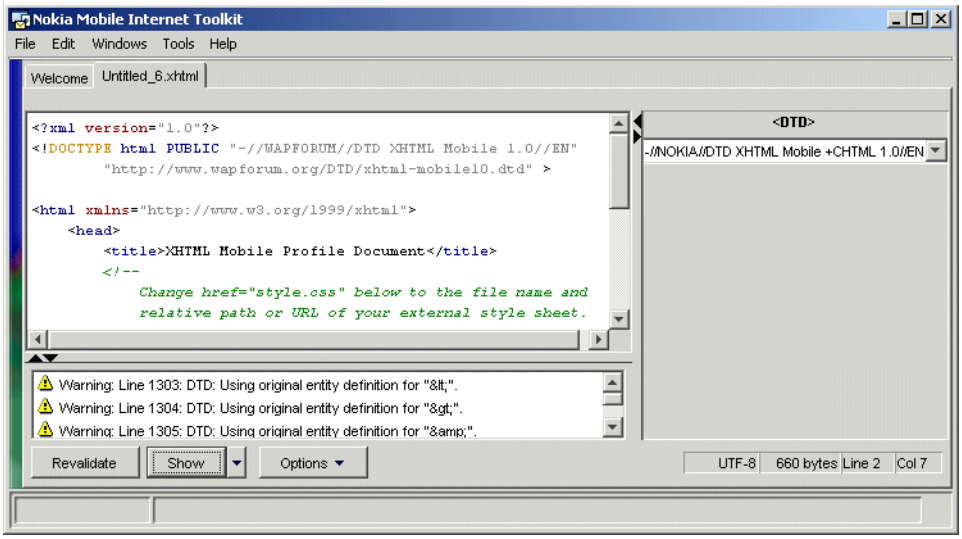
4 Edit the XML prologue in the document (the first line), if necessary, so that the `encoding` attribute specifies a value of `UTF-8`; for example:

```
<?xml version="1.0" encoding="utf-8"?>
```

5 Press the **Show** button to display the document on the selected phone SDKs. NMIT converts the encoding when it saves the document prior to its display on the phone SDKs.

## XHTML–MP

XHTML-MP




The XHTML-MP content type references the XHTML Mobile Profile 1.0 DTD. This DTD is the generally recognized standard DTD for XHTML content for mobile devices. Nokia recommends using this content type when creating XHTML-based content. The required elements of this DTD are described below.

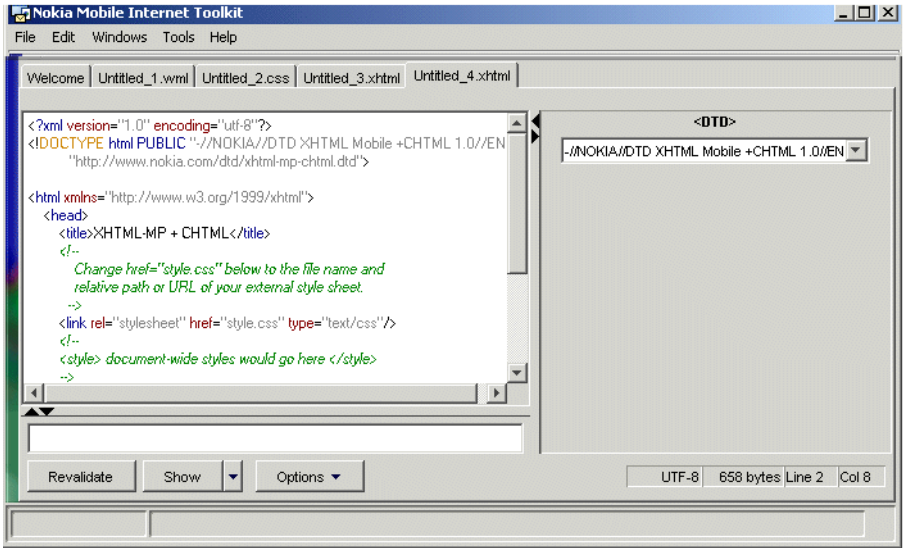
The first XML statement `<?xml version="1.0"?>` is required by all XML documents. The second `<!DOCTYPE>` statement specifies the DTD Public ID and the URL of its Internet location.

The third element is `<html>`, and the fourth is `<head>`. The last required element is the `<body>` element within which you enter the bulk of your document content.

Editing features available while editing XHTML-MP content are described in the section titled Editing Features For Validatable Text Content.

## XHTML–MP+CHTML

XHTML-MP+CHTML




The **XHTML-MP+CHTML** content type contains all XHTML Mobile Profile DTD elements plus the following additional ones:

| | | | |
|---|---|---|---|
| `<center>` | `<dir>` | `<marquee>` | `<font>` |
| `<menu>` | `<plaintext>` | `<blink>` | |

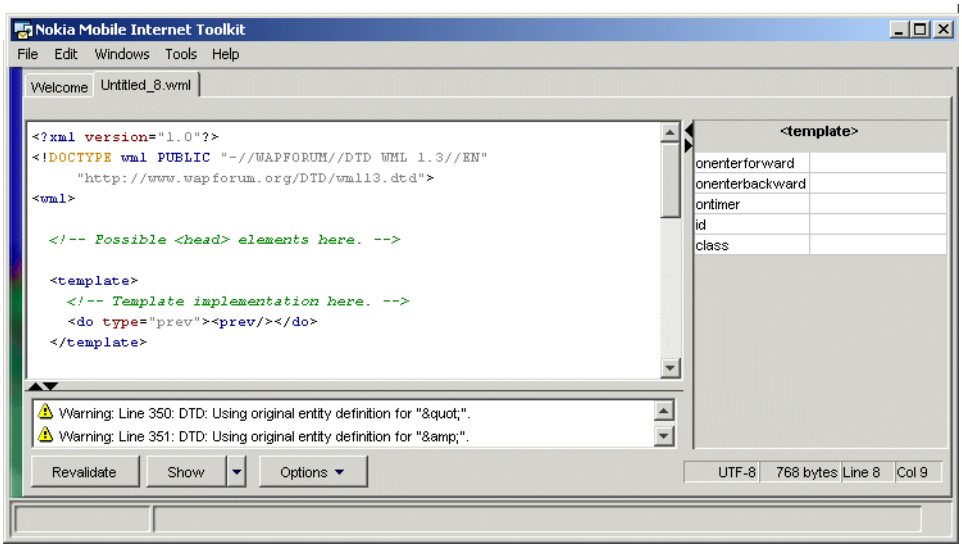
The **XHTML-MP+CHTML** content type contains all XHTML Mobile Profile DTD attributes plus the following ones:

| | |
|---|---|
| `href="tel:<phone-number>"` | In the `<a>` element, the `href` attribute accepts the `tel:` scheme. When the link is selected, the number specified in `tel:` can be directly dialed or stored in the phone book, as the user chooses. |
| `accesskey` | In the `<a>`, `<textarea>`, and `<input>` elements, creates an access key shortcut. |

Editing features available while editing XHTML-MP+CHTML content are described in the section titled Editing Features For Validatable Text Content.

## WML 1.3 Deck

WML Content Editor




By default, the NMIT WML editor references the WML 1.3 DTD. This DTD is the generally recognized standard DTD for XHTML content. The required elements of this DTD are described below.

The first XML statement `<?xml version="1.0"?>` is required by all XML documents. The second `<!DOCTYPE>` specifies the DTD Public ID and the URL of its Internet location.

The third required element is `<wml>`. Within the `<wml>` element, the `<head>` and `<template>` elements are optional.

DTDs for WML versions 1.1 and 1.2 are also registered for use in NMIT editors, and you can change the WML 1.3 DTD to 1.1 or 1.2 by placing the cursor in the `<!DOCTYPE>` declaration and selecting either of these from the DTD drop-down selector in the side panel, as described in the section titled Validating the Same Content Against Different DTDs.

Editing features available while editing WML content are described in the section titled Editing Features For Validatable Text Content.

**Note:** The WML compiler does not support the following encodings: Windows-1252, ISO-8859-4, ISO-8859-8, GBK, BIG5 or Shift-JIS. So, if any of these are used in your WML document, choosing **Save Binary** causes a failure with compilation errors.

## CSS

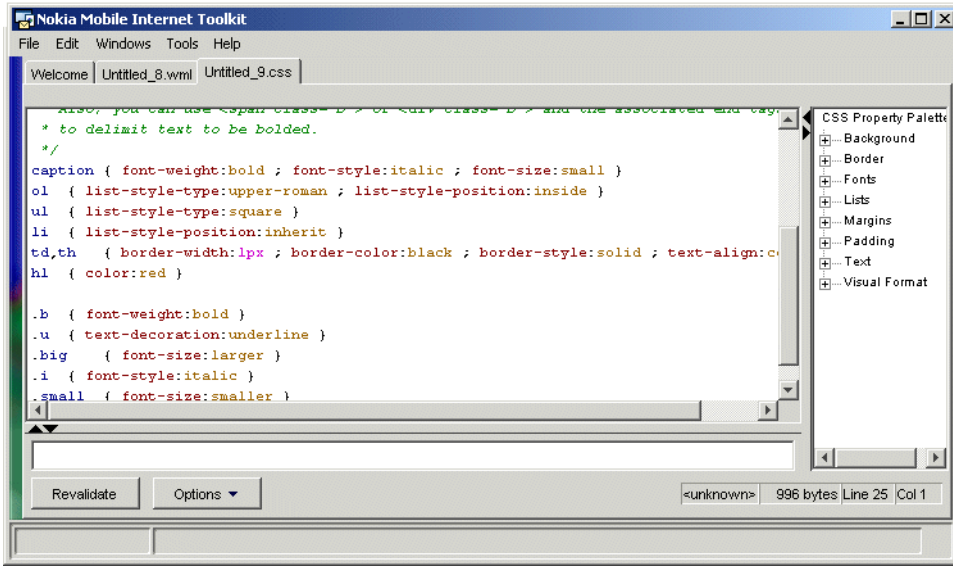
Wireless CSS is a style sheet language designed specifically for mobile devices; it is a subset of *Cascading Style Sheets Level 2* (CSS2). Many browsers in mobile handsets have the capability to use style sheets to apply formatting to XHTML documents.

You use the CSS Editor to create an external style sheet (a CSS file) that contains formatting rules to be applied to an XHTML document. You then reference the CSS file within the XHTML document using the `<link>` element within the `<head>` element. For example, assuming you have created the CSS file `style.css`, your XHTML document might contain the following code:

```
<head>
<link rel="stylesheet" href="style.css" > </head>
```

Because the rules in a CSS style sheet apply to elements or a group of elements within the XHTML document, modifying a style sheet is an easy way to change the appearance of that document. The separation of content from formatting makes it possible to display a single XHTML document on multiple phone handsets or SDKs, no matter how different their display sizes or other characteristics may be. This is done by simply creating a separate CSS file for each output handset or SDK. In fact, using CSS, one could change the look and feel of an entire mobile portal simply by modifying its CSS file, leaving the content itself unchanged.

The CSS editor is shown in the following figure




Note the following with respect to the text editing region in the above figure:

- The following color coding is used in the text editing region:

| | | |
|---|---|---|
| Green | Comment | The sample style sheet contains a comment showing the proper syntax for comments. It also contains information about CSS syntax. |
| Blue | Selector | A selector is either (1) an XHTML element name or (2) a class attribute name. Class attribute names are preceded by a period (.)<br>The selector specifies the entity to which the CSS property should be applied. |
| Red | Property | CSS property name. |
| Tan | Property value | CSS property value. |

- The following color coding is used in the CSS Property Palette region:

| | | |
|---|---|---|
| Black | Property category | Name of a property category. These names are not part of the CSS specification but are used as a way to organize the many property names to make them easier to locate. |
| Red | Property | CSS property name. |
| Blue | Property value | CSS property value. |
| <Gray> | Property value | Property value defined by the CSS specification. Some property values are not CSS keywords but rather text or number values that you must enter. For example, you enter the name of a font family in the `<family-name>` value for the `font-family` property name. In the CSS Property Palette region, any value within angle brackets ("<" and ">") is a value that you type in rather than select. |

- A sample style sheet is displayed in the text editing region. This style sheet is included simply to give an idea of the correct syntax for entering rules (see the section below titled CSS Style Sheet Design and Syntax). You could, therefore, safely delete the sample style sheet and start from scratch; or, only keep those XHTML elements you intend to modify, deleting the rest.
- Right-click popup menus, the message region, active buttons, and document summary areas of the editor are described in the section Editing Features For Validatable Text Content.

## Inserting CSS Property Values

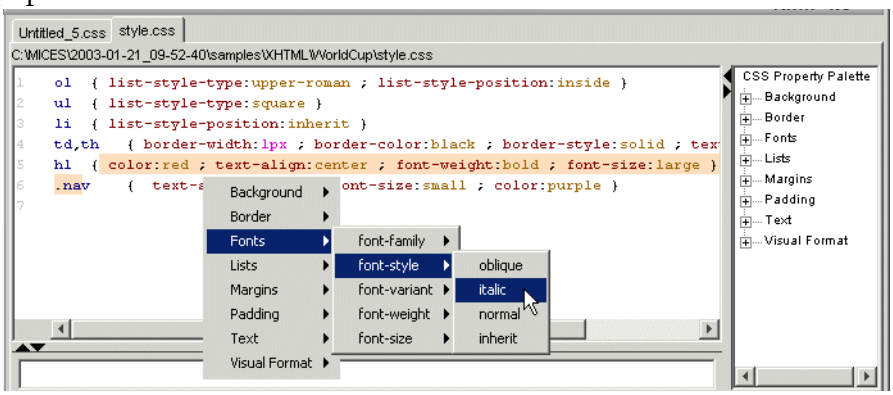
The first step in formatting a particular element is to type the selector name, followed by the required braces. Then, once this syntactical structure is in place, you can enter formatting properties for that element in the three ways described below.

1 **Direct text entry.** Within the braces, you can type in a property name, a colon (:), and then the property value. This is the manual method and enables maximum control. It may be helpful to use the CSS Property Palette to view the list of property values you can select for the property you wish to specify.

2 **CTRL-K Combination.** Within the document editing window, with your cursor positioned anywhere on a line containing a selector and within the required braces, press **CTRL+K** to display a popup menu list of property categories.

   From within the desired property category, navigate to the desired property and property value by using the arrow keys or the mouse, and then press **ENTER** or click the left mouse button. The selected item is inserted properly within the braces on the line containing the desired selector.

For example, in the following figure, the cursor is positioned in the document window on line 5 (`h1`) when **CTRL+K** is pressed. Navigation to the `italic` value in the popup menu is shown.



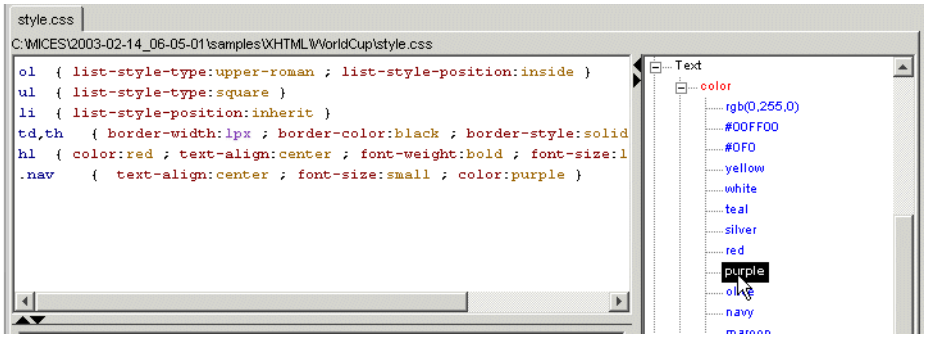

After releasing the mouse button, the CSS editor inserts the selected property and property value within the braces on the line where the `h1` selector is specified, as shown below:

```
5   h1  { color:red ; text-align:center ; font-weight:bold ; font-size:large
6    font-style:italic} {  text-align:center ; font-size:small ; color:purple }
```

3 **Property Palette.** Within the document editing window, with your cursor positioned anywhere on a line containing a selector and the required braces, explore the **CSS Property Palette** region along the right of the CSS editor window.

- Click one of the property categories to display the available properties within a property category.
- Expand the view of a property to view its available property values.
- Double-click a property value (or press **Enter**) to insert the property name and value into the document at the cursor location.

For example, in the following figure, the cursor is positioned in the document window on line 6 (on the line with selector `nav`). In the CSS Property Palette pane, navigation to `text`, then `color`, then `purple` is shown. Double-clicking `purple` inserts the property/value pair `color:purple` within the braces for selector `nav`.




Since the CSS language is not based on a DTD, there is no concept of a strictly conforming style sheet. Therefore, the **Options** menu does not offer the **Validate to DTD** or **Show Warnings** options. Revalidation is done only to validate CSS syntax.

Any style can theoretically be applied to any document; however, mobile handsets or phone SDKs may ignore unsupported styles.

## CSS Style Sheet Design and Syntax

To create a usable style sheet for your XHTML document, you must first decide upon the XHTML elements whose format you wish to control. This depends, naturally, upon which XHTML elements you have used in your XHTML document. Then, you create a rule for this selector using the following syntax:

```
selector { property: value; }
```

`selector` — In the simplest case, a selector is the name of an XHTML element (or group of elements) to which the rule applies. Use a comma (,) to separate multiple elements.

However, a selector can also be the value of an `id` or `class` attribute or other more complex expression. In this case, a period precedes the selector name as shown in the following line:

```
.p {font: courier new}
```

`property` — A property is a formatting element for which you can choose a value. For example, `font-weight` is a property, some of whose values are `normal`, `bold`, and `bolder`. The following rule specifies that all `<h1>` elements are to be formatted using a bold font weight:

```
h1 {font-weight: bold}
```

You can specify multiple properties for a selector; to do so, use a semicolon (;) to separate each property/value pair. For example, we can add a `font-size` property value to the `h1` selector as follows:

```
h1 {font-weight: bold; font-size: large}
```

All property/value pairs assigned to a given selector are enclosed within a single set of braces.

`value` — You may assign one or more values to a property, though it is usually only one. Use a comma to separate multiple values. When multiple values are specified in a rule, the browser will use the first value that it can implement, evaluating the rule from left to right. So, for example, a browser processing the following rule will attempt to reduce the font size to 30% of the original; if this is not possible, it will apply the `smaller` value.

```
h1 {font-weight: bold; font-size: 30%,smaller}
```

In CSS, subsequent rules override previous ones (assuming they apply to the same element); this is the meaning of *cascading* in cascading style sheet. A browser applies its default style sheet to a document; then applies those in an external style sheet (referenced using a `<link>` element within the `<head>` element); then applies styles on an element by element basis for each `<style>` element specified within the document body. So the processing order is (1) default browser style sheet, (2) external style sheet, and (3) individual `<style>` elements, in order of increasing precedence.

CSS language rules can be complex. It is recommended to read the CSS specification for a complete language description, as well as Nokia's *XHTML Guidelines*, which provides guidelines for the effective use of CSS styles. The latter document can be downloaded from Forum Nokia.

# Other Browsing Editors

NMIT provides the following two editors for creating content that is used in support of WML and XHTML content:

A WML Script editor for creating or modifying WML Script content, which is used to provide scripting capability from within a WML document.

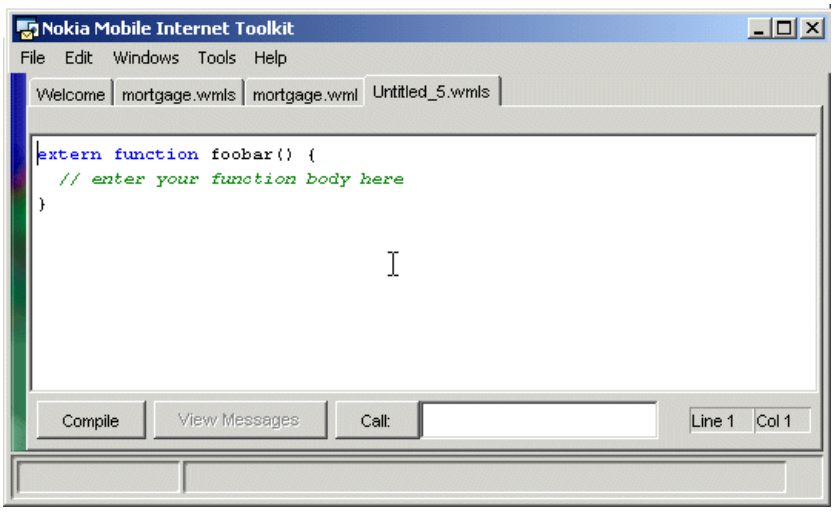
A WBMP Image editor for creating and editing WBMP images, and for converting images in GIF and JPG format to WBMP format.

## WML Script

The WMLScript Editor is opened whenever you take any of the following actions:

- Choose **File>New>WMLScript.**
- Choose **File>Open** and then choose a `wmls` file.

When creating a new document, the WMLScript Editor opens displaying a blank document as shown below.




After coding your function, you compile the code by choosing the **Compile** button.

After choosing **Compile**, a message region is opened beneath the editing window. Within this region, compile-time errors and status messages are displayed. You can hide these messages in order to restore the full text editing window by choosing the **Hide Messages** button, and after that, view the messages again by choosing the **View Messages** button, thus toggling between the two views.

## Executing a WMLScript Function

When your compilation is successful, you can test your function in the following ways:

- Open the WML document that calls the function defined in your WMLS document and then display the WML document on the phone SDK by choosing **Show**.

  All Nokia phone SDKs have a Diagnostics panel containing a **Current** view. Most phone SDKs also support the viewing of variables within the Current view. Using this view, you can view and modify variables used in your WML Script document and then reload the content again.

  Showing the WML document on the phone SDK enables you to see the results of both your WML and WMLScript content on the phone SDK display itself, as well as to observe the interaction between the two files in a browser context.
- Choose the **Call** button in the WMLScript editor. Choosing **Call** opens the **Calling Script** dialog into which you enter a call to one of the functions defined in the WMLScript currently displayed in the editor. Alternatively, you can type the call in the text box to the right of the **Call** button and then press **Enter**.

## Debugging With the tk_result Reserved Variable

The WML Script editor reserves the variable name `tk_result` for your use in testing. You may find this useful (1) if the phone SDK to which you display content does not have a Diagnostics view within which you can inspect variables or (2) you wish to test the value of a variable whose value may change during the course of script execution, almost as you would test using a breakpoint. You set the `tk_result` variable at the desired location within your WML Script document. Then when you call the function, its value is displayed in the phone SDK.

The following example shows the use of the `tk_result` reserved variable within the `mortgage.wmls` WMLScript sample file (available in the NMIT sample folder).

1 Open the `mortgage.wmls` document in the WML Script editor. The intermediate variable we wish to examine is the `tmp` variable, which is used as follows:

```
var tmp = Float.pow((1 + mi), num_payments);
payment= principal * (mi * tmp / (tmp - 1));
```

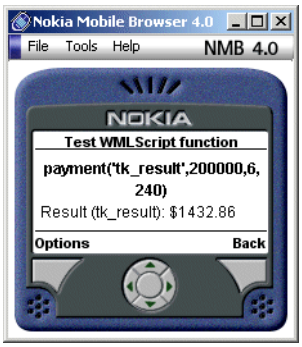
2 Enter a new `WMLBrowser.setVar` statement just below the line containing the `tmp` variable we are interested in, as shown below. This statement will load the value of `tmp` at that point into the reserved `tk_result` variable.

```
var tmp = Float.pow((1 + mi), num_payments);
WMLBrowser.setVar('tk_result',tmp);
payment= principal * (mi * tmp / (tmp - 1));
```

3 Call the `payment` function, which is the name of the function defined in the `mortgage.wmls` document, by entering the following in the **Call** text box and then pressing **Enter**:

```
payment('tk_result',200000,6,240)
```

4 The value of the `tk_result` variable is displayed in the phone SDK, as shown below. We now know the value of the `tmp` variable at that point in the execution of the WML Script document.




### WML Script Editor Controls

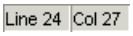

Compiles the WMLScript source file (file extension `wmls`) into an executable WMLScript file (extension `wmlsc`) in the same directory. The compiled file is removed when NMIT is closed.


After choosing **Compile**, compilation messages are displayed by default. Choose **Hide Messages** to restore the full text editing window and **View Messages** to open the message display again. These buttons allow you to toggle between viewing and hiding messages.


Choosing the **Call** button displays the **Calling Script** dialog. Enter the name of the function to call, as well as any required arguments, using the proper function call syntax, and choose **OK** to execute the call. NMIT calls the function with the arguments you specified by generating a dummy WML page and displaying it on the selected SDK.

Alternatively, enter the function call into the text box to the right of the **Call** button and press **Enter** to execute the function.

The above two methods of entering a function to call are equivalent.



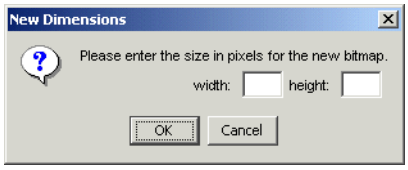

Display showing the current location of the cursor within the code editing area. The location is given as a line and column number.

## WBMP Image

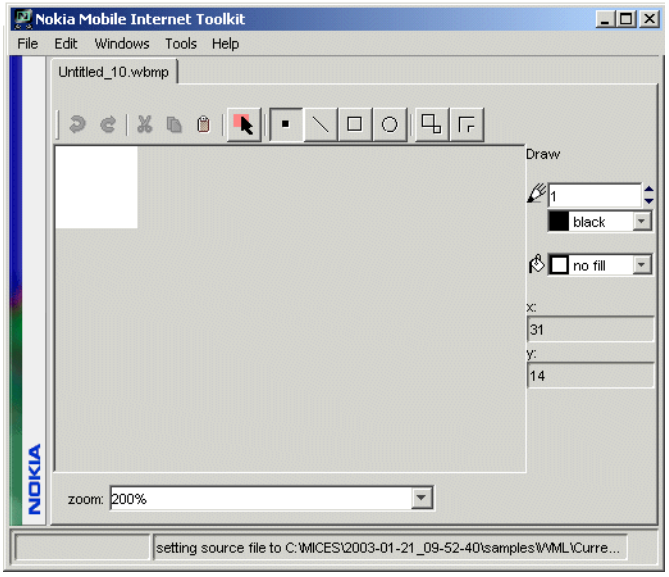
The WBMP Image Editor is displayed whenever you take any of the following actions:

- Choose **File>New>WBMP Image**.
- Choose **File>Open** and choose a file of any of the supported file types: WBMP, GIF, JPG. Note that, if you choose a file of type GIF or JPG, NMIT converts it to WBMP format before displaying it.

NMIT always displays the **New Dimensions** dialog whenever you choose to create a new WBMP image, as shown below:




A reasonably sized image for display on a phone SDK is 32 pixels by 32 pixels. Entering this value and then choosing OK displays the WBMP Image Editor main window, shown below:




The WBMP Image editor opens with a highlighted drawing region, sized according to the dimensions you entered in the **New Dimensions** dialog. A 32 x 32 pixel drawing region is rather small, so you may wish to enlarge it by using the **Zoom** drop-down list box at the bottom. You can choose to enlarge the drawing region as much as 5,000%.

The WBMP Image editor uses the following modal controls. Switch from one control to another by selecting a Toolbar button.

[WBMP Editor Toolbar Buttons]

[WBMP Editor Draw Modal Control]

[WBMP Editor Rescale Image Modal Control]

[WBMP Editor Resize Canvas Modal Control]

## WBMP Editor Toolbar Buttons

The Toolbar buttons are displayed horizontally along the top of the WBMP editor window. They are divided functionally by space between each functional grouping. That is, there are the first pair (**Undo** and **Redo**), a space, the next group (**Cut**, **Copy**, **Paste**), a space, the next (Select region), a

space, the next group (pixel, line, rectangle, ellipse), a space, and finally the two modal control switch buttons.

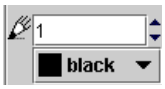
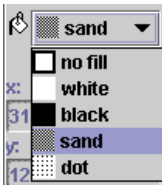
| | |
|---|---|
|  | Undo the previous action. You can undo the previous action, the action before that, the one before that, and so on, by choosing Undo multiple times. |
|  | Undo the previous undo action; that is, Redo. You can perform multiple Redo's. For example, you can restore the state of your drawing after choosing multiple Undo's by performing the same number of Redo's. |
|  | Cut the selected region and put it on the Clipboard. Use the  tool to select the region. |
|  | Copy the selected region to the Clipboard. Use the  tool to select the region. |
|  | Paste the contents of the Clipboard into an active select region. Use the  tool to select the region into which to paste the content. |
|  | Selects a rectangular region.<br>To use the tool, click the left mouse button at one corner of the desired region, drag the pointer to the opposite corner, and then release the mouse button. A red rectangle indicates the region you selected. Choose **Undo** if you wish to try again. To redraw a select region, click once to remove it and draw it again.<br>The Select tool is very useful:<br>You can remove anything in a select region by choosing **Cut**; so, for example, if the select region bisected a circle, choosing **Cut** would create a semicircle.<br>You can paste the contents of the Clipboard into a select area. |
|  | Draws a free-form line or inserts a single pixel.<br>To draw a free-form line: click at the start point, drag, then release at the end point.<br>To insert a pixel: position the cursor, then click and release. |
|  | Draws a straight line.<br>To draw a straight line: click at the start point and drag, then release at the end point. |
|  | Draws a rectangle.<br>To draw a rectangle: click at one corner and drag, then release at the opposite corner. |
|  | Draws a circle or ellipse.<br>To draw a circle or ellipse: click, drag in any direction to change shape or size, then release. |
|  | Displays the WBMP Editor Rescale Image Modal Control, at the right side of the window. |
|  | Displays the WBMP Editor Resize Canvas Modal Control, at the right side of the window. |

## WBMP Editor Draw Modal Control

The Draw Modal control region, to the right of the WBMP Editor window, is the default modal control. It is displayed when you first open the editor.

The Draw Modal control specifies settings for line weight and for fill color. These settings affect the behavior of each of the four drawing tools: , , , and . If the Draw Modal control is not displayed, choose one of the above drawing tools to display it.

### Draw Modal Control Options

| | |
|---|---|
|  | Sets the line weight and color.<br>To set the line weight: increase the line weight by clicking the up arrow, thereby increasing the displayed numerical value; decrease line weight using the down arrow. If you set a line weight of 0, the line appears invisible but still exists.<br>To set the line color: choose black or white from the drop-down list box. |
|  | Sets the fill pattern to be used when drawing a rectangle or ellipse.<br>To set the fill pattern: choose one of the five settings by choosing from the drop-down list. |

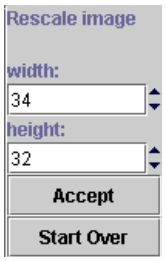
Choose a line weight and fill options before drawing a line or shape. The editor does not permit the changing of options after a line or shape has been created. So, for example, if you want a rectangle with a fill pattern, choose the fill pattern before creating the rectangle. You cannot use the selector tool to set a fill pattern for an already existing shape.

## WBMP Editor Rescale Image Modal Control

The Rescale Image control region, to the right of the WBMP Editor window, is displayed by selecting the Toolbar button.

Rescaling an image is akin to stretching or shrinking a canvas upon which a figure is already drawn. The figure stretches or shrinks along with the canvas, distorting the original figure in the process.

**Rescale Image Modal Control Options**




To rescale an image: adjust the height and width values (in pixels) of the existing image up or down, thus stretching or shrinking the image.

When you are satisfied with the result, choose **Accept**. If not, choose **Start Over**.

**Accept** has the effect of storing (but not to disk) the currently adjusted state of the canvas, establishing in effect a new starting point for further adjustment.

**Start Over** returns the canvas to its state at the time you last chose Accept. But you can only return to that point, never prior to it. The **Start Over** control is active only if you have made a change since you last chose Accept.

Remember to choose **File>Save** when you are done.

## WBMP Editor Resize Canvas Modal Control

The Resize Canvas control region, to the right of the WBMP Editor window, is displayed by selecting the  Toolbar button.

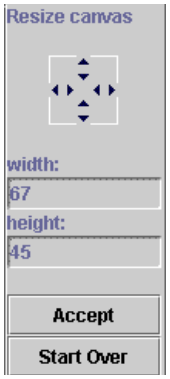
The canvas is the image area you specified at the time of creation in the **New Dimensions** dialog. Imagine the upper left corner of the canvas at position (0,0) on the (x,y) axes. The lower right corner is at position (width, height), where these values are given in the modal control display.

Resizing an image differs from rescaling in that it does not distort the image (by stretching or shrinking); instead, it involves adding or subtracting pixels to the edges of the canvas, thus changing the size.

Resizing an image is useful in repositioning a drawn figure on the canvas or cropping the figure, that is, removing a part of it. For example, you can reposition an image that is centered on the

canvas by increasing the canvas size on any edge. You can crop an image by decreasing the canvas size on one edge (sort of like rolling up a window shade upon which an image is drawn).

### Resize Canvas Modal Control Options




The Resize Canvas control uses controls (arrows) to increase or decrease the image along an edge, as follows:

- Decrease image width from the left or right edge.
- Increase image width from the left or right edge.
- Decrease image height from the top or bottom edge.
- Increase image height from the top or bottom edge.

When you are satisfied with the result, choose **Accept**. If not, choose **Start Over**.

**Accept** has the effect of storing (but not to disk) the currently adjusted state of the canvas, establishing in effect a new starting point for further adjustment.

**Start Over** returns the canvas to its state at the time you last chose Accept. But you can only return to that point, never prior to it. The **Start Over** control is active only if you have made a change since you last chose Accept.

Remember to choose **File>Save** when you are done

## Import and Convert JPG and GIF Images to WBMP Format

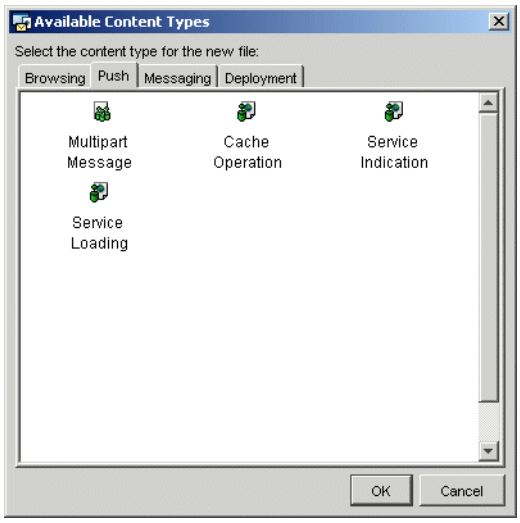
The NMIT WBMP editor can import and convert a JPG or GIF image to WBMP format. Since the WBMP format does not use color, any color in the JPG or GIF image is converted to black and white. A process called “dithering” is used to translate colors into different densities of black dots on a white background, thus producing different shades of gray.

1. Choose **File>Open** to open the desired JPG or GIF image. The image is opened in the WBMP image editor.
2. Choose **Save** or **Save As** from within the WBMP editor to save the file in WBMP format. Note that NMIT converts the image upon import; so you can modify the image within the editor before saving it.

# Push Editors

A Push Content Editor is opened in the NMIT editors window whenever you take any of the following actions:

- Choose **File>New,** choose the Push tab in the Available Contents Type dialog, select a Push content type, and then choose **OK.**




- Choose **File>Open** and then choose an existing Push content file. You can open both source files (of file types `si`, `sl`, and `co`) and encoded files (of file types `sic`, `slc`, and `coc`).

NMIT provides a separate editor for creating each of the three standard WAP Push content types (`si`, `sl`, and `co`) and a fourth for the special type `multipart`. These editors differ from NMIT

Browsing editors in that they are designed as Windows “wizards” in order to facilitate the entering of headers, and in the case of multipart, the inclusion of file parts.

**Push Content Editors**

| | |
|---|---|
| Service Indication (SI) Editor | Creates a message intended to be delivered to the phone SDK Push Inbox, not to the phone browser. The message serves as a notification to the user that new content is available for downloading. The user can then select the supplied URL to download the content. |
| Service Loading (SL) Editor | Creates a message intended to be used to force a user agent running on the client device to download new content without notifying the user. |
| Cache Operation (CO) Editor | Creates a message intended to be used to invalidate particular entries in the user agent cache, thus forcing a reload of content the next time the user requests content identified by those entries. |
| Multipart Message Editor | Creates a message consisting of one or more separate parts, with each part consisting of a header section and content body. The editor enables you to assemble a number of related pieces of content parts that can be pushed as a group in order to reduce network latency. |

# Overview of Push Messages

A Push message is an unsolicited message sent by a server to a client. In the network environment, a server application called a Push Initiator initiates the sending of a Push message. It sends this message via the Push Access Protocol (PAP) to a Push Proxy Gateway (PPG), which extracts the relevant client-bound content and encodes it, prepends the required content type header and any other included headers (all unencoded), and sends it using the WAP protocol over a wireless network to the destination mobile handset. The mobile handset decodes the message and, recognizing the content type as Push-related, dispatches it to the content handlers; for example, Service Indication (SI) content is dispatched to the handset Service Inbox. The user of the handset then views the message in the service Inbox and can retrieve the content by selecting the included URL.

In the NMIT environment, there is no Push Initiator and therefore no use of PAP or PPG; nor do messages traverse a wireless network. There is instead a Nokia-internal protocol whereby messages you create using NMIT Push editors can be delivered to phone SDKs running on the same local machine as NMIT. The aim is to enable you, working within NMIT, to construct Push messages just as a Push Initiator does, encoded and with appropriate headers, and to deliver these to phone SDKS in the same manner as a Push Initiator delivers Push messages to mobile handsets in the network environment. The delivered messages therefore are structurally identical to those received over the network.

NMIT provides Push message editors in order to enable you to observe on a phone SDK how Push messages you create would appear if deployed over the network to mobile handsets. This is

possible because phone SDKs are usually designed to behave as their namesake mobile handsets. Not all SDKs or phones support Push. Check the documentation.

In addition, supported Nokia phone SDKs employ a Diagnostics Panel through which you can view Push content details such as headers which would have been present had the message been transmitted over the network through a WAP gateway. The ability to view headers received by the phone SDK facilitates your design of custom headers which you might, for example, use in designing a new Push service.

## Encoding and Header Handling in Push Messages

Phone SDKs expect to receive only WAP-encoded Push message content. Therefore, messages that you push to phone SDKs are always encoded by NMIT.

When you store a Push message in binary format (**Store Binary** button), NMIT encodes the message content only (as do WAP gateways). You can see this by opening such a message in a binary editor (such as **hexedit**) and checking the content against the WAP SI specification. Also note that the binary file format does not include any Push message headers.

However, when NMIT saves Push messages as source (**Store Source** button), the headers are included as standard HTTP headers. For this reason, you should save Push messages that you may wish to edit again in source format.

According to Push specifications, headers received from a Push Initiator are passed through to the receiving mobile handset unencoded, but they are not required to be available for user viewing. So, for example, if a Push Initiator generates a message and includes the `X-Wap-Application-ID` header (to specify a handset application that is to process it), the WAP gateway will pass this along, unencoded. The handset software processes the message, using header information as necessary (for example, the `X-Wap-Application-ID`), decodes the encoded message content, and dispatches it to the handset's Service Inbox. When the user views the message, he views only the message content; headers received by the handset and using in processing have been discarded and are not available to the user.

However, in the NMIT environment, which is a developer environment and not a user environment, all headers must be preserved and accessible after the message has been received in the Service Inbox, including user-defined headers. To accomplish this, NMIT packages these headers separate from the message content so that a user of a phone SDK may view these using the SDK's Diagnostics Panel.

## Saving Push Messages

Like other types of WAP content, a Push message has both an encoded (binary) and a text (source) form. In any of the three Push editors, you can choose to store the message as text by choosing **Store Source** or as binary by choosing **Store Binary**. In either case, you will be able to later open and edit the message within a Push editor. Note, however, that NMIT stores Push message headers only when you store the message in source format.

You might choose to store as text (**Store Source**) when you are creating a message to use as a template or model for a message that you intend your Push Initiator to generate programmatically.

You might choose to store as encoded (**Store Binary**) when you are ready to have your Push Initiator (perhaps implemented as a servlet inside a WAP Push Gateway) read the encoded content, generate the required headers dynamically, and then send it to a client directly.

# SI, SL, and CO Editors

SI, SL, and CO messages are structurally similar: each consists of HTTP header information and a single XML-based content body. They differ principally in their intended uses within an application; they each use a different DTD and different XML elements.

The editor for each opens with the selected content type already entered for you in the required HTTP header field. You must enter data in the required fields, which are highlighted within the editor. You may also add optional headers and content, but these optional fields are hidden from view until you choose to display them using a control button.

The following sections describe the SI, SL, and CO editors:



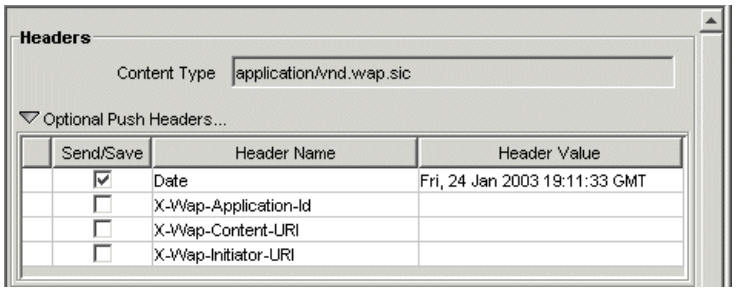
## Common Features

### Headers

All three Push message editors display a region titled **Headers** which displays the headers to be included with the message content.

The appropriate value for the `Content-Type` header is automatically inserted within each editor and cannot be modified. (Note that all non-editable fields are colored the same as the window background color.) The value displayed is based on the editor chosen: `application/vnd.wap.sic` for SI messages, `application/vnd.wap.slc` for SL messages, and `application/vnd.wap.coc` for CO messages.

You may specify optional push headers by clicking the ▷ button adjacent to the label **Optional Push Headers**. Some standard WAP Push headers are then displayed as shown below:




| | |
|---|---|
| `X-Wap-Application-Id` | Destination application ID. An identifier agreed upon by a server application and a client device which instructs the device browser to pass the content to a client-side application for processing. |
| `X-Wap-Content-URI` | Virtual URI of the message. Messages in the cache with this header will generate cache hits when other requests specify a matching URI. |
| `X-Wap-Initiator-URI` | URI of the Push Initiator Application, which is the application that generated the Push message. |

To enter a value for a **Header Name**:

- Click in the adjacent **Header Value** field and type the desired value.

To add a new line on which you can enter an additional header name and value:

- Click anywhere on the last line until a new line is displayed.

To delete a header line:

- Click in the left column of the line (to select it) and then press the **Delete** key.

To specify that NMIT send the header:

- Click in the **Send/Save** column to place a check mark on the header line.

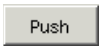
To specify that NMIT not send the header or to save the header when you save the message:

- Click in the **Send/Save** column to remove the check mark on the header line. This feature eliminates the need to delete and then re-enter values for individual test cases.

Note that, when you save a Push message in source format, NMIT includes only those headers for which there is a check mark in the adjacent Send column. (No headers are stored if you save in binary format, regardless of check marks.)

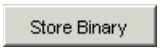
## Active Buttons

The SI, SL, and CO Editors all display the following three active buttons along the bottom of the editor window.




Pushes the current message to the phone SDK you have launched from the SDK Control Panel.

How the phone SDK responds to the reception of a Push message depends on the SDK software, as well as on the nature of the message. Processing requirements are described in the various WAP specifications for Push Content. SI messages, for example, are displayed in the phone SDK; CO messages may delete expired messages from the local cache; SL messages may cause the loading and execution of a specified service.



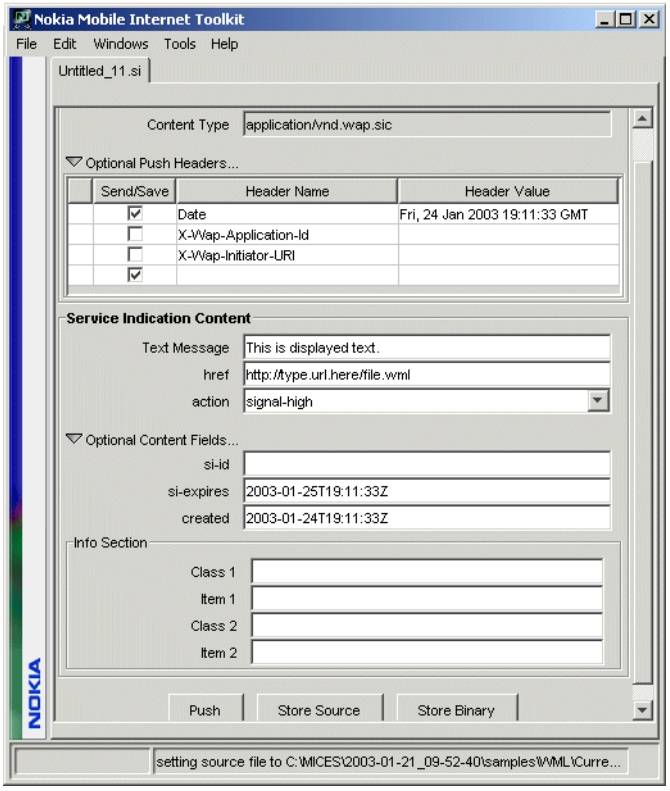

Opens the **Save As** dialog in which you can name and save the newly created message, excluding headers, in encoded format. The default file extension is `sic`, `slc`, or `coc`, which are all encoded file types.

Note that if you open an encoded file in a Push editor, headers are not displayed because encoded files are not intended for further editing in NMIT. They may, however, be examined in another editor to view the encoding.

By default, NMIT editors save files in the `\samples` directory, though you can change the directory from within the **Save** dialog.

## Service Indication (SI) Editor

The SI Editor consists of two window regions (Headers and Content) and a set of active buttons along the bottom of the window.



The **Service Indication Content** region of the SI Editor dialog contains the following fields.

**Content Fields in an SI Message**

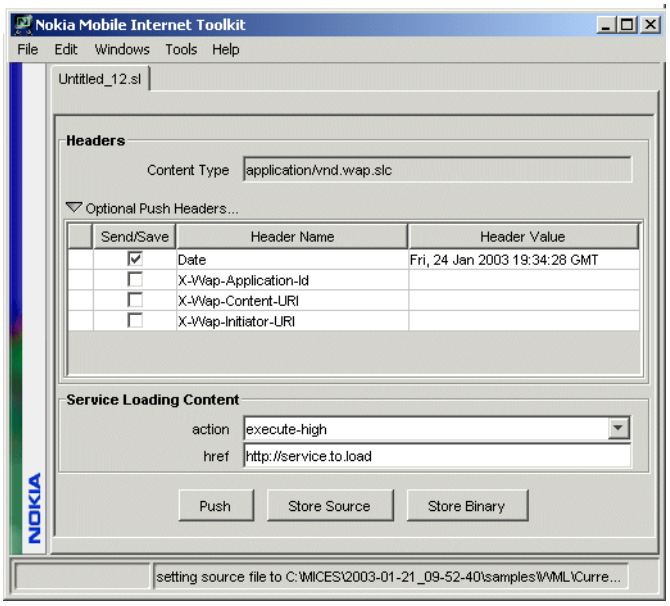
| | |
|---|---|
| **Text Message** | Text to be displayed on the phone SDK. For example, in a mail application, the text message "You have new mail" might be displayed on the user's phone, typically providing the user a means for fetching the mail via the URL specified in the `href` field. |
| **href** | URL to be fetched when the user chooses to process this message. For example, in a mail application, the URL specified by `href` might specify the location from which the user's mail may be downloaded. |
| **action** | Drop-down list box displaying five possible actions to be taken by the user agent at the time this SI message is received on the terminal device. The WAP specifications describe the precise behavior required or suggested by any of the following values: (1) signal-high, (2) signal-medium, (3) signal-low, (4) signal-none, (5) delete. Generally, these values specify the level of priority or urgency to be adopted by the user agent in processing the message, where signal-high is most urgent. |

**Optional Content Fields in an SI Message**

| | |
|---|---|
| **si-id** | Unique ID of this SI message.<br>This is used by the phone SDK to eliminate duplicate messages. |
| **si-expires** | Expiration date of this SI message.<br>The date-time format is **YYYY-MM-DDThh:mm:ssZ** where the "T" and the "Z" are required literal characters. See the WAP Service Indication Specification for details. |
| **created** | Creation date of this SI message.<br>The date-time format is the same as for the `si-expires` field (see above). |
| **Info Section** | The `<info>` element provides a way to specify additional information not provided by the attributes of the `<indication>` element. The element contains one or more `<item>` elements that specify the additional information. Each `<item>` element contains a `class` attribute describing what information the content of the `<item>` element contains. How a client uses this information is implementation dependent, and a client may discard the `<info>` element. Currently, Nokia SDKs do not use or support `SI <info>` sections. |
| `Class 1` | Class attribute describing the nature of the content specified in Item 1 (below). For example: `DogType` in the example above. |
| **Item 1** | Content identified by the Class attribute (above). |
| **Class 2** | (same as Class 1 above). |
| **Item 2** | (same as Item 1 above). |

## Service Loading (SL) Editor

The SL Editor consists of two window regions (Headers and Content) and a set of active buttons along the bottom of the window. The Headers window region is described in the section titled Common Features, and the active buttons in the section titled Active Buttons.




The **Service Loading Content** region of the SL Editor dialog contains the following fields:

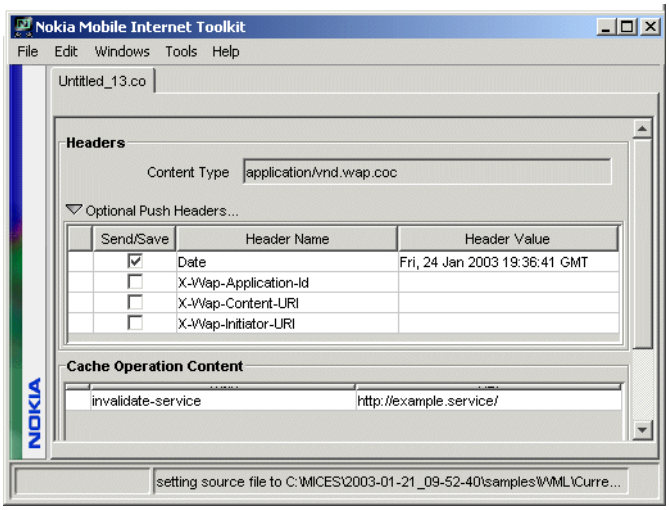
### Content Fields in an SL Message

| | |
|---|---|
| **action** | Drop-down list box displaying three possible actions to be taken by the user agent at the time this SL message is received on the terminal device. The WAP specifications describe the precise behavior required or suggested by any of the following values: (1) `execute-high`, (2) `execute-low`, (3) `cache`. The first two values specify the level of priority or urgency to be adopted by the user agent in processing the message, while the third specifies that the fetched message should be placed immediately in the local cache. If a value is not specified, `execute-low` is used. |
| **href** | (Required) URL of the service to be loaded and processed by the user agent. For example, a service provider might use an SL message to load a WML deck that notified the user that his balance had not been paid. |

Because SL messages can be used n denial-of-service attacks, some devices block this content type.

# Cache Operation (CO) Editor

The CO Editor consists of two window regions (**Headers** and **Content**) and a set of active buttons along the bottom of the window. The **Headers** window region is described in the section titled Common Features, and the active buttons in the section titled Active Buttons.




The **Cache Operation Content** region of the CO Editor displays a table with three columns (the first is quite narrow) and a single data entry beneath. Use this region to specify one or more invalidation directives. There must be at least one such directive in a cache operation message.

### Cache Operation Content Fields

Clicking in this empty area (not the column heading) in this left-most column selects the entire line. After it is selected, you can press the **Delete** key to delete the entire line.

Note that you cannot delete all the lines because a CO message must have at least one invalidate directive.


Clicking in the **Type** column (not the column heading) displays a popup selection box that enables you to toggle between the two CO directive types, **invalidate-service** or **invalidate-object**, thus changing the entered directive.

When a CO message containing an **invalidate-object** URI is processed by the user agent, any object in the local cache whose URL exactly matches the URI of the invalidate-object item is deleted from the cache.

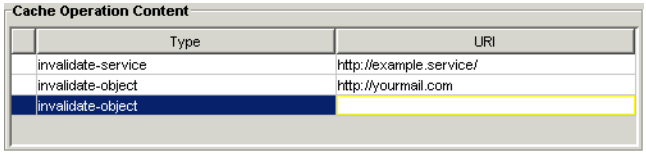
When a CO message containing an **invalidate-service** URI is processed by the user agent, any object in the local cache whose URL matches the URI prefix of the invalidate-service item is deleted from the cache. This item is useful therefore for removing multiple objects from the cache, all of which may have originated from the same service.

See the WAP specification for Cache Operation for detailed information regarding the purpose and use of these items.

### Cache Operation Content Fields

| | |
|---|---|
| URI | Clicking in the **URI** column (not the column heading) causes that item to become editable. You can add a URI to an empty entry, modify an existing entry, or use select, copy, and paste to change the entry to the desired value.<br>Note that typing into the URI column of the last row adds an empty row below so that you can enter as many directives as required. |

The following figure shows a Cache Operation Content dialog in which an additional directive has already been added, resulting in a new row below. Lines with blank URI fields are not included when the message is sent or saved.




# Multipart Message Editor

A multipart message is a single message composed of an ordered list of messages (or parts), each of which has a set of HTTP headers and a content body. The parts may be of any content type.

In WAP environments, the multipart format is used to enable the delivery of a set of related content parts to a device all at once in order to avoid multiple network requests, thereby reducing latency and thus the user's perception of the application response time. A phone SDK that receives a Push multipart message loads the first part and stores the remaining parts in cache. Multipart messages would, in real world applications, typically be dynamically generated on a content server either in response to a request or as server-initiated pushed content.

When you use the Multipart Editor to create a multipart message, you are actually assembling already existing content (in separate files) into an ordered package of parts and then specifying headers for both the entire package and for the individual parts.

This section describes how to quickly create a multipart message; the format, processing, and saving of multipart messages; and finally the use of the editor's display regions and button controls:

Quickly Creating a Multipart Message

Multipart Message Processing

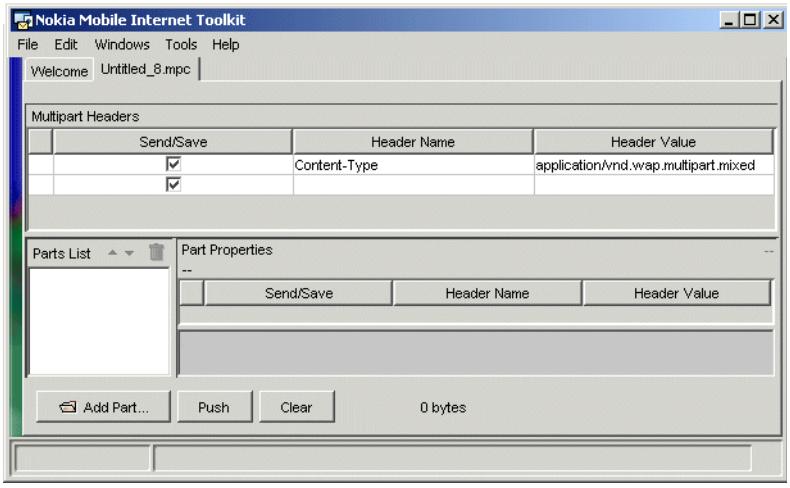
Editing Multipart Headers at the top of the window.

Including Parts region to the left.

Part Properties With Content to the right.

Multipart Button Controls along the bottom to the left.

This is the Multipart Message Editor:




## Quickly Creating a Multipart Message

The section titled Multipart Message Processing describes the nature and purpose of multipart messages. The task summary described here gives the steps used to create and display a multipart message containing two parts: a Service Indication (SI) push message and the WML file referenced by the SI message. The purpose of the multipart message is to enable the user, who has just read the SI message, to immediately view the accompanying WML message without experiencing network latency. This is possible because the WML file is as part of the Push message and is stored in cache.

1. Create a Service Indication Message. Choose **File>New**, then the **Push** tab, then **Service Indication**, and then **OK** to open the Service Indication editor. In the **Text Message** field, enter the text: `A weather update is available. View it now?` In the **href** field, enter the text `http://weather.com/today.wmlc`. Choose the **Store Binary** button. Name the file `Weather-Alert.sic` (be sure the file extension is `sic`) and store it in the `/samples/Push` directory.

2. Create a WML deck. Choose **File>New**, then **WML 1.3 Deck**, then **OK** to open the WML Deck editor. Choose **File>Save As** and save the file to the same `/samples/Push` directory with the name `today.wml`. Enter (or cut and paste) the following text into the newly created file:

```
<wml>
  <card title="Weather Update">
    <p>Today's weather is cloudy, with a 70% chance of showers.</p>
  </card>
</wml>
```

   Choose **Revalidate** and fix any parsing errors. Then choose **Show** to compile the file and display it on a phone SDK. NMIT encodes the WML file and stores it in the same directory with the file extension `wmlc`.

3. Create the Multipart file. Choose **File>New**, then the **Push** tab, then **Multipart Message**, and then **OK** to open the Multipart Message editor. Include the

encoded versions of the two files you just created by choosing the **Add Part** button; navigate to the directory in which the files are located, select them both, and choose **Open**.

4 Ensure that both files are displayed in the Parts List region in the following order: from top to bottom, first `Weather-Alert.sic` and then `today.wmlc`. If they are not so ordered, correct the order by selecting a file and using the up and down arrows (adjacent to the **Parts List** label) to position it.

5 Select the file `today.wmlc`. Edit its **X-Wap-Content-URI** header in the Part Properties region so its value is the URL `http://weather.com/today.wmlc`. (Note that some phone SDKs require that you specify this value in the **Content-Location** header.) Ensure the **Date** header has the value of today's date in standard HTTP format; for example, "Mon, 06 May 2002 11:00:45 GMT," without the quotes and in the required GMT time zone. Finally, add a **Cache-Control** header with the value `max-age=36000`; this ensures the file will remain cacheable for the next 10 hours.

6 In the **Multipart Headers** region at the top of the editor window, ensure the value of the **Content-Type** header is `application/vnd.wap.multipart.mixed`.

7 Choose the Push button to push the multipart message to a running phone SDK. In the phone SDK, you should see the text you specified in the `sic` file: "`A weather update is available. View it now?`" Then, when you choose to retrieve the content (and the way you do this may vary from SDK to SDK), you should see the content of the `wml` file you created, that is, the text "`Today's weather is cloudy...`"

## Multipart Message Processing

When the phone SDK receives a multipart message, it identifies the message type as a multipart message by the `Content Type` header and then proceeds to disassemble the message, separating out its constituent parts.

After disassembling the message, the phone SDK processes the first part immediately and puts any remaining parts into cache according to its cache management rules. After you push a multipart message, you can use the Diagnostics Cache view in the phone SDK to examine the contents of the phone SDK cache to verify that the parts are there.

The content type of the first part will determine whether or not anything is displayed on the phone SDK: if the first part is an SI content type, a notification is displayed; if a CO or SL content type, nothing is displayed; if a WML content type, nothing may be displayed, as the display of this content type is browser-dependent and not defined in the WAP Push specifications. For these reasons, the first part in most multipart messages will be of content type SI.

In normal WAP browsing, you request a URL and receive a response to that request. The browser places the response in cache and uses the request URL as the cache key for future requests to that URL. Now, a Multipart Push message is essentially a response without an associated request. WAP Push applications use the value of the header `X-Wap-Content-URI` as a virtual request URL in managing cache entries for the messages that have been pushed.

Any references in the content of the first part of a multipart message that cause a browser load request will also causes the browser to check its cache first. Such a reference might be an `href`

attribute specifying a URL an SI message. If the reference matches a cache entry (keyed with the `X-Wap-Content-URI` header value), then the resource is loaded from the cache and a network access is avoided. This mechanism is used by the browser to fetch and display the cached parts of a multipart message.

## Editing Multipart Headers

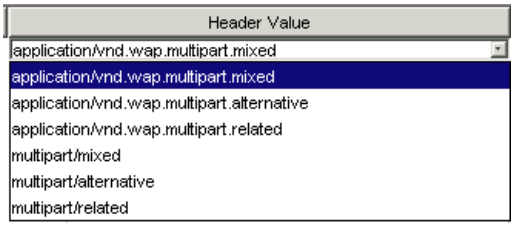
Multipart headers are distinct and unrelated to any of the constituent parts within the multipart message. Multipart headers serve a function much like the name on a file folder, describing the container rather than any specific part. Inside the folder, continuing the analogy, are the separate parts, each of which has its own headers.

Multipart is a MIME format which was initially created to enable the inclusion of various binary content types within a mail message which could be sent over the Internet. The MIME specification defines several multipart message types, three of which have been adopted by the Open Mobile Alliance and adapted for use in the WAP environment. You can create a message using any of the following MIME or WAP content types:

| MIME Format | WAP Format | Description |
|---|---|---|
| `multipart/mixed` | `application/vnd.wap.multipart.mixed` | This format is used typically for including various content types without specifying any relation between the parts. This is usually the default multipart message type used in WAP applications. |
| `multipart/alternative` | `application/vnd.wap.multipart.alternative` | This format is used typically for including two or more content types which are different representations of the same content: for example, content in both text and RTF formats. |
| `multipart/related` | `application/vnd.wap.multipart.related` | This format is used typically for including various content types which are related in some way. |

In the Multipart Headers region at the top of the editor window, you specify headers that apply to the entire multipart message. In this region, there are two columns titled **Header Name** and **Header Value**.

The first header name is the required `Content-Type`, whose value you choose from the drop-down list box in the adjacent Header Value column. These values are shown below:



- Choose one of the first three (encoded) types if you are assembling a multipart message targeted for display directly to a phone SDK. (Parts must be WAP-encoded first before being added to the message; NMIT does not perform any encoding.)

  After inserting headers, this message is ready to be used by any application capable of pushing content (such as NMIT or any kind of Push Initiator).
- Choose one of the last three (unencoded) types if you are assembling a multipart message that you wish to be readable by MIME tools, displayed in readable format in a text editor, or emailed. Include unencoded parts only.

Refer to the section titled Saving Multipart Messages for additional information regarding how your choice of the `Content-Type` header value influences NMIT's saving operations.

After you have selected a content type for the multipart message from the drop-down list in the **Header Value** column, you may enter other optional headers and values by clicking in the empty row beneath the **Header Name** column. The table cell becomes editable when you click it so you can enter a header name. Click the adjacent empty table cell in the **Header Value** column to enter a value for the header name you entered.

If you place a check mark in the **Send/Save** column adjacent to a particular header, NMIT sends that header, and when you save the file, writes it out as part of the message; otherwise, it does not. This feature eliminates the need to delete and then re-enter values for individual test cases.

## Including Parts

In the Multipart Parts region to the left in the Multipart Message editor, you assemble the multipart entries that are to be included in the message; that is, you insert the desired part files (use CTRL with mouse clicks to select multiple files) and then place them in the desired order.

If you have specified one of the three WAP `Content-Type` headers and expect the SDK to automatically encode the component parts when it pushes the multipart files to the SDKs, you must insert only encoded content parts, whereas if you have specified a MIME `Content-Type` header, you should insert only the source format of the content parts.

You cannot create a part within the Multipart Message editor. You must have already created, debugged, and saved each entry you wish to include. Also, note that nested multipart parts are prohibited by WAP Push specifications; NMIT cannot recursively unpack the parts.

### Multipart Parts Controls

| | |
|---|---|
|  | After you have selected a part in the Parts List display region, click this button to move that part up one position. The topmost entry in the region is the first multipart entry and will be processed first; the bottom entry is processed last. |
|  | After you have selected a part in the Parts List display region, click this button to move that part down one position. The topmost entry in the region is the first multipart entry and will be processed first; the bottom entry is processed last. |

**Multipart Parts Controls**

| | |
|---|---|
|  | Removes the selected part or parts from the Parts List region. (If you mistakenly remove a part, you can easily replace it by choosing **Edit>Undo**.) |
|   | Displays the Open dialog in which you can select one or more files to be included in this multipart message. The selected files are then inserted into the Parts List file display region. |
| Drag and Drop | You can use the Windows drag and drop feature to add files to a multipart message. In Windows Explorer, left click the file, holding down the mouse button; then drag it to the Parts List region and release the mouse button. |
| Double Click | When you double click the icon for an included part file, NMIT opens the part in an appropriate editor if it recognizes the content type. If NMIT does not support the content type, it will use the system's default editor for that content type. |

## Editing Part Headers

The section Multipart Message Processing described how phone SDKs managed Push messages in their cache. Assuming you have included all part files in the multipart message, you must now edit the Part headers in the Part Properties region to ensure the cache behavior will be what you intend.

Each part that you wish to be cached must specify the `X-Wap-Content-URI` header, or alternatively, the `Content-Location` header. The value for this header for a given part must match any references to that part in the content of other messages. So, for example, to enable an SI message (the first part) with an `href` attribute whose value is `http://myhost/page.wml` to fetch this (a succeeding part) from the cache, the `X-Wap-Content-URI` header of the succeeding part needs to match the string `http://myhost/page.wml`. So, you must edit the header fields of each part to coordinate with any references made in other parts.

If neither the `X-Wap-Content-URI` nor `Content-Location` header is present for a given part, the phone SDK will discard the part.

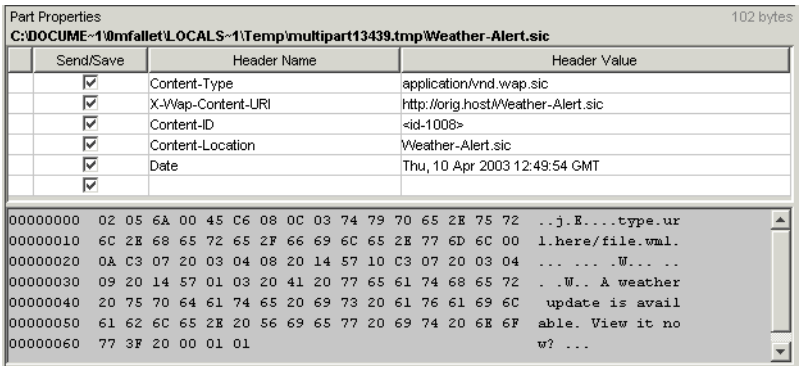
The `Date` header specifies the creation date, which, for multipart messages is the date and time when the part is included in the multipart message. NMIT includes the value of this header automatically.

NMIT's Multipart editor initializes the value of the `Content-Location` header with the file name used when you insert the part into the multipart message. NMIT uses this header to name the part when NMIT opens up an existing multipart file.

## Part Properties With Content

The Multipart Part Properties region to the bottom right of the Multipart Message editor is used to display information about a part which has been selected in the Parts List region.

The following figure shows the Part Properties region in which a `sic` file is displayed.




As described in Editing Multipart Headers, you can modify any header value by clicking on it. You can also delete any header and its corresponding value by selecting the entire row and pressing **Delete**. However, you cannot delete the first (required) header **Content-Type**, though you can change its value.

The content of the multipart entry is displayed in an area just below the **Header Name** and **Header Value** region. In the example shown, the content is encoded. Any non-ASCII content type (such as encoded content) is displayed as a hexadecimal byte stream. The text equivalent of this byte stream is displayed, line by line, to the right.

Note the dotted split-pane divider along the left edge of the **Part Properties** region. If you rest the cursor on this region, the cursor changes, allowing you to resize the **Part Properties** and adjacent **Parts List** regions.

## Multipart Button Controls

The Multipart Message editor contains the following button controls.

**Multipart Message Controls**

| | |
|---|---|
| **Add Part** | Opens a navigation dialog that enables you to locate a part file to be included in the multipart message. |
| **Push** | Pushes the multipart message to the phone SDK. |
| **Clear Contents** | Removes all multipart entries from the current Multipart editing window, thus allowing you to easily start over. |

## Saving Multipart Messages

You can save a multipart message to a file in either an unencoded `.mp` or an encoded `.mpc` format by choosing **File>Save As**.

To save in unencoded (standard MIME) format, you must have selected one of the following MIME header values for the `Content-Type` header in the **Multipart Headers** region:

```
multipart/mixed
multipart/alternative
multipart/related
```

To save in WAP-encoded format, you must have selected one of the following WAP header values for the `Content-Type` header in the **Multipart Headers** region:

```
application/vnd.wap.multipart.mixed
application/vnd.wap.multipart.alternative
application/vnd.wap.multipart.related
```

The saved file includes (1) multipart headers, (2) headers of all parts, and (3) contents of all parts.

If you separately modify a part that is included in an already saved multipart message, the multipart message is not updated with the changed part. To update a multipart message with an updated part, you must reopen the multipart message, delete the part which has been modified, and then reinsert the updated part.

You might wish to open an unencoded MIME multipart file in a common text editor such as Windows Notepad to see how NMIT has constructed the multipart file. Encoded multipart files, on the other hand, are ready to be deployed by Push applications such as a Push Initiator program or by the NMIT Push editor (to a phone SDK).

Because NMIT does not inspect or process the content section of any of the parts within a multipart message, it is possible that a multipart message saved in encoded format may contain unencoded data. This would occur, for example, if you included an unencoded part when you constructed the multipart message. It is your responsibility to ensure that the multipart message parts you include are consistent with the multipart format in which you wish to save the multipart message.

Encoded multipart files are ready to be used by Push applications such as a Push Initiator program or by the NMIT Push application, as this is the format in which the multipart message is pushed over the network.

# Messaging Editors

A Multimedia Messaging Service (MMS) message is a specially formatted message containing rich media content: one or more of text, image, or audio files, with an optional SMIL (Synchronized Multimedia Integration Language) file for specifying how the included media files should be displayed.

NMIT provides the following three messaging editors:

| | |
|---|---|
| MMS Wizard | The MMS wizard presents a series of dialogs in which you enter addressing information and specify media parts for inclusion in an MMS message. It finishes by opening the MMS Message Editor and populating this editor with the information you entered.<br>It is recommended to use the wizard if you are including a SMIL file as one of the media parts, as in this case, the wizard also automatically inserts header values that enable sequential processing of the parts in accordance with the SMIL file.<br>The MMS wizard is opened whenever you choose **File>New>MMS Wizard.** |
| MMS Message Editor | This editor displays all MMS message headers, all media parts and media part headers, and provides functions for adding and deleting parts, modifying headers, creating an encoded MMS message from the parts and headers, pushing the message to a phone SDK, and saving the message to a file.<br>The MMS Message editor is opened whenever you (1) choose **File>New>MMS Message**, (2) choose **Finish** in the MMS Wizard, (3) open an already existing MMS message, or (4) press the **Generate MMS** button in the SMIL editor. |

| | |
|---|---|
| SMIL Editor | Creates a SMIL file that can be included in an MMS message as a media part. The SMIL part identifies the media parts and specifies how these should be presented in the phone SDK.<br>The SMIL editor is opened whenever you (1) choose **File>New>SMIL** or (2) choose **Tools>DTD Manager**, select a SMIL content type, and choose **Open Editor**. |

This chapter contains the following sections:



# Quickly Creating a Multimedia (MMS) Message

The task summary described here gives the steps used to quickly create and display an MMS message.

1 Choose **File>New** in the NMIT main window or in the NMIT editors window.

2 Choose the **Messaging** tab in the Available Contents dialog, select **MMS Wizard**, and choose **OK**.

NMIT opens the MMS Wizard, which guides you through the process of entering essential information such as the recipients, the media file content, and the use (or not) of a SMIL presentation script.

After completing the wizard dialogs, NMIT opens the MMS Message Editor, which displays the MMS data you entered so far. Here you have the opportunity to (1) edit header values, (2) add or delete headers, (3) choose to send or not send certain headers, (4) reorder the sequence of included media files, (5) add or delete files, and (6) edit header values or add or delete headers for included media files.

3 Choose the **Push** button in the MMS Message Editor to display the message on the current phone SDK. The User's Guide for the phone SDK describes how the received MMS message is handled within the phone SDK. You must ensure that the phone SDK you are using is capable of receiving MMS messages, as not all are.

# Overview of MMS Messages

MMS is a WAP-based, store-and-forward, rich-content messaging system. It uses the WAP WSP protocol stack for communication between a client device and a WAP gateway, and it uses a Multimedia Messaging Service Center (MMSC), which is a kind of combination MMS Proxy-Relay and MMS Server, as the store-and-forward center (similar to regular email).

The following steps summarize MMS messaging in the network environment:

1 You create an MMS message on a mobile handset and send it to the addressee. (The value of the `X-Mms-Message-Type` header is `m-send-req`.) The mobile handset initiates a WAP connection and sends the entire message with all part content as a WSP Post.

2 The MMSC receives the MMS message and responds to the originating handset over the same WAP connection "Message sent."

3 The MMSC sends an MMS message to the addressee (receiving handset) containing the URL for retrieving the MMS message. (The value of the `X-Mms-Message-Type` header is `m-notification-ind`.)

4 The receiving handset initiates a WAP connection and retrieves the message using a WSP Get.

5 The MMSC sends the MMS message to the receiving handset over the same WAP connection using a WSP Get Response. (The value of the `X-Mms-Message-Type` header is `m-retrieve-conf`.) The receiving handset displays "Message received."

6 The receiving handset displays "Message received" and sends to the MMSC over the same WAP connection using a WSP Post an MMS message (type `m-acknowledge-ind`) indicating the message has been received.

7 If requested by the sender, the MMSC sends an MMS message to the message sender (originating handset) "Message delivered."

Note that in only steps 1 and 5 is the entire message transmitted. Within NMIT, the MMS message you create will correspond to that sent in either step 1 or 5 depending upon the value you choose for the `X-Mms-Message-Type` header. The MMS message is encoded and sent directly to the phone SDK, through a private interface, not through the WSP protocol stack. It does not pass through a WAP gateway or through an MMSC.

Therefore, to summarize, within NMIT, the sending of an MMS message having the message type `m-send-req` to a phone SDK enables you to see what an MMSC would receive from the sender (originating application), whereas sending an MMS message of type `m-send-conf` enables you to see what a recipient receives from the MMSC.

The following additional topics are discussed in this section:

**Structure of an MMS Message**

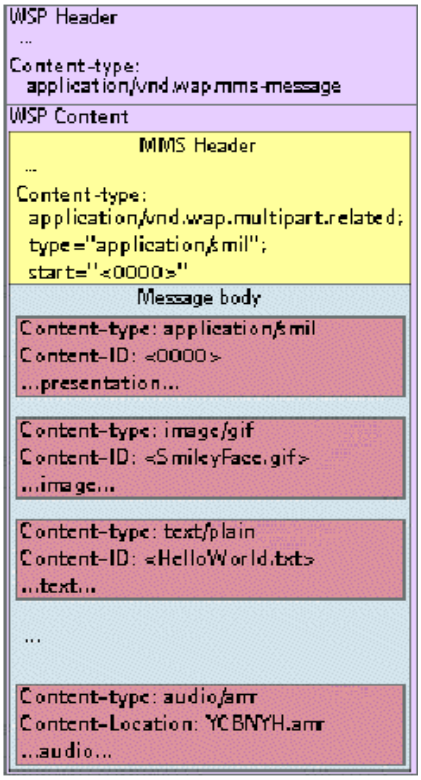
**Relationship Between MMS Media Parts**

**Saving MMS Messages**

## Structure of an MMS Message

The following figure depicts an MMS message as it is constructed to be sent over the network within both NMIT and the network environment.

### MMS Message Structure




At the outermost (WSP) level, an MMS message consists of a WSP Header section and a WSP Content section.

The `Content-Type` header specified within the **WSP Header** section must be `application/vnd.wap.mms-message`.

The **WSP Content** section itself consists of a multipart message in WSP-encoded format consisting of an MMS Header section and an MMS Message Body.

The **MMS Header** section must specify a `Content-Type` header of (1) `application/vnd.wap.multipart.related` if there is a SMIL file included or (2) `application/vnd.wap.multipart.mixed` if there is no included SMIL file.

The `start` parameter specifies a value that is used to determine which part in the Message Body is to be processed first: specifically, that part whose `Content-ID` header specifies that same value. When a SMIL part is included, its `Content-ID` header must specify this value and the `Content-Type` header must include a `start` parameter that matches the `Content-ID`.

The **Message Body** section consists of each included part, where each part consists of a Part Header section and Part Content section.

The **Part Header** section consists of a number of headers particular to the part.

The **Part Content** section contains the actual part content.

When you create an MMS message using the NMIT MMS Message editor, you are creating a message that conforms to the structure shown above. When you push the message to a phone SDK, NMIT inserts the WSP Header value `application/vnd.wap.mms-message` automatically, encodes the message, and pushes it to the phone SDK through a private interface.

## Relationship Between MMS Media Parts

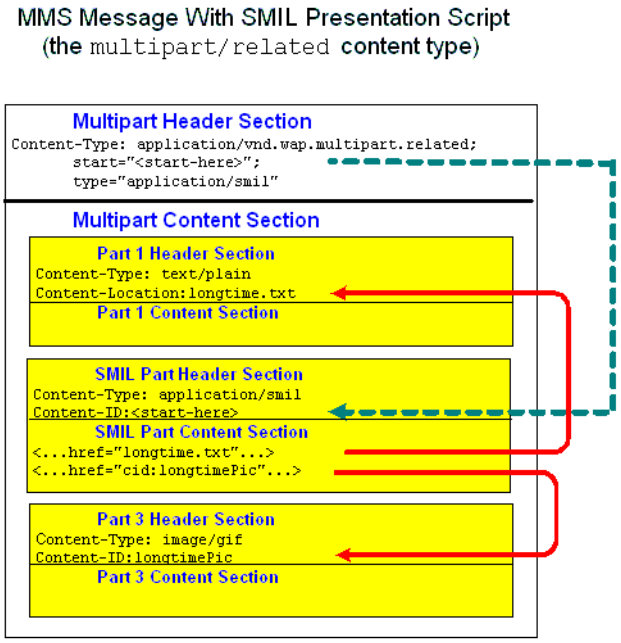
MMS devices expect the content part (the body) of an MMS message to be formatted as either a `multipart/mixed` or as a `multipart/related` WSP-encoded message.

If `multipart/mixed`, MMS devices infer that no presentation (SMIL) file is present as one of the included parts. In this case, the media files are presented to the device display in the order in which they appear in the message.

If `multipart/related`, MMS devices infer that a presentation file is present. In this case, MMS devices expect to find (in the Multipart Header section of the MMS content) a `start` parameter to the `Content-Type` header, which it uses to locate the SMIL file within the content body, as this SMIL part must be processed first.

The following figure shows an MMS message containing a SMIL file. The green dotted line shows how headers are used to identify the SMIL part. and the way in which header values are used by the MMS device to parse the content in order to determine the order in which individual media files are to be processed.

Use of Headers in MMS `multipart/related` Messages




First, the phone SDK locates the part whose `Content-ID` header value matches the value given in the `start` parameter to the multipart/related content type header. This part, the SMIL presentation file, may be considered the "root" of the `multipart/related` structure.

The phone SDK retrieves the required `type` parameter, which specifies the content type of the part identified by the `start` parameter; this allows the phone SDK to determine whether it can handle the message before unpacking it. The value of this `type` parameter should match the value of the `Content-Type` header of the SMIL file part (if it does not, behavior of the phone SDK is unpredictable).

Next, in processing the SMIL part, the device learns the number of included media parts and the order in which they should be processed. It locates these parts by matching string values given in

the SMIL file to the values of headers in the individual parts. Specifically, it seeks to find the part whose `Content-Location` header value (typically, the original filename) equals the value of an `href` attribute in the SMIL file. Or, if the value of an `href` attribute in the SMIL file is prefixed with the `cid:` scheme, it seeks the part whose `Content-ID` header value (typically, some meaningful identifying string) equals the value specified by the characters immediately following the characters `cid:` and enclosed within brackets (< and >). When a match is found, the device then uses that part as the referenced media object. The following shows an example of this mapping:

**SMIL File Referencing of Included Media Parts**

| Attribute Values in SMIL File | matches | Header Values in Part Files |
|---|---|---|
| `href="longtime.txt"` | ⟶ | `Content-Location: longtime.txt` |
| `href="cid:longtimePic"` | ⟶ | `Content-ID: <longtimePic>` |

## Saving MMS Messages

When you save an MMS message, NMIT writes the entire package to a file; the package includes (1) MMS message headers, (2) headers of all parts, and (3) contents of all parts.

Note that if you separately modify a part that is included in an already saved MMS message, the MMS message is not updated with the changed part. To update an MMS message with an updated part, you must reopen the MMS message, delete the part which has been modified, and then reinsert it.

When you reopen an MMS message, the MMS Wizard is not invoked. Instead, the message is opened in the MMS Messaging editor within which you can view the headers and content of each part, as well as rearrange the sequence of the parts.

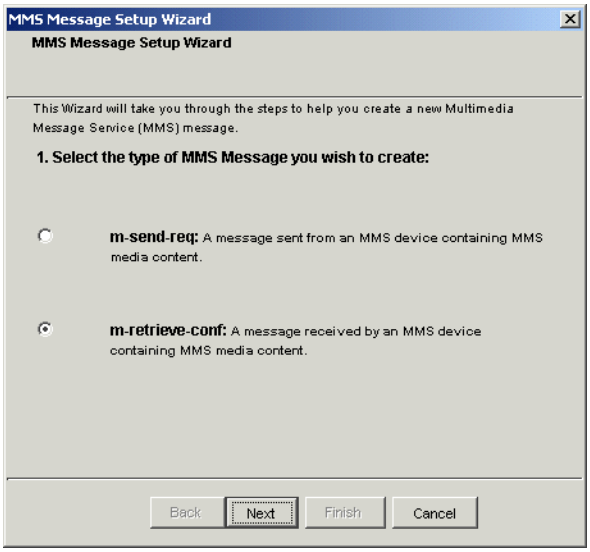
# MMS Wizard

Use the MMS Message Setup Wizard to begin creating an MMS message. The MMS Message Setup Wizard prompts you for required information such as headers and parts to be included in the message, autogenerates a SMIL file if you wish to include one, and then assembles these components and presents them to you in the MMS Message Multipart editor. From within the multipart editor, you will then have the opportunity to add, delete, or rearrange parts, modify header values, and so on, prior to the generation of the final encoded MMS message.

It is recommended to use the MMS Wizard if you are creating a new MMS message and are including a SMIL file. This is because header references must be used to coordinate the presentation of parts, as shown in the table titled SMIL File Referencing of Included Media Parts.

The following are the MMS Wizard dialogs described in sequential order:

### 1 Select MMS Message Type



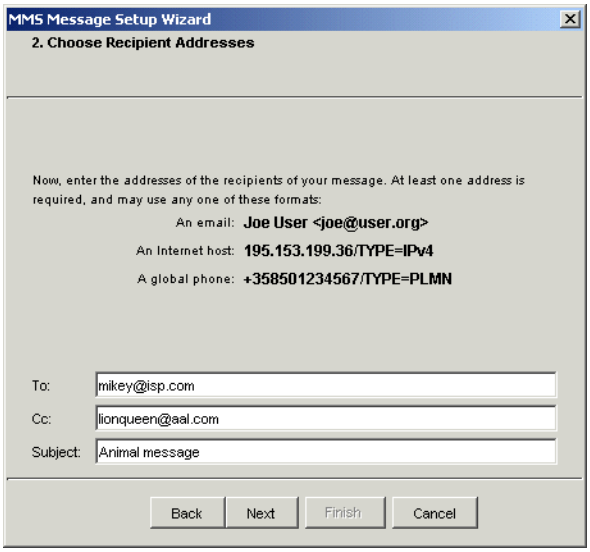

This dialog requests that you choose the message type. The selection you make results in a header value being inserted for the `X-Mms-Message-Type` header.

- Choose the `m-send-req` radio button to create an MMS message that replicates an MMS message as it would be received by an MMSC from an originating application (sender) in the network environment.
- Choose the `m-retrieve-conf` radio button to create an MMS message that replicates an MMS message as it would be received by a terminating application (receiver) from an MMSC in the network environment.

See the section titled Overview of MMS Messages for a more detailed description of how these differing message types are deployed in the MMS messaging environment.

Note that some headers differ for these two message types. Headers for the message type you choose will be displayed within the MMS Message editor after you choose **Finish** in the MMS Wizard.

### 2 Choose Recipient Addresses



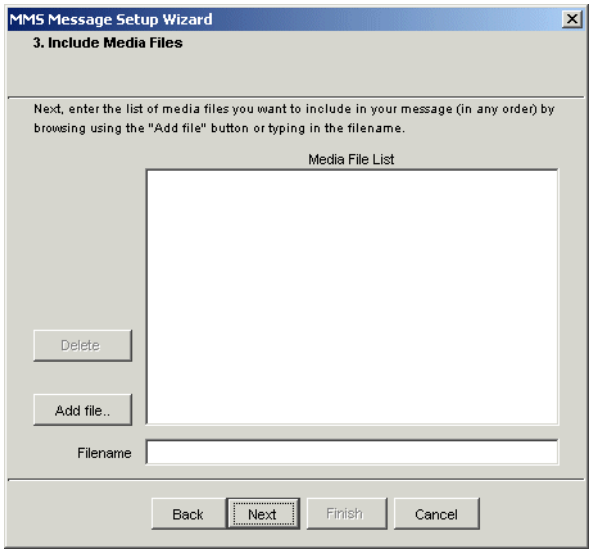

This dialog requests that you enter information about the recipients for the MMS message. You must specify a value for at least one of the following fields: **To**, **Cc**, or **Bcc**.

#### MMS Message Addressing

| | |
|---|---|
| **From** | The **From** header is required, but NMIT automatically inserts the value `mms-editor@toolkit`. Later, you can change this value within the MMS Message Editor. |
| **To** | Address of the MMS recipient terminal. The address format may be either an email or device-address. These are precisely defined in the WAP MMS Encapsulation specification.<br>Specify an email address with the "<"and ">" delimiters; for example: `<MyFriend@isp.com>`.<br>Specify a device-address as a global phone number or internet host followed by `/Type` where the `Type` specifies the type of the device. For example, to address an internet server, specify the IP address followed by `Type=IPV4`; if a mobile phone, the phone number followed by `Type=PLMN`. |
| **Cc** | Address of individuals or devices to which a carbon copy of the message is to be sent. The address formats are as described as in the **To** field. |
| **Bcc** | Address of individuals or devices to which a blind carbon copy of the message is to be sent. (These addresses are not included in the messages sent to addresses specified in the **To** or **Cc** fields. The address formats are as described as in the **To** field. |
| **Subject** | (**Optional**). Subject of the MMS message. |

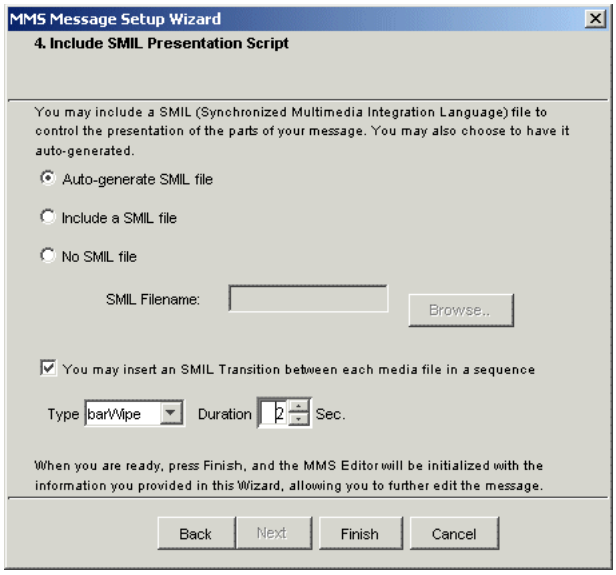
### 3 Include Media Files

The wizard dialog requests that you specify the location of the media files that you wish to include in the MMS message.




| | |
|---|---|
| **Add file...** | Opens the **Select media file** dialog from which you can select the media files to be included. Use the **CTRL** and the **Shift** keys when using the mouse in order to select multiple files. Then choose **Open** to add these files to the media file list.<br><br>Note that media files can be added in any order, as you will have an opportunity later to rearrange them; however, the files must exist in order to be added to the list. |
| **Filename** | Full path to the media file to be added to the MMS message. You can enter the path directly into the text box or press the **Add file...** button to navigate to it. |
| **Delete** | Deletes the file or files currently selected in the Media File List. Files are deleted from the list only, not from disk. |

### 4 Include SMIL Presentation Script




This dialog requests that you specify whether you wish to include a SMIL presentation script with the MMS message. If you do, you must either specify the location of an existing SMIL file or specify that you wish to have one generated for you.

| | |
|---|---|
| **Auto-generate SMIL file** | Choose this button if you wish NMIT to generate a SMIL file for this MMS message.<br><br>In generating the SMIL file, NMIT uses default values for the presentation layout, that is, the areas of the display to be occupied by text and image media files.<br><br>NMIT also adds the file locations of all media files you specified for the MMS message, and, based on the file types of these files, identifies the files as either "text" or "image" and uses these values for the `region` attribute.<br><br>NMIT generates a SMIL file capable of presenting the media files to the display in a default configuration. You can modify the SMIL file later using the SMIL editor.<br><br>After you choose **Finish**, NMIT saves the SMIL file using the file name `default.smil` and inserts `default.smil` into the new MMS message as the first part. |
| **Include a SMIL file** | Choose this button if you wish NMIT to include an existing SMIL file for this MMS message. (See the section titled SMIL Editor for information about creating your own SMIL file.)<br><br>After choosing this button, the **SMIL filename** text entry box becomes active. Enter the path to the file directly into the text box or choose the **Browse** button to open the **Select media file** dialog within which you can navigate the local disk directory structure to find the file to be included.<br><br>Finally, choose **Finish** to include the SMIL file. |

| | |
|---|---|
| **No SMIL file** | Choose this button if you do not wish to add a SMIL file to the MMS message. |
| **SMIL Transition** | Lets you insert a transition mode and length of time between each media file in a sequence. **Type** controls the style of the transition and **Sec** controls the length of time in seconds. |

After you have made your selection, choose the **Finish** button. The MMS Wizard closes and the MMS Message Editor opens, displaying the information you entered.

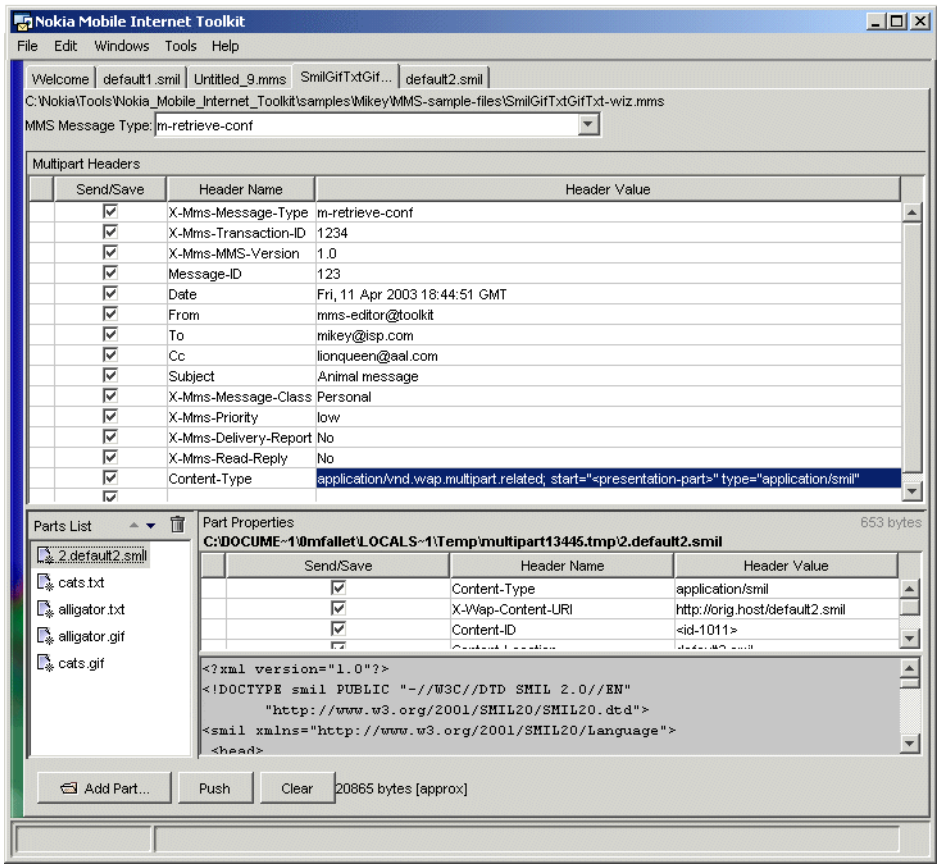
# MMS Message Editor

You use NMIT's MMS Message Multipart editor to assemble existing files (parts) into a single multipart message, to specify headers for the individual parts and for the entire MMS message, and then finally to push the MMS message to one or more phone SDKs.

The MMS Message Editor is displayed when you perform any of the following actions:

- Choose **File>New** and then choose **MMS Message** on the **Messaging** tab.
- You complete the dialogs of the MMS Message Setup Wizard, choosing **Finish** in the last dialog.
- You open an existing MMS message.

The following figure shows the MMS Message editor:




Note the following with respect to the contents shown above:

The **MMS Message Type** drop-down list at the top shows the type to be `m-retrieve-conf`. You can change the type to `m-send-req` and then back at any time and header values will be retained.

The MMS Message Headers section is expanded to show all headers available for the content type. These vary slightly for `m-retrieve-conf` and `m-send-req`.

The Parts List region at the lower left shows five files. The first file in the list is a SMIL file; because it is selected, its headers and content are displayed in the adjacent Part Properties region.

Button Controls at the bottom of the window enable operations on the current message, specifically, pushing the message to phone SDKS and adding and clearing contents.

## MMS Message Headers

MMS Message headers are distinct and unrelated to any of the constituent parts within the multipart message. Multipart headers serve a function much like the postal instructions on a mail envelope, indicating the nature of the items within the container.

The WAP MMS specifications define precisely the multipart header names and the sequence in which some of them must appear. Therefore, within NMIT's MMS Message Editor, you cannot delete these specified headers or reorder them, though you can choose that optional headers not be sent or saved to a file (using the **Send/Save** check box). Any additional headers you wish to add can be added to the end of the displayed MMS Message headers.

The Multipart Headers region at the top of the MMS Message editor window contains a four-column table in which you perform operations on headers.

**To enter a value for a header name:**

- Click in the adjacent **Header Value** field and type the desired value. Values for some MMS headers are fixed and cannot be changed. Values for others are constrained within a list of values.

**To add a new line on which you can enter an additional header name and value:**

- Click anywhere on the last line and enter text until a new line is displayed.

**To delete a header line:**

- Click in the left column of the line and press the **Delete** key. Note that you cannot delete standard MMS headers, only additional headers you have added.

**To specify that NMIT send the header when pushing the MMS message to a phone SDK or saving the message to a file:**

- Click in the **Send/Save** column to place a check mark on the header line.

**To specify that NMIT not send the header and not save it to a file:**

- Click in the **Send/Save** column to remove the check mark on the header line. This feature eliminates the need to delete and then re-enter values for individual test cases. Note that you cannot remove the check mark adjacent to a required header, as required headers must always be sent.

The following table describes each header and its possible values. In this region, there are two columns titled **Header Name** and **Header Value**.

### MMS Message Headers

| Header Name | Header Value |
|---|---|
| `X-Mms-Message-Type` | (Required) The value is either `m-retrieve-conf` or `m-send-req`, selectable from the drop-down list.<br>The section titled Overview of MMS Messages describes the difference between these two message types. |
| `X-Mms-Transaction-ID` | (Optional) Identifies either (1) the transaction started by the MMSC in sending an `M-Notification` to the MMS recipient terminal and without that terminal sending an `M-NotifyResp` or (2) a new transaction where deferred delivery was requested by the terminal. |
| `X-Mms-MMS-Version` | (Required) The MMS version number 1.0 is fixed and cannot be changed. |
| `Message-ID` | (Optional) Unique ID assigned to the message. It is required whenever the MMS originating terminal requests a read reply. |

### MMS Message Headers

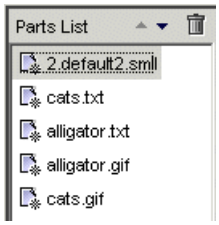
| Header Name | Header Value |
|---|---|
| `Date` | (Required) Date and time that the message is sent (from the MMSC to the MMS recipient terminal). NMIT's MMS Message Setup Wizard creates this date, but you can change it. |
| `From` | (Optional) See the table titled MMS Message Addressing. |
| `To` | (Optional) See the table titled MMS Message Addressing. |
| `Cc` | (Optional) See the table titled MMS Message Addressing. |
| `Bcc` | (Optional) See the table titled MMS Message Addressing. |
| `Subject` | (Optional) See the table titled MMS Message Addressing. |
| `X-Mms-Message-Class` | (Optional) One of four message classes: `Personal`, `Advertisement`, `Informational`, `Auto`. If not specified, the default is `Personal`. |
| `X-Mms-Priority` | (Optional) One of three priorities: `Low`, `Normal`, `High`. If not specified, the default is `Normal`. |
| `X-Mms-Delivery-Report` | (Optional) Specifies whether the sender wishes to receive a delivery report from each recipient. Values are `Yes` or `No` (the default). This is not supported by NMIT. |
| `X-Mms-Read-Reply` | (Optional) Specifies whether the sender wishes to receive a read report from each recipient. Values are `Yes` or `No` (the default). This is not supported by NMIT. |
| `Content-Type` | (Required) Enter the value `application/vnd.wap.multipart.mixed` if the MMS message contains no SMIL file.<br>If the MMS message does contain a SMIL file, enter the value `application/vnd.wap.multipart.related`, followed by the `start` and `type` parameters. See the section titled Relationship Between MMS Media Parts for a description of these parameters.<br>See the table titled Editing Multipart Headers for more information regarding the differences between the `multipart/mixed` and `multipart/related` content types. |

## Parts List

In the Parts List region to the left in the MMS Message editor, you assemble the parts that are to be included in the message; that is, you insert each part (file) and then place the files in the desired order.

You cannot create a part within the MMS Message editor. You must have already created, debugged, and saved to persistent each part you wish to include. Also, note that nested parts are prohibited.

The following figure shows the Parts List region containing five parts.



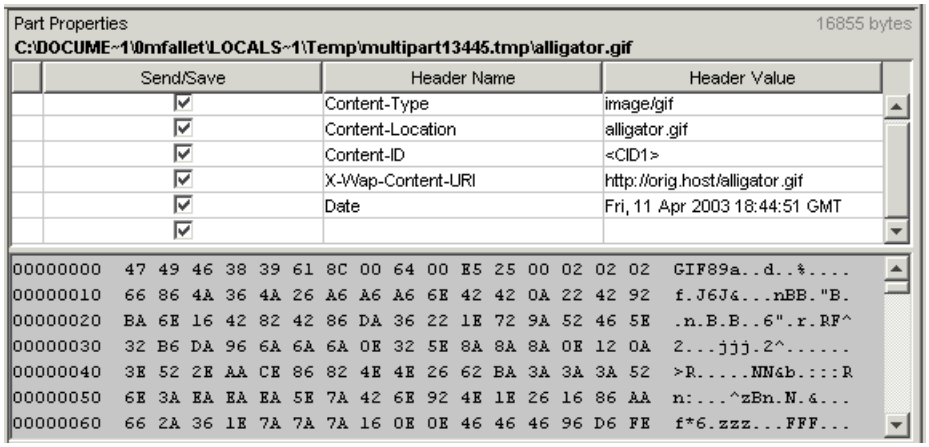

**Parts List Region Controls**

| | |
|---|---|
|  | After you have selected a part in the Parts List display region, click this button to move that part up one position. The topmost entry in the region is the first multipart entry and will be processed first; the bottom entry is processed last. |
|  | After you have selected a part in the Parts List display region, click this button to move that part down one position. The topmost entry in the region is the first multipart entry and will be processed first; the bottom entry is processed last. |
|  | Removes a selected part from the Parts List region. Note that if you mistakenly remove a part, you can replace it by choosing **Edit>Undo**. |
|  | Displays the Open dialog in which you can select a file to be included in the multipart message. |
| Drag and Drop | You can use the Windows drag and drop feature to add files to the Parts List. In Windows Explorer, left click the part file, holding down the mouse button; then drag it to the Parts List region and release the mouse button. |
| Double Click | When you double click the icon for an included part file, NMIT opens the file in an appropriate editor if it recognizes the content type. If NMIT does not support the content type, it attempts to open the file in the system's default editor for that content type. |

## Part Properties

The Part Properties region of the MMS Message editor is used to display information about a part which has been selected in the Parts List region. It provides an opportunity for you to modify the values of the default headers, as well as to add additional headers (and values).

The following figure shows the Part Properties region in which a `gif` image part is displayed.




As described in the section MMS Message Headers, you can modify any header value by clicking on it. You can also delete any header and its corresponding value by selecting the entire row and pressing **Delete**.

The empty row below the displayed headers and values can be used to add a new header and its value: click in the row to enter the desired content.

The content of the part is displayed below the header section.

Note the dotted split-pane divider along the left edge of the **Part Properties** region. If you rest the cursor on this region, the cursor changes, allowing you to resize the **Part Properties** and adjacent **Parts List** regions.

## Button Controls

The MMS Message editor contains the following button controls.

### MMS Message Controls

| | |
|---|---|
| Push | Pushes the MMS message currently displayed to the phone SDK. |
| Clear | Removes parts from the current editing window, thus allowing you to easily start over. |

# SMIL Editor

The Synchronized Multimedia Integration Language (SMIL) is an XML-based language that Lets you write interactive multimedia presentations. SMIL makes it possible to control both the timing and layout of these objects as they are displayed on a screen.

NMIT's SMIL editor supports two version of the SMIL spec:

- SMIL 2.0, authored by the W3C standards organization. The editor supports all 2.0 language features (element and attributes) and validates SMIL documents against the SMIL 2.0 DTD.

- SMIL 3GPP, which consists of the modules required by SMIL Basic Profile (and SMIL 2.0 Host Language Conformance) and additional MediaAccessibility, MediaDescription, MediaClipping, MetaInformation, PrefetchControl, EventTiming and BasicTransitions modules. The editor validates SMIL3GPP against the SMIL 3GPP DTD.

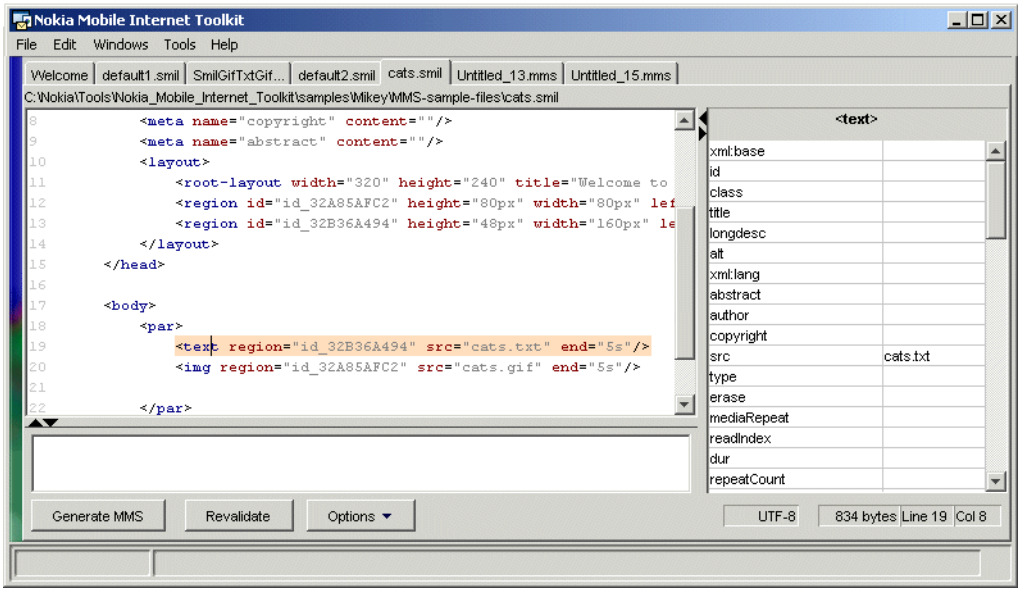
You use NMIT's SMIL editor to specify layout, presentation, and timing properties to be used by the phone SDK when displaying the parts of an MMS message. Specifically, you can specify:

- Layout properties for the size of the screen upon which the MMS part contents will be displayed.
- Screen regions, identified by ID values, wherein MMS parts having that ID will be displayed.
- URIs where the media parts to be displayed are located.
- Timing values (in seconds) that indicate how long each part is to be displayed.
- The transition type and length of transition between parts.

Finally, having specified the information listed above, you can generate an MMS message based on the SMIL file.

Some Nokia phone SDKs currently ignore SMIL part within an MMS message. Check the documentation for information about support for SMIL parts.

The following figure shows the SMIL Message editor:




SMIL features currently supported in existing devices are far fewer than those defined in the specification. For this reason, several corporations, including Nokia, have agreed to support a limited subset of SMIL elements in order to achieve the minimal presentation capabilities required by the first phase in MMS message implementation. An *MMS Conformance Document*, which describes the minimum requirements needed for end-to-end interoperability of MMS handsets and MMS servers, is available at http://openmobilealliance.org/.

Since SMIL is an XML-based content type, NMIT's SMIL editor has all of the features of a structured editor as described in the section titled Editing Features For Validatable Text Content. Refer to this section for a detailed description of the editor features.

The SMIL editor is unique in that it provides the following additional active button:

| | |
|---|---|
| Generate MMS | Opens the MMS Message editor and within this editor (1) includes the current SMIL document and all part files referenced by the SMIL file and (2) inserts header values related to the included parts. By default, the MMS message type header is set to `m-retrieve-conf`, though you can change this (in the MMS Message editor) to `m-send-req` if desired.<br>The SMIL document must be valid in order to generate an MMS message, and the part files it references must be in the same directory. |

## Generating an MMS Message From a SMIL File

To generate an MMS message starting only from a SMIL document.

1 Create a new SMIL document or open an existing SMIL document containing all standard, required elements; add the desired information, including that for all parts; then choose **Revalidate** to validate the SMIL document and fix any errors. Finally, save the SMIL document using the desired file name in the same folder as the message parts.

2 If you are testing the message on the Series 60 Content Authoring SDK for Symbian 1.0x OS, Nokia Edition: In the SMIL document, delete the following lines at the top of the document and save the document using the same file name:

```
<?xml version="1.0"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
       "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
```

This deletion is required in the Series 60 Content Authoring SDK for Symbian 1.0x OS, Nokia Edition, and is not required for later phone SDKs that support SMIL. The Series 60 Content Authoring SDK for Symbian 1.0x OS, Nokia Edition, cannot process a standard SMIL document, which begins with the `<xml>` element, but instead requires that the SMIL document begin with the `<smil>` element.

3 Choose **Generate MMS** to generate the MMS message. The message is opened in the MMS Message Editor.

4 In the MMS Message editor, edit the Part headers if desired. Then choose **Push** to display it on the phone SDK. The parts should be displayed in accordance with the order in which they were specified in the SMIL document.

## Generating an MMS Message From Media Parts

This section describes how to generate an MMS message when you have the media parts to be included.

1 Launch the MMS Wizard and include all your media parts. See the section titled MMS Wizard for instructions.

2 Choose either the **No SMIL file** radio button or the **Auto-Generate SMIL file** radio button in the dialog, titled **Include SMIL Presentation Script**, of the MMS Wizard and then press **Finish**. The MMS Message editor is automatically opened. See the section titled MMS Message Editor for details regarding how to use this editor.

3 If there is no SMIL file, you may wish to order the parts within the Parts list, from top to bottom, in the order in which you wish these parts to be presented in the phone SDK, from first to last. If there is a SMIL part, the SMIL part dictates the order.

4 Modify any MMS or Part headers, as desired.

5 In the MMS Message editor, choose **Push**. The MMS message is displayed in the selected phone SDKs.

## Modifying an Existing MMS Message

If you have an existing MMS message, you may wish to open it in the MMS Message editor in order to display it on a phone SDK or to modify it and then display it. To display your MMS message:.

1 Choose **File>Open** to open the MMS message. The message is displayed in the MMS Message editor.

2 In the MMS Message editor, choose **Push**. If there are no errors, the message is displayed in the currently selected phone SDKs. If there are errors, you must correct these.

The following sections describe modifying parts.

### Modifying Non-SMIL Parts in an Existing MMS Message

This section describes modifying parts within an existing MMS message which does not include a SMIL part. If the MMS message does include a SMIL part, see the section titled Modifying a SMIL Part in an Existing MMS Message.

Modifying a part in an MMS message may mean one of the following:

- **Deleting a part**. To delete a part, select the part and choose the trash can icon at the top of the Parts list. You can then choose the Push button to display the MMS message on selected phone SDKs.
- **Adding a part.** To add a part, choose the **Add Part** button. A dialog is displayed in which you can locate the part to be included. After choosing **Open**, the part is displayed in the Parts list. Ensure that it is positioned in the list in the order in which you wish it to be displayed, as when no SMIL file is used, parts are displayed sequentially from top to bottom as they appear in the list.

- **Editing a part**. To edit an existing part, in the MMS Message editor, double-click the part in the Parts list. The part is opened in its default editor (based on file type). Edit the part as desired and then save it (**File>Save**). The MMS Message editor detects the part has changed and uses the new part. (Click on another part to verify the part has been refreshed.)

  If the part is not refreshed, delete the part and then add the one you just saved. In this case, custom part headers for the deleted part are removed and will need to be added again.

## Modifying a SMIL Part in an Existing MMS Message

To modify a SMIL part within an existing MMS message: In the MMS Message editor, double-click the SMIL part in the Parts list. The part is opened in the NMIT SMIL editor. Edit the SMIL part as desired and then save it (**File>Save**).
The MMS Message editor detects the part has changed and uses the new part. (Click on another part to verify the part has been refreshed.)

If you do choose the **Generate MMS** button in the SMIL editor, a new message having a different file name is opened in the MMS Message editor. This message is different from the one you started from. So, if you have edited any MMS or part headers in the original MMS message, you will generally not wish to create a new MMS message because this header information will not be present in the new message. You could, however, copy the header information from the old message to the new one using Cut and Paste, field by field.

See the section titled Relationship Between MMS Media Parts for a description of the use of headers within an MMS message that includes a SMIL part. These headers will need to be reinserted in order for the MMS message to be properly displayed.

# DRM Editors

Digital Rights Management (DRM) is a system of content types and cell phone behavior that allows a content provider to control how a phone user uses and re-distributes media content (such as a song or a game). This control is achieved by pairing the media content with rights that control how often and for how long a phone user can use the content on the cell phone. For example, a content provider can permit an phone user to receive and play a song once as a preview. Or the content provider can allow the phone user to purchase the rights to play the song indefinitely.

DRM provides three message types that protect content with varying levels of security:

| DRM Message Type | Description |
|---|---|
| Forward lock | A single message that lets a phone user use the content without limits on that device (cell phone), but prohibits the phone user from forwarding the content to another device. |
| Combined Delivery | A single message that lets a phone user use the content with some limitations on that device, but prohibits the phone user from forwarding the content to another device. |
| Separate Delivery | A pair of messages (one with content, one with the rights to the content) that let a phone user use the content with some limitations on that device if the user has received the rights to use the content. The phone user can forward the contents to another phone user but not the rights. Information about obtaining rights is embedded in the content file so a new receiver of the content can obtain the rights to use the content. |

The DRM Message Editor lets you:

- Select and prepare the media content to wrap in a digital rights message. The DRM message works with a copy of the original content so that the original content file is never altered.

- Create and prepare the rights that allows the phone user to use content.
- Push the message on an SDK that is capable of implementing DRM messages.

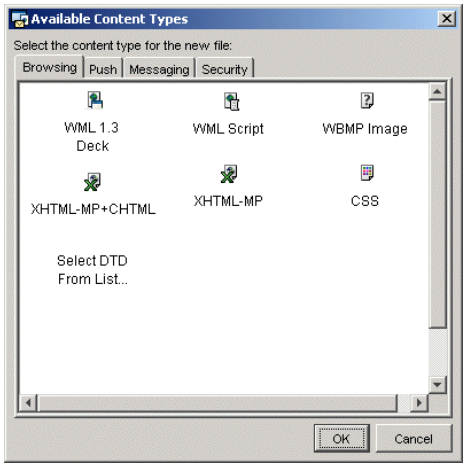
How the content provider chooses to deliver the content and its rights depends on several factors, including whether the content has more than one set of rights available and how the content will be distributed. No content can be used without its associated rights.

## Accessing the DRM Editors

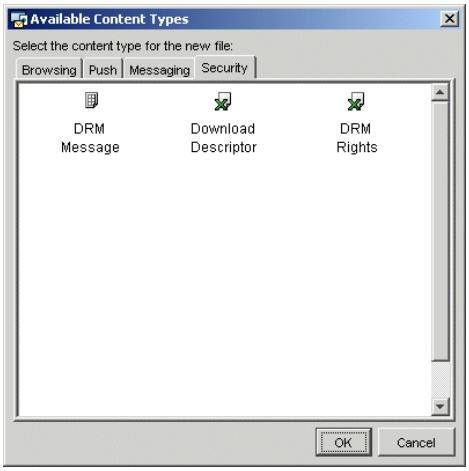
To access the DRM editors:

1 Open the SDK on which you want to display the DRM message.

2 Select **File>New**. The **Available Content Type** window appears:




3 Click the **Deployment** tab to access the DRM editors:



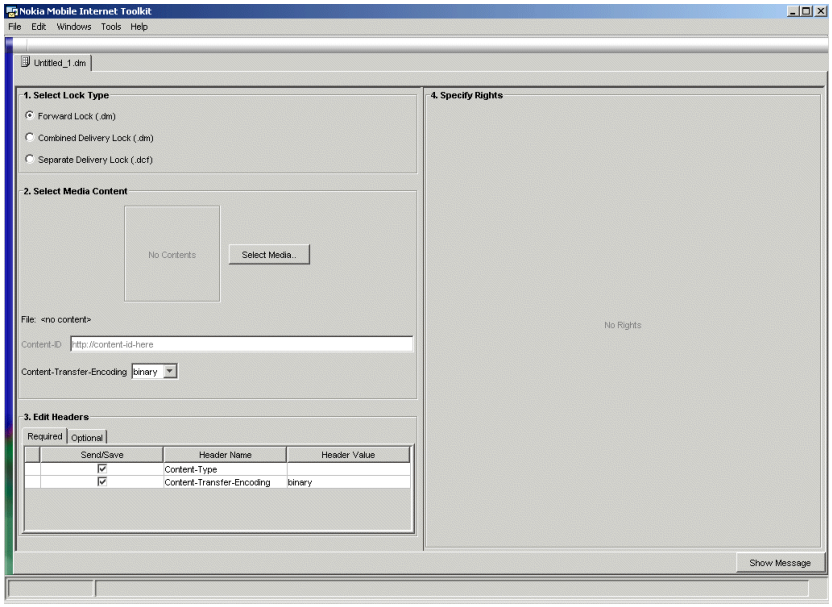
From this window, you can select the DRM component you want to work with:

| To | Click | And see |
|---|---|---|
| Work with content and rights together | **DRM Message** | Creating a DRM Message |
| Work with rights in an XML editor | **DRM Rights** | Working With Digital Rights in the Digital Rights Editor |

# Creating a DRM Message

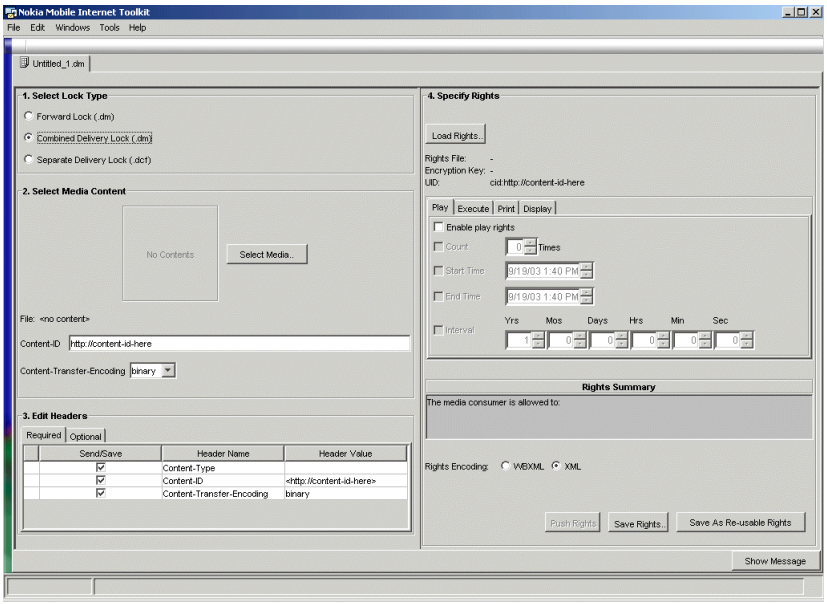
To create a Forward Lock, Combined Delivery, or Separate Delivery message:

1 From the **Deployment** panel, click the **DRM Message** icon, and click **OK**. The DRM Message Editor window appears:




The content of the DRM window varies, depending on the type of message you select. The image above shows DRM window when **Forward Lock** is selected.

When you select **Combined Delivery** or **Separate Delivery**, the digital rights side of the window appears:




This side lets you specify the content (for all message types)

This side lets you specify the rights to the content (Combined Delivery or Separate Delivery only)

Drag the bar between the content and rights sides to adjust the area of the content and digital rights.

From this window, you can create a DRM message in four basic steps. The complexity of each step depends on what type of DRM message you are creating. Some fields that apply to one type of message do not apply to others. The basic steps are:

1 Select the type of message. (Forward Lock, Combined Delivery, Separate Delivery)

2 Select the content. (Browse to the location of the file that contains the content to which you are going to assign rights.)

3 Confirm the required headers and edit the optional headers, if necessary.

4 Specify the rights. (Not applicable to Forward Lock messages).

After you create the message, you can display the message on an SDK to test how the message works.

## Creating a Forward Lock Message

A Forward Lock message contains content that:

- Can be used without limitations on the device that receives it
- Cannot be forwarded to any other device

When you prepare a Forward Lock message, you select the content and save the message in a file that has a `.dm` extension. A `.dm` file uses a standard multipart format with the media content as its only part. The content of a Forward Lock message is always unencrypted.

To prepare a Forward Lock message:

1 Under **Select Lock Type**, select **Forward Lock**.

The **Specify Rights** section on the right of the DRM Message window remains grayed out because a Forward Lock message does not have a rights object. A Forward Lock message has implied unlimited rights associated with its content.

2 Under **Select Media Content**, click **Load Content** to browse to the content file you want to wrap in the message. After you select the file:

- A thumbnail representing the content appears.
- The **Content-Type** header under **Edit Headers** reflects the contents type of the file.

**Content-ID** is not applicable to this type of message.

3 Under **Content-Transfer-Encoding**, select one of the following:

- **Binary** - Leaves the content unencoded. Recommended for most content.
- **Base64** - Encodes content by converting it from binary to 7-bit (ASCII) using a Base64 encoding scheme. (Recommended for use with mail programs that do not handle binary message content.)

4 Under **Edit Headers**, check that the headers listed under the **Required** tab are correct:

- The **Content -Type** value reflects the content type of the file. If this entry is incorrect, you can edit the header. The entry might be incorrect if the original content file did not have the correct suffix for its type.
- The **Content-Transfer-Encoding** value matches the value you selected in step 3.

No headers are listed under **Optional**.

5 Do one of the following:

- Click **Show Message** to display the message on the SDK you've chosen.
- Select **File>Save** or **File>Save As to** save the message as a `.dm` file.
- Select **File>Save As Template** to save the message lock type and its associated information as the initial content for this editor. For more information, see Saving a Template for the DRM Editor.

## Creating a Combined Delivery Message

A Combined Delivery message delivers in a single file:

- Content
- Rights to use the content on the receiving device

For example, an phone user who receives a song in a Combined Delivery message can play the song in accordance to the rights that are in the file - say, five times over one month. The phone user

cannot forward the message to another phone user because a Combined Delivery message contains rights and an phone user cannot forward rights.

When you prepare a Combined Delivery message, you select the content and set the rights to the content. You can then push the file to the SDKs that are currently active in NMIT or save the message in a file that has a `.dm` extension. A `.dm` file uses a standard multipart format. The content is always unencrypted.

Generally, you use a Combined Delivery message when you have a content that you always ship with a specific set of rights, and when you do not require the content to be encrypted for security.

To create a Combined Delivery message:

1 Under **Select Message Type**, select **Combined Delivery**.

The **Specify Rights** section of the DRM Message window becomes active. You will later specify the rights to the content in this section of the window.

2 Under **Select Media Content**, click **Load Content** to browse to the content's file you want to wrap in the message. A thumbnail representing the content appears after you select the file.

3 Under **Content-ID**, enter an identifier for the content. This identifier must be unique to the media object because it links the object to the rights.

The **Content-ID** you enter is reflected identically in these places:

- Under **Edit Headers> Required** tab**>Content-ID** header.
- Under **UID** in the **Specify rights** section of the DRM window.

4 Under **Content-Transfer-Encoding**, select one of the following:

- **Binary** - Leaves the content unencoded. Recommended for most content.
- **Base64** - Encodes content by converting it from binary to 7-bit (ASCII) using a Base64 encoding scheme. (Recommended for use with mail programs that do not have binary conversion schemes.)

5 Under **Edit Headers**, check that the headers listed under the **Required** tab:

- The **Content -Type** value reflects the content type of the file. If this entry is incorrect, you can edit the header. The entry might be incorrect if the content file does not have a suffix.
- The **Content-ID** value matches the value you entered under **Content-ID**.
- The **Content-Transfer-Encoding** value matches the value you selected for Content-**Transfer_Encoding**.

No headers are listed under **Optional**. Combined Delivery messages do not have optional headers.

6 Under **Specify Rights**, enter the rights you want to associate with the content. See Working With Digital Rights.

7 Do one of the following:

- Click **Show Message** to display the message on the SDK you've chosen.
- Select **File>Save** or **File>Save As to** save the message as a `.dm` file.
- Select **File>Save As Template** to save the message lock type and its associated information. For more information, see Saving a Template for the DRM Editor.

## Creating a Separate Delivery Message

A Separate Delivery message is composed of two discrete files that work together to form one DRM message:

- One file contains the encrypted content.
- One file contains a key to unencrypt the content and the rights to use it.

The phone user can use the content on that device only when he or she acquires the rights to that content.

The separation of content and rights permits the phone user to forward the content to another phone user who can then acquire (purchase) the rights from the content provider to use the content. The phone user cannot forward the rights. Each subsequent receiver of the content must acquire new rights from the content provider to use the content.

When you prepare a Separate Delivery message, you select the content and set the rights to the content, as you would in a Combined Delivery message. Then, when you save the message:

- The content is encrypted and saved in a file with a suffix of `.dcf`.
- The rights to the content and the encryption key are automatically saved in another file that has the same prefix as the content file but with an extension of `.dr`.

For example, if you save the content `nifty_tune` in a Separate Delivery message, the two files that make up the message are `nifty_tune.dcf` (content) and `nifty_tune.dr` (rights).

Naming the files with the same prefix is one of the ways the DRM Message Editor matches rights with content. For more information, see How Contents and Rights Are Matched.

The Separate Delivery message has only one file type for the content (`.dcf`). But you can save rights in a Separate Delivery message rights in two formats:

- Unencoded (`.dr`) - Unencoded `.dr` is the default format you get whenever you save a message. You can open, read, and edit a `.dr` file in a DRM Rights editor. Use this format if you plan to re-use the rights with multiple contents.
- Encoded (`.drc`) - This format encodes the rights with WBWML, a binary format that NMIT cannot open, read, or edit. Use this format when you know you will be dealing with a gateway that uses a binary format. For more information about `.drc` files, see Working With Digital Rights.

If you plan to have more than one set of rights for one content, see Creating Multiple Rights Files for a Single Content File.

To create a Separate Delivery message:

1 Under **Select Message Type**, select **Separate Delivery**.

The **Specify Rights** section on the right of the DRM Message window becomes active. You will later specify the rights to the content in this section of the window.

2 Under **Select Media Content**, click **Load Content** and browse to the file that contains the content you want. A thumbnail representing the content appears.

3 Under **Content-ID**, enter an identifier for the content. This identifier must be unique to the content because it links the object to the rights.

The **Content-ID** you enter is reflected identically in these places:

- In the **Content-ID** field where you enter the unique identifier.
- Under **UID** in the Specify rights section of the DRM window.

**Content-Transfer-Encoding** is not used in Separate Delivery messages.

4 Under **Edit Headers**, check the headers listed under the **Required** tab, to ensure that the **Content -Type** value reflects the content type of the file. If this entry is incorrect, you can edit the header. The entry might be incorrect if the content file does not have a suffix.

Separate Delivery messages stores the information under **Content-ID** as part of its format but not as a header.

5 Under **Edit Headers**, edit the headers listed under the **Optional** tab. Only Separate Delivery messages have **Optional** headers that you can edit.

6 Under **Specify Rights**, enter the rights you want to associate with the content. See Working With Digital Rights.

7 Do one of the following:

- Click **Show Message** to display the content on the SDK you've chosen.
- Select **File>Save** or **File>Save As to** save the content in a `.dcf` file and the rights in a `.dm` file. Both files will have the same prefix.
- Select **File>Save As Template** to save the message lock type and its associated information. For more information, see Saving a Template for the DRM Editor.

# Working With Digital Rights

You specify digital rights when you create a Combined Delivery or Separate Delivery message. A Forward Lock message has implied unlimited rights associated with its content, so you do not need to specify rights.

You can create and edit rights in the:

- DRM Message window, which presents digital rights in a graphical user interface and can associate the rights with whatever content you have selected by automatically generating a UID and an Encryption key.

- DRM Rights editor, which presents the digital rights in XML format.

Digital rights have these parts:

- Permissions - general rights that fall into these categories: Play, Execute, Print, and Display.
- Constraints - limitations applied to a permission. For example, you can constraint the Play permission to a single execution of the content for ten minutes. Unless you set constraints, a permission is unlimited.

## How SDKs and Phones Apply Digital Rights

When an SDK or phone receives the rights to a content, the SDK or phone applies only permissions that are applicable to the content. In other words, if the content is music in an `.mp3` file, an SDK applies only Play permissions, regardless of what other rights are enabled. An SDK determines what rights it applies to a content based on its content-type header, which, in turn, NMIT generates from the extension of content's file.

The SDK and phone also control what it does when rights conflict. For example, if the rights end date is a week from now, but the rights interval constraint is one month, the specific DRM implementation of the SDK determine which right prevails. Typically, the more limited constraint prevails.

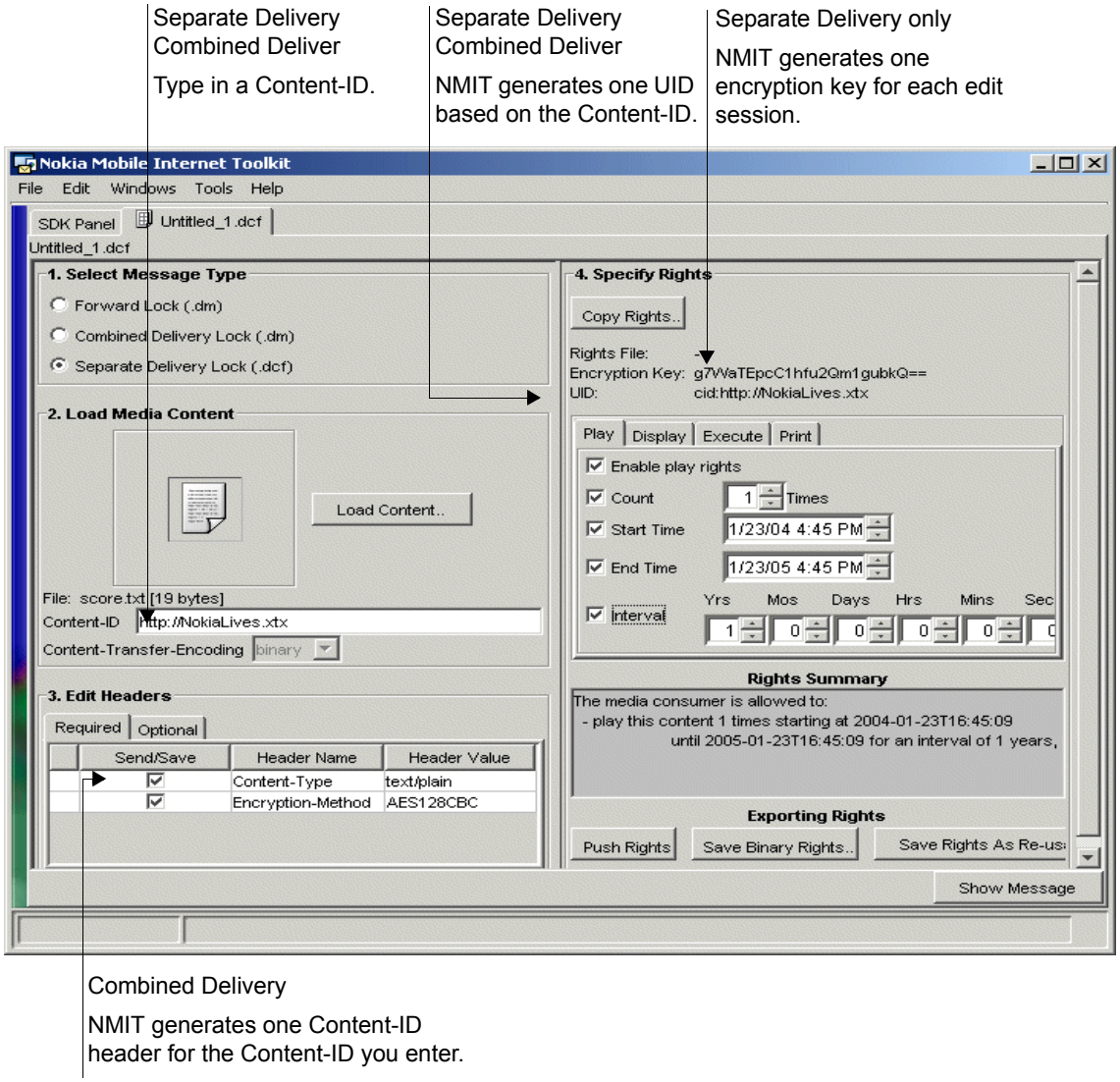
The SDK and phone must also have a mechanism to ensure that rights cannot be forwarded. A phone user cannot forward rights from a cell phone. A phone user cannot forward a Forward Lock or a Combined Delivery message because they contain rights. In a Separate Delivery message, the phone user can forward the content but not its rights.

## How Contents and Rights Are Matched

Content and its rights are linked internally with these identifiers:

| Identifier | Definition | Used in |
|---|---|---|
| Content-ID | A unique ID that pertains only to one content file in NMIT. You assign this ID to the content's file. | Combined Delivery<br>Separate Delivery |
| UID | A unique ID that pertains only to one content file in NMIT. The UID is taken from the Content-ID you assigned to a content's file. | Combined Delivery<br>Separate Delivery |
| Content-ID header | A required header that reflects the unique ID you assigned to a content's file in the Content-ID field when you selected the content's file. | Combined Delivery |
| Encryption key | A unique key that unlocks its associated content for use. This key is automatically generated by the DRM Message Editor and is linked to the UID of a file. | Separate Delivery |

The locations of these identifiers on the DRM window are shown below:




When a device receives several DRM messages, each with contents and rights, the rights must be correctly linked with specific content for which it is meant. Preserving this association is especially important in Separate Delivery messages where the content and its rights are sent separately.

NMIT associates the content/rights pair by giving the content and rights file the same prefix with a different suffix. For example, `nifty_tune.dcf` and `nifty_tune.dr`. are part of the same message. The DRM Message Editor automatically provides the name symmetry whenever you save a Separate Delivery message. When the UID in the Rights file does not match the Content-ID in the content file, NMIT displays an error message.

For the DRM Message Editor to find the pair of files, the `.dcf` and the `.dr` files:

- Must have the same prefix
- Must be located in the same directory

The DRM Message Editor does not consider `.drc` files when looking for a pair of files with which to create a Separate Delivery message.

### Opening a .dcf File Without an Associated .dr File

... is intentionally not possible.

NMIT treats a pair of `.dcf`/`.dr` files with the same prefix as single object When you try to open a `.dcf` file without its corresponding `.dr` file, NMIT considers the message corrupted.

A `.dcf` file can only be unencrypted with the matching Encryption key in the `.dr` file. When you open encrypted content (`.dcf`) from a Separate Delivery message, the DRM Message Editor automatically searches the directory for the `.dr` file that has the same prefix. If the DRM Message Editor does not find the rights file for the contents file, it will not open the contents file.

### Opening a .dr File Without a .dcf File

If you decide to open a digital rights file (`.dr`) separately from its content (`.dcf`), the rights file opens, not in the DRM window, but in the Digital Rights editor, which is an XML editor. The rights file displays the UID and the Encryption key, if they are present. See Working With Digital Rights in the Digital Rights Editor.

## Setting Digital Rights in the DRM Window

You can set these permissions only when you create a Combined Delivery or a Separate Delivery message:

- **Play** - controls permissions for audio and video files (for example, `.mp3`)
- **Execute** - controls permissions for executables (for example, `.jar` or `.exe`)
- **Print** - controls permissions for printing, if the cell phone is equipped to perform wireless printing. (for example, `.jpeg`, or `.gif`.)
- **Display**- controls permissions for viewing images (for example, `.jpeg` or `.gif`.)

The way you set each permission is identical. The rights you set are typically dictated by the content and the way you want the phone user to use the content.

### Enabling a Permission

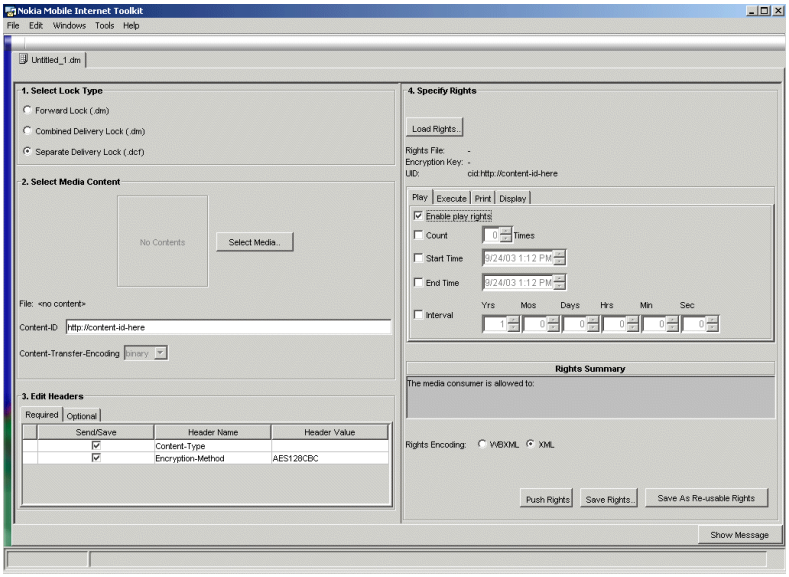
If you do not enable a permission, the phone user will have no rights to that permission.

To enable a permission:

1 On the left side of the DRM window, make sure you've selected one of the following:
   - **Combined Delivery**
   - **Separate Delivery**

2 On the right side of the DRM window, click the tab of the permission you want to enable (**Play**, **Execute**, **Print**, or **Display**).

3 To set unlimited rights within that permission, check **Enable <permission> rights**:




## Applying Constraints to a Permission

Constraints allow the content to be executed with a permission, but only under the terms you specify.

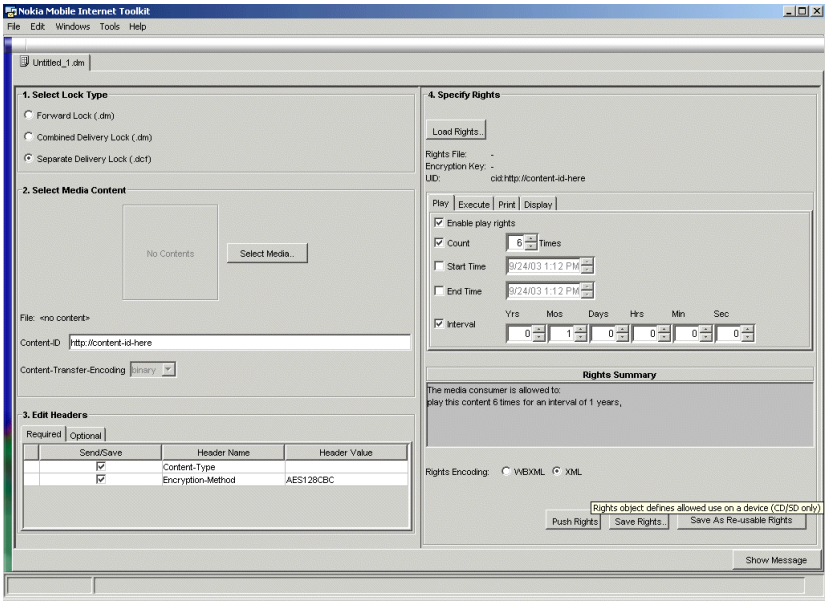
To apply constraints to an enabled permission:

1 Make sure **Enable <permission> rights** is checked.

2 Check the constrains you want to apply and enter a value for each one:

- **Count** - enter the number of times the content can be used
- **Start Time** - enter the date and time after which the content can be used
- **End Time** - enter the date and time after which the can no longer be used
- **Interval** - enter the maximum period of time during which a content can be used, beginning from the first use.

For example, these constraints let content be played six times within one month from the time of the first use:




A summary of the permissions and constraints appears under **Rights Summary** as you enter the rights.

## Exporting Digital Rights

When you are finished setting up permissions and their constraints, you can export the rights without content in these ways:

| Export Option | Available with | What the option does |
|---|---|---|
| **Push** | Separate Delivery | Sends the digital rights file(`.drc`) to the SDK you have selected. The WBXML encoded content is wrapped in an HTTP response message. |
| **Save Binary Rights** | Separate Delivery<br>Combined Delivery | Creates a WBXML-encoded digital rights file that is associated with the content.Use this option when you work with a push server that cannot encode Rights Expression Language files (.dr.) The suffix of this file type is `.drc`.The DRM Message Editor does not look for `.drc` files when it matches rights and content. |
| **Save as Re-usable Rights** | Separate Delivery<br>Combined Delivery | Saves the digital rights without any UID or Encryption key. This option lets you re-use the rights with different content. See Creating Re-usable Rights. |

# Working With Digital Rights in the Digital Rights Editor

You can edit any unencoded rights file (`.dr`) in the Digital Rights editor, which is an XML editor. You cannot open encoded rights files (`.drc`) in the Digital Rights editor.

## Creating a New Digital Rights File

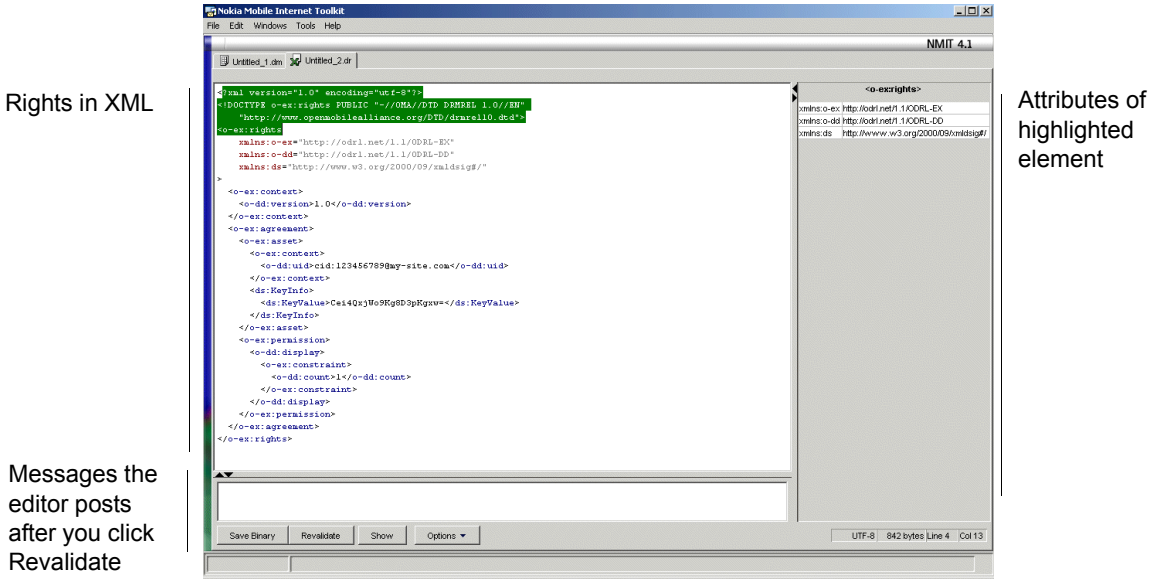
The Rights Editor automatically opens with the system default DRM template. If you have saved another Rights file as a template, that template automatically opens instead.

To open the Digital Rights Editor:

1 Select **File>New**. The **Available Content Type** window appears.

2 Click the **Deployment** tab to access the DRM Editors.

3 Click **DRM Rights**:




For more information about validating editors like the Digital Rights Editor, see Editing Features For Validatable Text Content.

## Opening an Existing Digital Rights File

To open an existing `.dr` file in the Digital Rights editor, select **File>Open** and select the `.dr` file you want. If the file you have opened has a UID and an encryption key, they are displayed in the XML text.

To create a re-usable rights file in the Rights Editor, delete the UID and the Encryption key, then save the file under a new name. See Creating Re-usable Rights.

### Setting Options

Click **Options** to set the output encoding of the rights file and to control some aspects of your workflow in the Rights Editor. The selections you make here override the settings in **Tools>Preferences**, which are used to initialize all editors.

For information about what these options do, see Options.

### Processing the Rights in XML

When you are finished editing the rights, you can do any of the following:

- **Revalidate** - Checks the syntax of the document conforms to the DTD. For more information, see Revalidate.
- **Push** - Sends the digital rights file(`.drc`) to the SDK you have selected. The WBXML encoded content is wrapped in an HTTP response message.
- **Save Binary** - Creates a WBXML-encoded digital rights file (`.drc`.). Use this option when you work with a push server that does not have capability for `.dr` files.

## Creating Re-usable Rights

The difference between a rights file that is associated with a content and a re-usable rights file is that the re-usable rights file:

- Does not have a value for the UID
- Does not have a value for the Encryption key
- Can be applied to more than one content after the UID and Encryption key are generated

To create re-usable rights in the DRM window:

1. Select one of the following so you can access the digital rights side of the DRM window:
   - **Combined Delivery**
   - **Separate Delivery**
2. Enter the permissions (**Print, Execute, Play, Display**) with the constraints (**Count, Start Time, End Time, Interval**) you want.
3. Click **Save as Re-usable Rights** and navigate to the place you want to store the re-usable rights file.

The digital rights file that is saved contains all the rights you entered, but omits the value for the UID and the Encryption key.

Another way to create re-usable rights is to open a rights file in the Digital Rights editor and delete the UID and Encryption key values. See Working With Digital Rights in the Digital Rights Editor.

# Copying Digital Rights

In the DRM Message Editor, you can copy the rights from any `.dr` file, including associated rights files and re-usable rights files into the current editing session. When you copy digital rights:

- Permissions and constraints are copied without the UID or Encryption key.
- The original rights are not changed.

To copy digital rights, click **Copy Rights** under **Specify Rights**.

# Creating Multiple Rights Files for a Single Content File

You can create multiple rights files for one content file in the DRM Editor or the Digital Rights Editor. This capability is useful when you want to provide consumers with assorted levels of rights for a single content.

## Creating Multiple Rights Files in the DRM Editor

Let's say you plan to have preview rights and unlimited rights for `nifty_tune.mp3`. In the DRM Editor, you can create a Separate Delivery message that packages `nifty_tune.mp3` with the preview rights and give the message pair the name of `nifty_tune_preview` - as in `nifty_tune_preview.dcf` and `nifty_tune_preview.dr`. You can then package `nifty_tune.mp3` with the unlimited rights and call that message pair `nifty_tune_unlimited` - as in `nifty_tune_unlimited.dcf` and `nifty_tune_unlimited.dr`.

The outcome of this approach is several `.dcf` files that:

- Are always paired
- Have the identical content because the DRM Message Editor uses the prefix of the Separate Delivery pair to recognize the `.dcf` and the `.dr` files as the two parts of one message

The other outcome is that every time you update the rights to one content, you have to resend the content again.

## Creating Multiple Rights Files in the Digital Rights Editor

You may want to be able to send rights files without having to resend a content file because:

- You can prevent duplication of content files.
- Rights files are generally smaller than content files and are therefore less costly to send.

To accomplish this scenario, you must create several `rights` files and associate them with a single content file:

1 In the DRM Message Editor, select **Separate Delivery**, create and save a `.dcf`/`.dr` pair (for example, `nifty_tune1x.dcf` and `nifty_tune1x.dr`, which lets the phone user play the tune one time as a preview).

2 In the Digital Rights Editor, open the `.dr` file you created in the DRM Message Editor (`nifty_tune1x.dr`).

   `Nifty_tune1x.dr` has an encryption key and a UID that associate the rights with `nifty_tune1x.dcf`. Do not change these in any way.

3 In the Digital Rights Editor modify the permissions and constraints (for example, in `nifty_tune1x.dr.`, make the Nifty Tune playable ten times in one month).

4 Save the `.dr` file with the modifications with a new file name (for example, `nifty_tune10x1m.dr`).

Now you have two `.dr` files (`nifty_tune1x.dr` and `nifty_tune10x1m.dr`) associated with one content file (`nifty_tune1x.dcf`). The files are associated by the UID and the encryption key, which are identical in both rights file.

After you send `nifty_tune1x.dcf`, you can send either `nifty_tune1x.dr` or `nifty_tune10x1m.dr` as the rights file. However, in the DRM Message Editor, you can open only `nifty_tune1x.dr` with `nifty_tune1x.dcf`. You cannot open `nifty_tune1x.dr` and `nifty_tune10x1m.dr` because their prefixes don't match, even though they have the same UID and encryption key.

## Saving a Template for the DRM Editor

As with the other editors, you can save the current content so that the editor is initialized with that content when you next open it or select **File>New**. When you save a template for the DRM Editor, some differences happen:

- Only one template is saved in the DRM Editor. The template is for one (not for one of each) of the following: Forward Lock, Combined Delivery, or Separate Delivery.
- For Separate Delivery, two files instead of one are saved as a template - `.dcf` and `.dr`.; however, the content of the `.dcf` file is not saved. All other editor settings are saved.

To save a template for the DRM editor, select **File>Save As Template**.

For more information about saving a template for an editor, see Save as Template....

# Download Descriptor Editor

A Download Descriptor is a file with information about downloadable material that can help an phone user decide whether to download the content. Such information includes the size, type, and location of the content. A download descriptor is not required for a DRM message but is a courtesy to the potential content user.

## About Download Descriptors

A Download Descriptor contains the following information:

| Attribute | Description |
|---|---|
| Object URI (mandatory) | The URI (usually URL) from which the user can download the content. |
| Content Type (mandatory) | The content type of the content.<br>This attribute indicates to the user whether the content to be downloaded is a media type that the device supports. |
| Size (mandatory) | The size in bytes of the content to be loaded from the URI.<br>The storage size and the execution size are dependent on the environment and may be different from the value of the size attribute. |
| Name | A name that identifies the content to the user. |
| Description | A short textual description of the content. |
| Vendor | The organization that provides the media object. |
| Info URL | A URL that describes the content. |

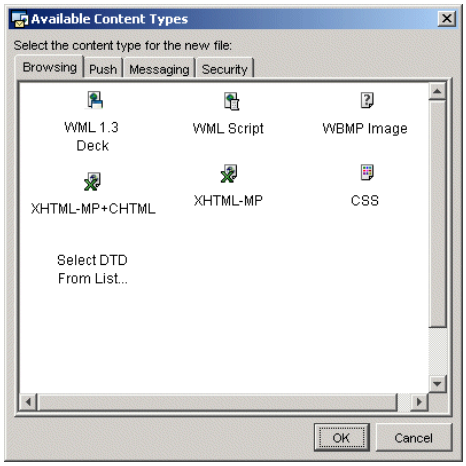
| Attribute | Description |
|---|---|
| Next URL | The URL to which the user can navigate when the end user chooses to browse after downloading the content.<br>Next URL provides a way for the download service to guide the user to continue with browsing actions. |
| Install URL | The URI (usually URL) to which a installation status report is sent, for a successful or failed download. |
| Icon URL | The URL of an icon that can be used by the client to represent the content (for example, an application) in the content installation user interface (that is, the application manager). |
| Install Parameters | Parameters specific to this download to be passed to the installation program. |

For more information about download descriptors and their attributes, see *Generic Download Over The Air (OMA-Download-OTA-v1_0-20030221-C)*. This document is freely available from http://www.openmobilealliance.org.

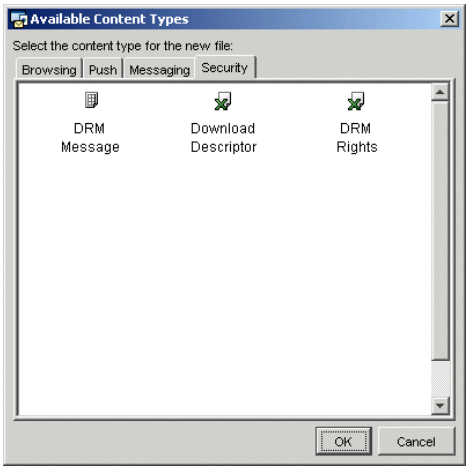
# Creating a Download Descriptor

To create a download descriptor:

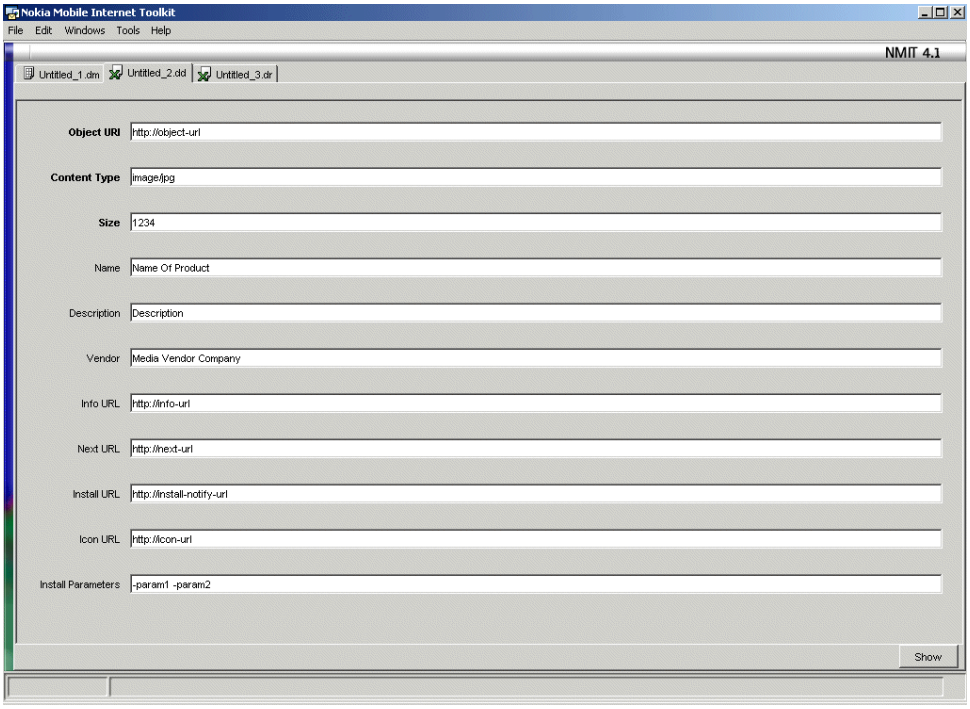
1 Select **File>New**. The **Available Content Type** window appears:



2 Click the **Deployment** tab:




1 From the **Deployment** panel, click the **Download Descriptor** icon, and click **OK**. The Download Descriptor window appears:




2 Enter information about the attributes you want.

**Object URI**, **Content Type**, and **Size** are mandatory. All other attributes are optional.

# Testing a Download Descriptor

To test a Download Descriptor, click **Show**.

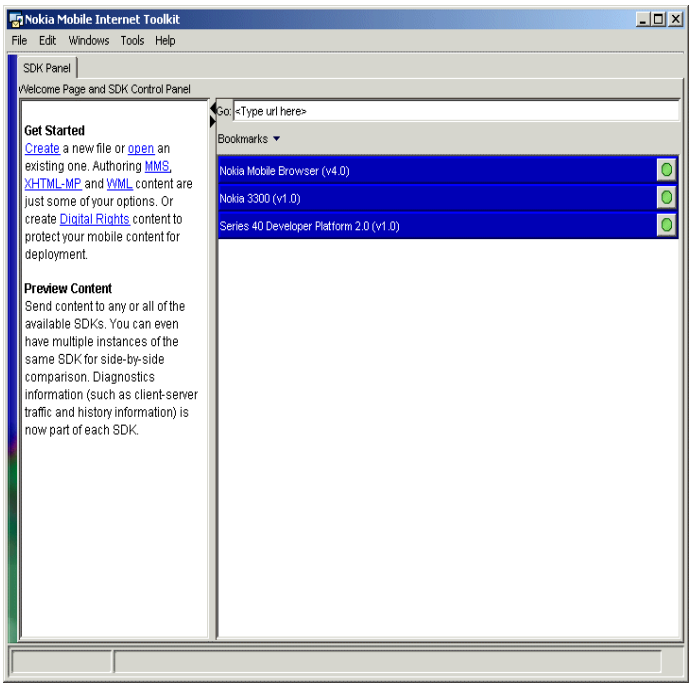
The file is sent to the SDKs you've selected that support this content. On the SDK, you can see content attributes, and you are able to download the content by clicking on the ObjectURL field. You may need a WAP Gateway to test various URL fields.

# Working With SDKs

The SDK Panel lets you:

- View all phone SDKs installed on your computer that work with NMIT.
- Launch and close any of the displayed phone SDKs.
- Enter a URL directly or from a bookmark and see it displayed on all running phone SDKs that have been launched from the panel.
- Create and save bookmarks so that you can conveniently display frequently accessed mobile Internet and local file content.

If the SDK Panel is not displaying when you first start NMIT, select **Tools>SDK Control Panel**. In the following figure, the phone SDKs are displayed. These SDKs were discovered (found) by NMIT upon launch:




Subsequent sections in this chapter cover the following topics:

[Discovery of Running Phone SDKs]

[Launching and Closing Phone SDKs]

[Displaying Editor Content on Phone SDKs]

[Browsing Mobile Internet and File Content]

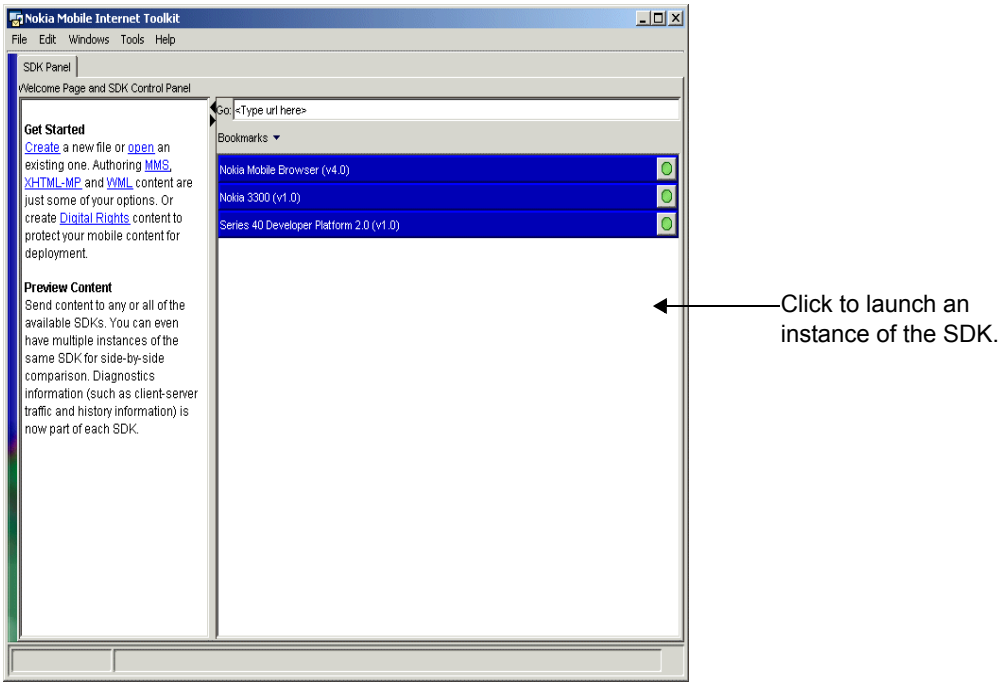
## Discovery of Running Phone SDKs

In general, you must launch one or more phone SDKs using the panel before you can display document content on the SDKs. However, NMIT also attempts to discover running instances of phone SDKs that were not launched using its SDK Control Panel. The following rules elucidate discovery:

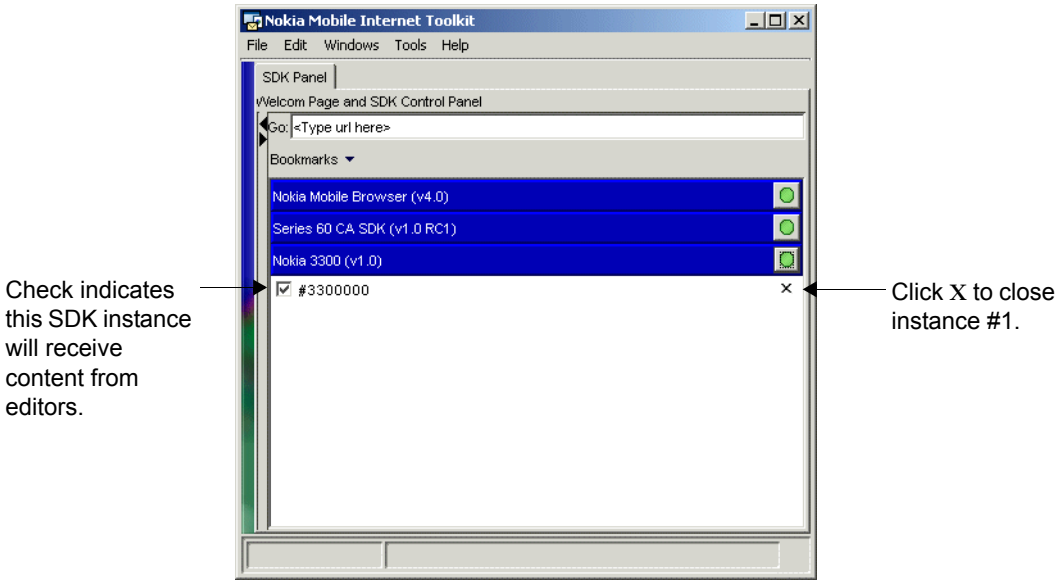
- Only running phone SDKs can receive content.
- If you first launch a phone SDK standalone or through another program and then launch NMIT, NMIT tried to discover these running instances and will list these in its SDK Control Panel.
- If you first launch NMIT and then launch a phone SDK standalone or through another program, NMIT does not “discover” it and does not list it in its SDK Control Panel. Thus, it is not possible to “refresh” the NMIT discovery procedure.

# Launching and Closing Phone SDKs

To start a phone SDK, click the button adjacent to its name.



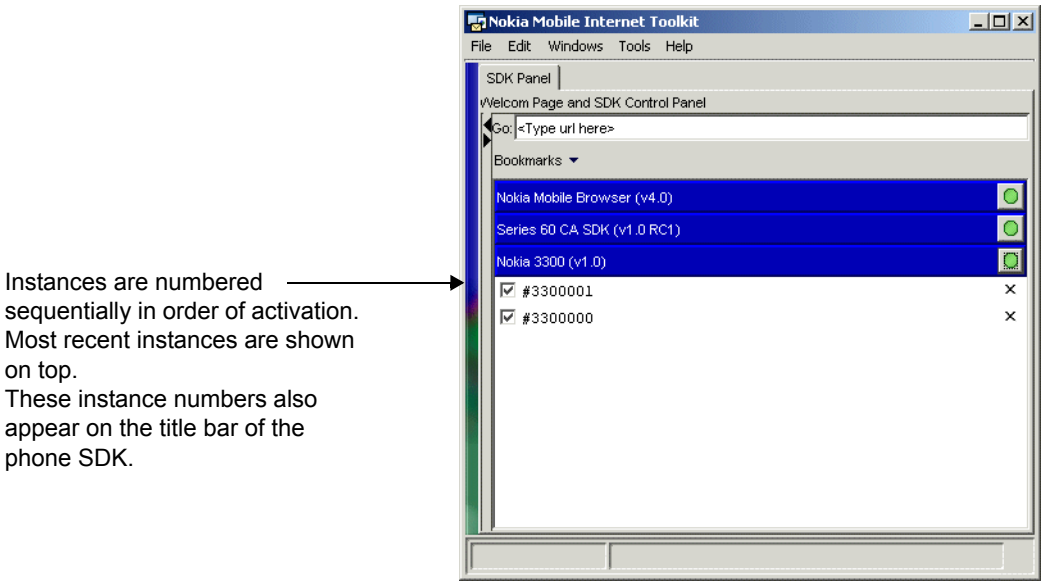

After clicking the button adjacent to Nokia 3300 SDK, NMIT launches the Nokia 3300 SDK. The panel displays this changed state as follows:




Click the **X** to the right of the instance number to close that instance.

You can launch multiple phone SDK instances if this is supported by the SDK. The Nokia NMB, 3300 SDKs support multiple instances; the Series 60 Content Authoring SDK does not. The

following figure shows the panel after clicking the button adjacent to Nokia 3300 a second time:




# Displaying Editor Content on Phone SDKs

This section describes the sending a document content from an NMIT editor to multiple SDKs. The procedure is the same for all of the document content types: WML, XHTML, Push, DRM, and MMS messages. The figure shown following the procedure depicts the sending of a WML document to both the Nokia 3510i and the Nokia 7210 SDKs.

1 Ensure that the phone SDKs to which you wish to send the document support the content type of the document you are sending. Some phone SDKs do not support some content types.

2 Launch each phone SDK that you wish to receive the content. An instance of the newly launched phone SDKs are displayed below the SDK names in the panel. By default, a check mark is placed to the left of the instance numbers, indicating they will receive any future document that you send.

3 Within NMIT, choose the **Show** button. As shown in the following figure, the phone SDK instances selected to receive the content display a “Loading” message status, and the content is displayed on both phone SDKs. Note also that the **Go** text entry bar in the SDK Control Panel displays the content currently being displayed.

**Note:** To display local files on the Series 60 Emulator, you might first need to launch Nokia WAP Gateway Simulator (NWGS).

# Browsing Mobile Internet and File Content

You can browse mobile Internet or file content and display this content on one or more SDKs. As with displaying content from NMIT editors, phone SDKs must be running and a check mark placed on the target SDK instances.

**Note:** To display local files on the Series 60 Emulator, you must first launch Nokia WAP Gateway Simulator (NWGS).

## Browsing File Content

To browse file content on a local or network drive, first ensure that the desired target SDK instances are checked and that they support the content type you would like to display (for example, the Nokia 3510i and 7210 SDKs support WML but not XHTML content). Then do any of the following:

- Type the full path of the file into the **Go** text entry box at the top of the SDK Control Panel, and then choose **Enter**.
- In Windows Explorer, navigate to the directory containing the desired file, click in the **Address** bar to select the directory path, paste this path into the **Go** text entry box at the top of the SDK Control Panel, append the file name to the pasted path, and then choose **Enter**. (This is a shortcut designed to save typing long path names.)
- In the SDK Active Control Panel, choose **Bookmarks** to display the Bookmarks Panel, and then choose one of your file bookmarks or choose **Samples Directory** to pick one of the NMIT-supplied files. See the section titled Using Bookmarks for more information about the Bookmarks Panel.

## Browsing Mobile Internet Content

To browse mobile Internet content:

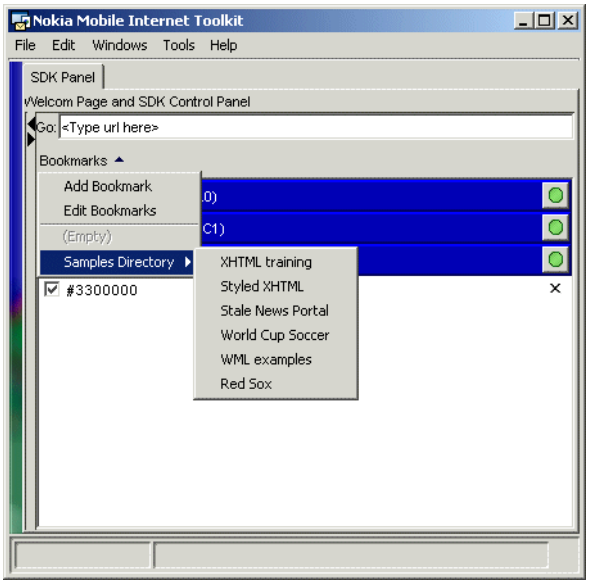
1. Launch the desired target SDKs using the SDK Control Panel and ensure that the resultant instances are checked if you wish those instances to display the content.
2. Configure each target phone SDK to use the desired WAP Gateway. This can be either an external WAP Gateway whose IP address you must know or the Nokia WAP Gateway Simulator (NWGS). By default, Nokia phone SDKs are configured to use NWGS whose IP address is 127.0.0.1. Check the WAP Gateway settings from within the phone SDK interface: for example, using the **Tools>Settings** menu item on the SDK menu bar.
3. If you are using NWGS as the WAP gateway, launch NWGS from within NMIT by choosing **Tools>WAP Gateway**. If your computer has access to the Internet through a proxy server, you will also need to configure NWGS to use this proxy server (use the NWGS **Settings>Proxy** menu option). (Note that NWGS is available as a separate download at Forum Nokia.)

4 Choose the URL to display using either of the following methods:

- Type the URL into the **Go** text entry box at the top of the SDK Control Panel, and then choose **Enter**.
- In the SDK Active Control Panel, choose **Bookmarks** to display the Bookmarks Panel, and then choose one of your bookmarks. See the section titled Using Bookmarks for more information about the Bookmarks Panel.

## Using Bookmarks

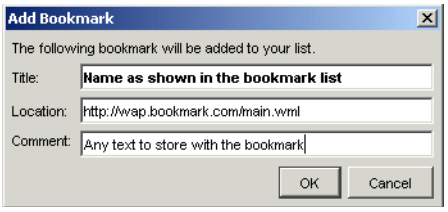
You can store the locations of URLs or local files as bookmarks and then easily select these for display on phone SDKs. NMIT enables you to access and organize your bookmarks using the **Bookmarks** button in the **SDK Control Panel**, shown below:




### Adding Bookmarks

You can add a bookmark in either of the following ways:

- Click or right-click the Add Bookmark button to display the following **Add Bookmark** dialog:



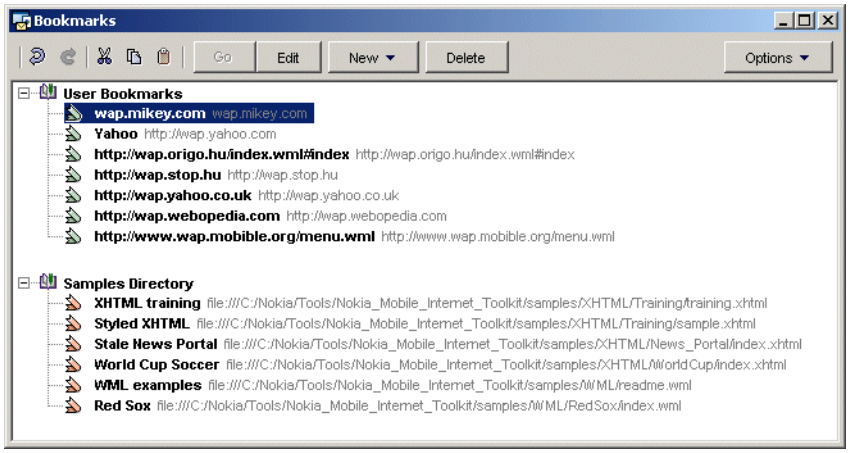

  In the **Title** text box, enter a name for the bookmark. In the **Location** text box, enter either a mobile Internet URL or a file URI in the format `file://C:/folder-name/file.wml` (or `.htm` or `.xhtm`). In the **Comment** text box, enter any comment you wish.

  Be sure to enter a complete URL in the **Location** text box; for example, do not omit the `http://` portion of the URL. Otherwise, the newly entered bookmark will be grayed out in the bookmarks list, preventing you from choosing it.

- Click or right-click the Edit Bookmarks button to display the **Bookmarks** dialog (see Editing and Deleting Bookmarks below). Then choose the **New** button to open the **Add Bookmark** dialog (described above).

## Editing and Deleting Bookmarks

To create, edit, and delete bookmarks, click or right-click the Bookmarks button on the SDK Control Panel and then choose the Edit Bookmarks button from the drop-down list. The Bookmarks dialog is displayed:

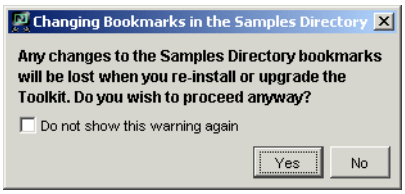
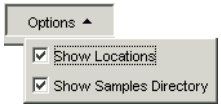



The following table describes the buttons which enable you to create, edit, delete, and otherwise manage your bookmarks.

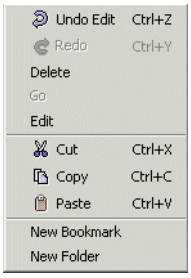
### Bookmarks Buttons and Other Controls

| | |
|---|---|
| Go | Displays the currently selected bookmark on phone SDKs. Note that the **Go** button is grayed out if NMIT does not recognize the bookmark URL. This might occur if you do not enter a complete Internet or file URI; for example, incomplete URIs such as `wap.yahoo.com` or `C:\myfile.xhtml`. |

### Bookmarks Buttons and Other Controls

| | |
|---|---|
|  | Opens a dialog in which you can edit the settings for a selected bookmark or a bookmark folder that you created. (You cannot edit NMIT's User Bookmarks or Sample Bookmarks folders.) |
|  | Displays the following drop-down menu:<br>• **New Bookmark** opens the **Add Bookmark** dialog (described above) in which you can enter a new bookmark. The new bookmark is added to your bookmarks list below the currently selected entry, so you may wish to first select an existing bookmark before creating a new one in order to position the new one correctly.<br>• **New Folder** opens a dialog in which you enter the name of a new bookmark folder. The new folder is added to your bookmarks list below the currently selected entry: if the entry is a folder, the new folder is created as a sub-folder; if a bookmark, as a sub-folder of the folder containing the bookmark.<br>Note that creating a new bookmark or folder within the **Samples Directory** folder results in the following warning dialog:<br>  |
|  | Deletes the selected bookmark or folder. Deleting a folder deletes all bookmarks in that folder. (Choose the  button to restore a mistaken deletion.) |
|  | Displays the following drop-down menu:<br>• **Show Locations** when checked causes the bookmarks list to display the URL location of each bookmark adjacent to its name.<br>• **Show Samples Directory** when checked displays the NMIT-supplied sample files as bookmarks. Unchecking this control removes this folder from the bookmark display. |
|  | Undo the previous action. |
|  | Redo the previous action |
|  | Cut the selected bookmark or folder to the Clipboard. |
|  | Copy the selected bookmark or folder to the Clipboard. |
|  | Paste the contents of the Clipboard |

### Bookmarks Buttons and Other Controls

| | |
|---|---|
| Single Click | On a bookmark or folder name, selects the bookmark or folder.<br>On a selected bookmark or folder name, opens that item for editing.<br>On a collapsed folder symbol ( ), displays the folder contents.<br>On an expanded folder symbol ( ), hides the folder contents. |
| Right Click | Displays the following popup menu. The **Undo** and **Redo** item names vary depending on the previous action.<br>  |
| Drag and Drop | Moves a selected bookmark or folder to another location in the bookmark list. A bold line indicates where the item will be placed when the mouse button is released. |

# DTD Manager

A document type definition (DTD) defines the elements that you can use within an XML document, including what elements can appear within the document with which attributes and what attribute values. Basically, a DTD is a set of rules that an XML parser uses to create a parse tree of a document.

The DTD Manager lets you manage the Document Type Definitions (DTDs) that NMIT editors use to validate document content.

NMIT can validate a document against only DTDs that are registered with NMIT. When NMIT is installed, a set of DTDs are automatically registered for use with NMIT editors. These are held in a private cache but are also stored in the directory `<NMIT_Install_Directory>\dtd`, which also contains other DTDs and DTD modules.

Using the DTD Manager, you can register additional DTDs and DTD modules, but you cannot modify or delete pre-registered DTDs.

If an XML-based document you open in an NMIT editor references an unregistered DTD, document validation will fail - that is, you will not be able to determine whether the document conforms to the rules specified by the referenced, unregistered DTD. NMIT never attempts to fetch an unregistered DTD over the network. Therefore, to validate the document, you must first register the DTD.

Some DTDs include various modules that define subsets of features; DTDs based on the XHTML specification, in particular, do this. These DTDs can be divided into two types:

- *Modular DTD*, which references one or more modules
- *Flat DTD*, which contains all definition in a single file without reference to any external modules.

You should always try to register a flat DTD rather than a modular version because flat DTD is easier to maintain and debug. (Many standards organizations offer both flat and module-based DTDs for the same content type.)

If a flat version of that DTD is not available, you can register the DTD and then use DTD Manager's **Add Module** button to include additional modules for the DTD.

The DTD Manager Interface is described in these sections:

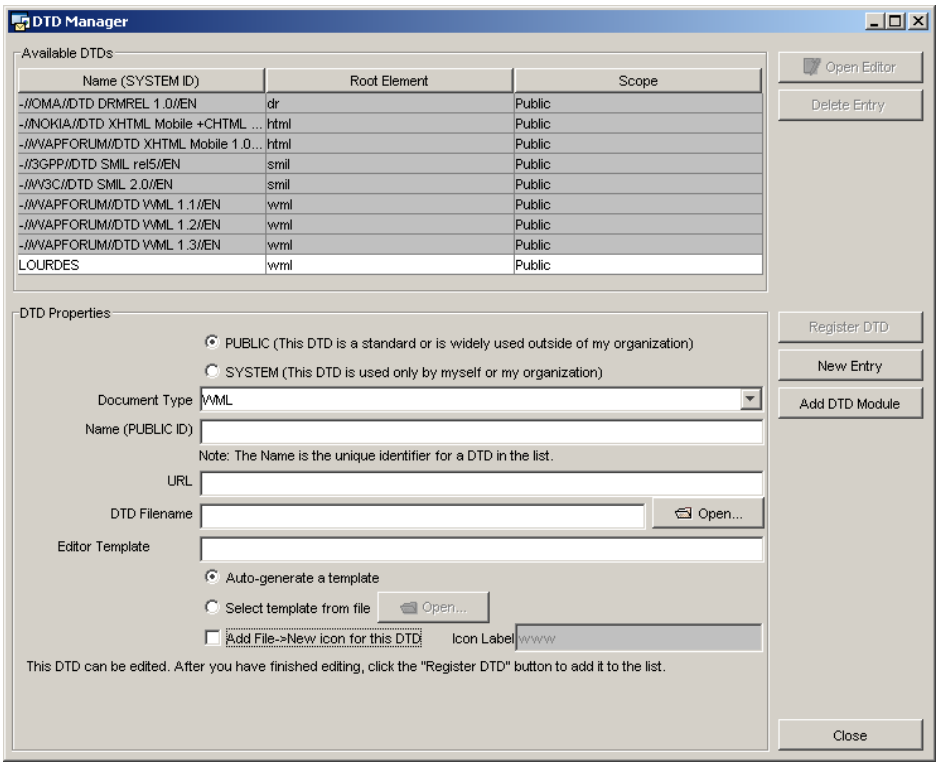


DTD Manager Task Procedures are described in these sections:



# DTD Manager Interface

The DTD Manager opens in a separate window whenever you choose **Tools>DTD Manager**:




## DTD Manager Available DTDs Region

The Available DTDs region displays all currently registered DTDs, giving the name, XML root element, and scope of each. You cannot directly modify or delete the data in this table.

Whenever you register a DTD, the DTD is added to the Available DTDs region. NMIT also provides six installed DTDs. The following table describes the columns of this table:

| Column Name | Description |
|---|---|
| **Name** | Specifies a unique string that identifies the DTD to NMIT is used by other XML-based processor. Conventions for specifying this name is given in the Table "DTD Properties" (below). |
| **XML Root Element** | Specifies the root element for the document, which is the XML element that completely encloses all other elements defined by the DTD. For example, the root element for the XHTML Mobile Profile DTD is `html`. |
| **Scope** | Specifies whether the DTD is a Public DTD or System DTD. A description of the differences between these is given in the Table "DTD Properties" (below). |

## DTD Manager DTD Properties Region

The DTD Properties region:

- Displays information about the DTD you select in the Available DTDs region.
- Lets you enter information about the DTD when you register a new DTD

This table describes the fields in this area:

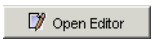
| Field | Description |
|---|---|
| **PUBLIC** | Click the **PUBLIC** button if this DTD is intended to be used by content developers outside your organization. "PUBLIC" is an XML keyword that will be specified in the `DOCTYPE` declaration of a document using the DTD. Industry standard DTDs are typically PUBLIC DTDs. |
| **SYSTEM** | Click the **SYSTEM** button if this DTD is intended for use by a single author or group. "SYSTEM" is an XML keyword that will be specified in the `DOCTYPE` declaration of a document using the DTD. |
| **Document Type** | Select the document type of the document. The document type indicates the XML element name used to demarcate the content body in a document. For example, the name of the XML root element for the XHTML Mobile Profile DTD is `html`. All content except the XML prologue and the `DOCTYPE` declaration is enclosed within an XML-based document's root element tags.<br>The XML Root Element determined by the document type you select is displayed under XML Root Element in the Available DTDs region. |

| Field | Description |
|---|---|
| **Name** (**PUBLIC ID**)<br>(for Public STSs only) | Enter a unique string that identifies the DTD. NMIT displays this name in the Available DTDs region after a DTD is registered.<br>If you have downloaded a DTD, you will need to open the file (using, for example, Windows Notepad) and search for this name which is usually located near the top of the file within a comment.<br>There is no required syntax for name text. However, there is a convention for specifying the name for a public DTD, which you can see by looking at any of the pre-installed public DTDs in the **Available DTDs** region. This convention calls for (1) the string "ISO" if the DTD is an ISO standard, or a plus sign (+) followed by the name of a non-ISO standards body, or a hyphen (–) if no standards body has approved the DTD, (2) two slashes (//), (3) the name of the DTD owner, (4) two slashes (//), (4) the type of the document the DTD describes, (7) two slashes (//), and (8) an ISO 639 language identifier. |
| **URL** | Enter the URL specifying the location of the DTD file. What you enter in this field depends on whether the DTD you are registering is a Public or System DTD:<br>• For a Public DTD, enter an absolute or relative URL from which any XML processor (such as NMIT) may locate the DTD in the event the DTD is unavailable locally. Note that NM IT should not need to use this URL after you have registered it.<br>• For a System DTD, enter the URL used by documents to reference this DTD. Registering network DTDs (DTDs whose URL begins with the characters `http://`) enables NMIT to validate against these DTDs without incurring the latency associated with network access. The string you enter must match the URL string referenced by XML-based documents. |
| **DTD Filename** | Enter the local file name where the DTD file is located, or choose the **Open** button to navigate to this location. |
| **Editor Template** | Enter the file name of the template file for this DTD. Each DTD must have an associated template file. This file serves two purposes: (1) It is displayed in NMIT's content editing region of any new document based on the associated DTD, and (2) NMIT's validating parser requires its presence to function correctly as it uses it only to validate.<br>The template file name must end in one of the following XML-based suffixes; only these are supported by NMIT: `.html`, `.smil`, `.wml`, `.xml`, `.xhtml`,`.dr`.<br>To select a template, choose one of the following:<br>• **Auto-Generate Template**. NMIT will attempt to generate the file for you automatically. If the generated template fails, you must create your own template and reregister the DTDs.<br>• **Select Template From File**. Now you can either enter the file name directly into the text-entry box, or choose the **Open** button to navigate to its location. |

You can also add an icon that represents the DTD in the tabs you see when you select **File>New** by checking **Add File>New icon for this DTD** and entering a name for the icon. When you click the icon in the **File>New** editor tab, the XML editor opens with the template that references the DTD you registered.

## DTD Manager Button Controls

You use the following buttons, positioned along the right side of the DTD Manager window, to perform DTD operations.

| | |
|---|---|
|   | Choose **Open Editor** to open a new document referencing the DTD you have selected in the adjacent **Available DTDs** region. The NMIT Editor Window is displayed, showing the template that is associated with the DTD. |
|  | Choose **Delete Entry** to delete a user DTD entry from the table of DTDs listed in the adjacent **Available DTDs** region.<br>A user DTD is a DTD that you have previously registered with NMIT. You can delete user DTDs, but none of NMIT's pre-installed DTDs. When a pre-installed DTD is selected, the **Delete Entry** button is inactive.<br>When you delete a DTD, the copy NMIT uses that is not the disk is also deleted. |
|   | Choose **Register DTD** to register a DTD with NMIT. You do this after you have entered data for all the DTD properties in the **DTD Properties** region.<br>NMIT copies the DTD information you specify into its DTD database file where information about the DTD is persistently stored, and it copies the DTD file and templates into its local DTD cache.<br>The step by step procedure to follow when registering a DTD is provided in the section titled Registering a New DTD. |
|   | Choose **New Entry** to clear all fields in the **DTD Properties** region so that you can enter fresh data. Note that the **New Entry** button does not perform an operation on a DTD; it simply clears all text entered in the **DTD Properties** region. |
|   | Opens the **Select DTD Module File** dialog in which you can navigate to a local module file (`.mod`) that you wish to copy to the `<NMIT-Install-Directory>\dtd` folder. Module files referenced by DTDs must reside in the same `\dtd` folder as the DTD itself. Refer to the section titled Adding a Module to a DTD for additional information. |

# DTD Manager Task Descriptions

This section describes the following typical tasks you can perform in the DTD Manager:

Registering a New DTD

Testing Document Content Validity Against Multiple DTDs

## Registering a New DTD

Use this procedure to register a DTD that NMIT does not yet recognize. Once you have registered a new DTD, you can create documents based on that DTD and then use NMIT's validating parser to check for their conformance to the DTD rules.

Each registered DTD uses an accompanying document template, which you see in the editor's content editing window when you create or open a file based on that DTD. Information about the DTD must be included in this template, specifically the `DOCTYPE` declaration. This declaration has the following syntax:

```
  <!DOCTYPE root-element-name PUBLIC "DTD_name" "DTD_URL">
or
  <!DOCTYPE root-element-name SYSTEM "DTD_URL">
```

When you register a DTD, you will be entering the information that you see in the `DOCTYPE` declaration (above); these are the properties of the DTD. This information is described in the section titled DTD Manager DTD Properties Region.

The registration procedure described here is a two-step process:

- Acquiring the DTD
- Collecting the information you will need before beginning the registration process

  Usually, this will mean downloading a DTD, but you might also be creating a custom DTD to reflect a specific device. In either case, you will want to open the DTD in an editor such as Windows Notepad. Then, you can easily search the DTD for the information you'll need during the registration process.

If you have created your own DTD, you will already know the properties of your DTD. In this case, you can skip the first step (collecting the registration data) and immediately register the DTD.

You can refer to the property descriptions provided in DTD Manager DTD Properties Region as you follow the procedures below.

### Collecting Required Registration Data

To collect the required registration data for a downloaded DTD

1 Download the DTD. Save the file in a local directory such as the `\temp` directory. It is best to download a flat (non-modular) DTD if such is available. If, however, the DTD includes other modules, these modules must be placed into the `\DTD` subdirectory of your NMIT installation directory. Check this directory before downloading any modules because there are a number of standard modules already provided there. Download any modules still needed, so that when you are finished all can be found in the `\DTD` directory.

2 Determine whether the DTD is a PUBLIC or SYSTEM DTD.

3 Determine the reference information. For a PUBLIC DTD this is the name and URL; for a SYSTEM DTD, just the URL.

4 Create a template for the DTD. Though you can use NMIT's auto-generation feature, it is recommended to create your own. See the **Editor template** item in the DTD Manager DTD Properties Region for information about creating a template.

## Registering a DTD

To register a new DTD:

1 Choose **Tools>DTD Manager** to open DTD Manager.

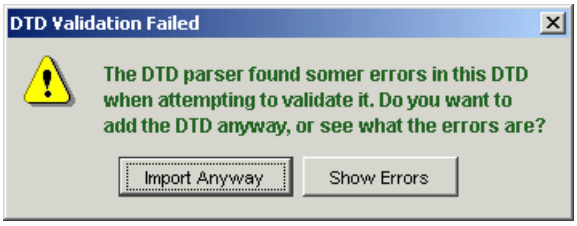
2 Choose the **New Entry** button to clear the fields displayed in the **DTD Properties** region.

3 Fill out all fields in the **DTD Properties region**. In doing this, you may be referring to the information you collected as you performed the preceding procedure titled "To collect the required registration data for a downloaded DTD."

4 Choose the **Register DTD** button. NMIT then validates the specified template file against the DTD file.

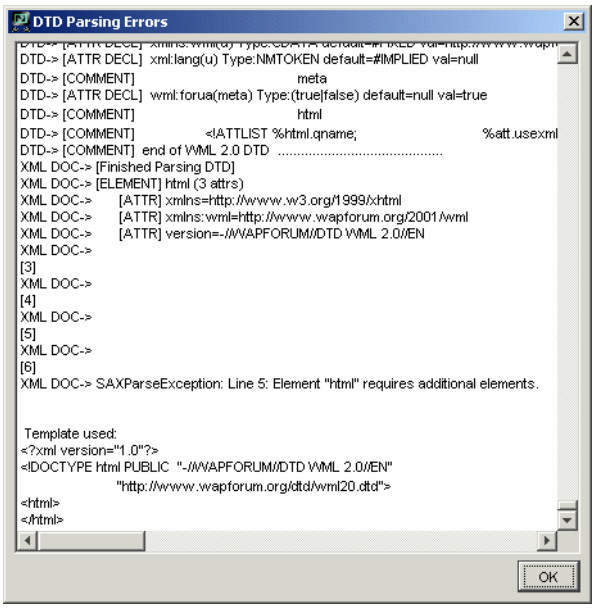
If validation is successful, NMIT copies the (local) DTD file into the `\DTD` directory and stores other information into a separate DTD database file. You will then see the newly registered DTD displayed in the **Available DTDs** region.

However, if validation fails, you will see the error message shown below. In this case, you may need to edit either the DTD or the template (whether or not you chose to auto-generate the template) and then try to register the DTD again.




Choosing **Import Anyway** in the above dialog causes NMIT to register the DTD despite the errors found. If you do this, validation errors may occur later when you attempt to validate new or existing content against the DTD. Therefore, it is sensible to do this only if you have diagnosed the problem and plan to resolve it by, for example, copying a required `mod` file to the `\DTD` directory or editing the DTD itself in its newly registered location within the `\DTD` directory.

Choosing **Show Errors** in the above dialog displays the **DTD Parsing Errors** log (shown below) produced by NMIT's validating parser, and aborts the registration.




Inspecting the log may help you locate the source of the parsing error so that you can edit the DTD. Errors are shown at the end of the error log and will specify the line numbers of offending elements. Note that these line numbers refer to the template file, not the DTD file.

Keep in mind that errors may lie in either the DTD or in the template. If you trust the source for the DTD and you have included all `mod` files that the DTD requires, then you may adopt a working assumption that the error is in the template.

The template file is located in the `<NMIT Install Directory>\dtd` directory. If you have elected to have NMIT auto-generate the template, the template file name is the name of the DTD file but with the characters `.template` appended to the end of the name. For example, for the NMIT-supplied DTD `wml2.0.dtd`, the template name is `wml2.0.dtd.template`. You can open the template file with Windows Notepad or other editor to check for errors.

The template must include all required elements but not all elements in a DTD are required. So, first ensure that your template includes all required elements.

If you have used NMIT's template auto-generation feature to generate the template, this template will include only the required root element. So, if your DTD specifies any required elements as child elements of the root element, a template for this DTD generated by NMIT will fail validation. In such a case, simply open the template file and include the necessary required elements; then choose the **Select template from file** radio button and then **Open** to choose the template file you just edited.

## Testing Document Content Validity Against Multiple DTDs

When editing a document based on any of NMIT's XML-based content types, you can check to see if the document contents would be valid if a different DTD were referenced within that document. To do this, it is necessary to replace the DTD name in the `DOCTYPE` declaration with the new DTD name you wish to check. This might prove helpful, for example, if you were creating an

XHTML Mobile Profile file but also wanted to check if the file would be valid using the XHTML Mobile Profile + CHTML DTD. The procedure is as follows:

1 With your original file open in one of NMIT's XML-based editors, place the cursor anywhere within the `DOCTYPE` declaration (the second line from the top). In the editor's side panel to the right of the content region, you see the DTD drop-down list whose list entries are all the registered DTDs that have a root element that matches that of the current document, as shown below:


2 Click the down arrow to reveal the available DTDs, and then select the one against which you wish to validate the current content. NMIT inserts the selected DTD name as the value of the `DOCTYPE` declaration in your original file.

3 Choose the **Revalidate** button at the bottom of the window to validate the content against the new DTD.

4 Inspect the Message Region below the text editing region in an NMIT editor to check for error or warning messages. The presence of an error message indicates the content is not valid against the newly substituted DTD; otherwise, it is valid. You may wish to inspect further any warning messages, though their presence does not affect the validity of the document.

## Adding a Module to a DTD

A *modular DTD* references one or more modules, whereas a *flat DTD* is a DTD in which all definition is included in a single file, without reference to any external modules. DTD designers of certain DTDs elected to subset DTD element, attribute, and entity definition into modules, mostly because this enabled ease of code re-use, that is, the sharing of a module among multiple DTDs.

You can register either a flat DTD or modular DTD with NMIT. Registering a flat DTD is simpler because no modules are involved. However, when you register a modular DTD, you must ensure that all modules referenced by that DTD are copied to the `\dtd` folder along with the DTD itself. The **Add Module** button in DTD Manager is provided as an aid in copying modules into the `\dtd` folder. You could, alternatively, use Windows Explorer to copy modules into the `\dtd` folder.

You should copy all modules into the `\dtd` folder before you register a modular DTD. This is because when you register the DTD, DTD Manager parses the DTD file in order to resolve references. If DTD Manager encounters a reference to a module, it attempts to locate that module in the `\dtd` folder. If the module is not present, a parsing error is generated.

In the face of a parsing error, you could choose to proceed anyway with importing the DTD; however, this is not recommended. After all, if there are parsing errors with the DTD itself, you will never be able to successfully validate a document against that DTD, that is, errors would always be generated. In such a case, you would have difficulty determining whether the error resided in the document or in the DTD.

In sum, when registering a new modular DTD:

1 Copy all modules related to that DTD into the `\dtd` folder. Use either DTD Manager's **Add Module** button or Windows Explorer.

2 Register the DTD.

3 Debug any errors encountered during registration.

## Upgrading a Registered DTD to a Newer Version

If a DTD is already registered with NMIT, it is displayed in DTD Manager. Such DTDs are either DTDs that you have registered or DTDs that were pre-registered with the NMIT installation. You can upgrade only DTDs that you yourself have registered.

You can upgrade a registered DTD selecting the DTD you wish to upgrade and choosing the **Delete Entry** button. Then register the newer version as you would a new DTD.

# WAP Gateway

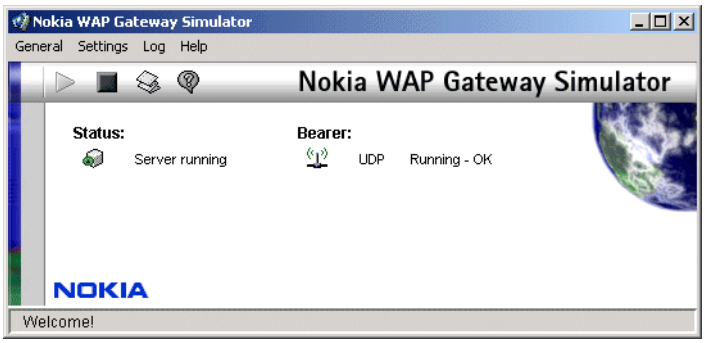
The Nokia WAP Gateway Simulator a single-user WAP gateway, is a separate application, not integrated with NMIT, and is summarized only in this chapter.

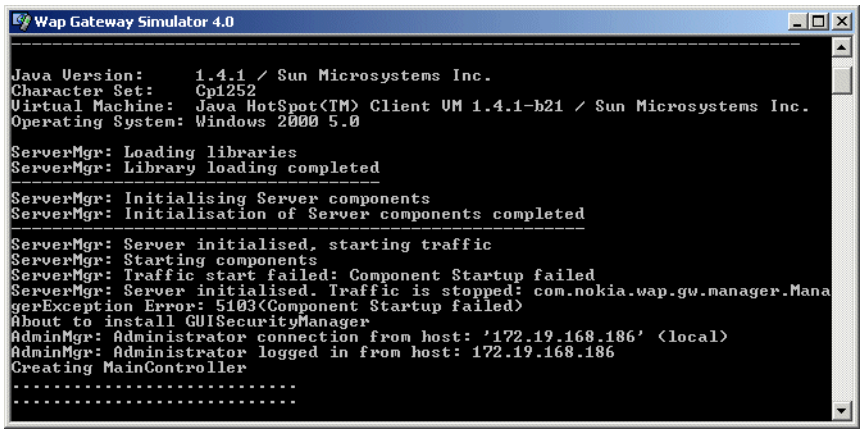
Choosing NMIT's **Tools>WAP Gateway** menu option launches Nokia WAP Gateway Simulator (NWGS). This is a single-user WAP Gateway based on the multi-user Nokia Activ Server. When installed on your computer, NWGS enables you to access the mobile Internet through the phone SDKs that you use in conjunction with NMIT.

NWGS provides a subset of the features provided by Nokia Activ Server such as the following:

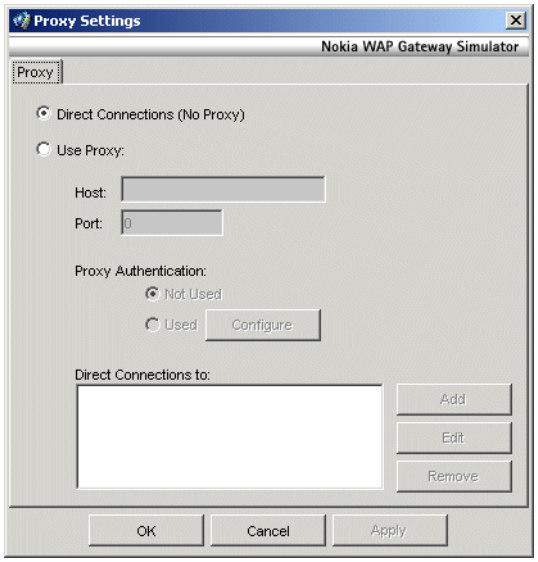
- WAP 2.0 compliance with support for `wml`, `wmlscript`, `xhtml`, `css` and Push message content types.
- Encoders. NWGS encodes WML and WMLScript content on the way from the Origin Server to the client phone SDK.
- UDP/IP bearer adapter. This adapter enables communication between NWGS and other client user agents running in a local area network, such as Nokia Mobile Internet Toolkit (NMIT) and phone emulators. NWGS does not support other bearer adapters supported by Nokia Activ Server as these are designed to enable radio communication between devices and the mobile Internet.
- User administration through a GUI, providing the ability to stop and start traffic, configure proxy and cache settings, and view and configure log file settings.

Upon launch, NWGS displays both its Administration window and its server application running in a Command Prompt window, as shown below:




Depending on your network configuration, you may need to specify an HTTP proxy server. This would be the case, for example, if your computer were located within a corporate Intranet that used an HTTP proxy server as a gateway to the Internet. If so, choose the NWGS menu option **Settings>Proxy** and then enter the host and port for the proxy in the dialog, shown below:




For additional information about NWGS, see the *Nokia WAP Gateway Simulator User's Guide*, which is accessible from the NWGS menu option **Help>User's Guide**.

# Appendix: HTTP File Response Format

NMIT supports a new Nokia-proprietary file format that is based on the HTTP 1.0 response message. This file format (`.http` file type and its associated content type, `application/vnd.nokia.http-response`)enable Push editors to transmit user-edited HTTP headers along with the content to phone SDKs using the file system.

See RFC 19455 "Hypertext Transfer Protocol Specification 1.0" for a full definition of an HTTP response message. Note that the later RFC 2068 (HTTP 1.1) does not add relevant changes for the purposes of this file definition. The format is summarized below.

The file format consists of a `Status` line, a `Header` section, and a `Body` section. Each of these parts is separated by a CRLF character sequence. The status line and headers are ASCII-encoded strings.

A `status` line (which MAY be ignored by readers) has the following format:

```
Http-Version SpaceChar Status-Integer SpaceChar ReasonString
```

The `header` section consists of a series of headers. Each headers is of the form:

```
HeaderName ":" HeaderValue CRLF
```

The only required header in the header section is the `Content-Type` header. The header section is terminated by a second CRLF.

The `body` section is the message content. It has the format specified by the `Content-Type` header, including any character encoding.

Note the following:

- Multiple response messages in a single file (as defined in HTTP 1.1 for a single stream) is not supported in this format.
- Readers of these files SHOULD support any of 1.x versions of HTTP indicated on the status line, but MAY ignore the line altogether. NMIT will write the string "`HTTP/1.1 200 OK`" as a status line for all push files that it creates.

- Readers of these files SHOULD handle CR being the line separator character sequence (instead of CRLF). Supporting this would allow users to hand-edit files on systems (specifically, UNIX-based) that use CR as line separators.

The following is a simple example of the HTTP Response file format:

```
HTTP/1.0 200 OK
Content-Type: text/plain

This is a text message.
```

Some Nokia phone SDKs such as the Nokia 7210 and the Nokia 3510i do not recognize the `.http` file type or its associated content type, `application/vnd.nokia.http-response`. To compensate for any SDK that does not recognize `.http`, NMIT prepends the file URL with the string `http://Toolkit/`. This string causes the file to be loaded through the Nokia WAP Gateway Simulator (when NWGS is running).

# Glossary

**API (Application Programming Interface)**

An API is a set of functions within a module of a program that are available to other modules and which when called by other modules provide access to the functions performed by that module.

**ASCII (American Standard Code for Information Interchange)**

ASCII is a standard developed by the American National Standards Institute (ANSI) to define computer-intelligible values for characters used in text. The ASCII set of 128 characters includes upper-case and lower-case letters of the English alphabet, numbers, punctuation, and 33 control codes (such as tab, bell, carriage return). ASCII uses 7 bits to represent each character. You may see ASCII characters identified by a decimal number from 0 to 127.

The standard ASCII character set uses just 7 bits for each character, consequently one bit of each octet is not used. Larger character sets, known as extended ASCII or high ASCII, use all 8 bits, allowing as many as 128 additional characters to be defined. Numerous extensions to ASCII have been devised and quite a few have become national or international standards. Notable among them is a family of international standards, ISO-8859, that defines extensions appropriate to certain language groups which ASCII alone cannot support. The most important member of this group is ISO-8859-1, known as ISO Latin-1, which provides for the languages of western Europe.

**Attribute**

A syntactical component of an XML element.

**Author**

An author is a person or program that writes or generates WML, WMLScript or other content.

**Bandwidth**

Bandwidth is the capacity that a telecommunications medium has for carrying data. For analog or voice communication, bandwidth is measured in the difference

between the upper and lower transmission frequencies expressed in cycles per second, or hertz (Hz). For digital communication, bandwidth and transmission speed are usually treated as synonyms and measured in bits per second. The actual speed or transmission time of any message or file from origin to destination depends on a number of factors. Most Internet transmissions travel at very high speed on fiber optic lines most of the way. Switching en route, lower bandwidths on local loops at both ends, and server processing time add to the overall transmission time.

**Bearer**

A service that enables over the air transmission of digital data. Bearers support different protocol-based data formats such as SMS (Short Message Service), CSD (Circuit Switched Data), and so on.

**Bearer Adapter**

A software module (an implementation of WDP) that enables the WPS (Wireless Protocol Stack) to work with a particular bearer.

**Byte**

A sequence of consecutive bits treated as a unit. On almost all modern computers, a byte is comprised of 8 bits, though other numbers were formerly encountered. To avoid ambiguity, the term octet is used in the language of international standards to refer to an 8-bit unit.

Large amounts of memory are indicated in terms of kilobytes (1,024 bytes), megabytes (1,048,576 bytes), and gigabytes (approximately 1 billion bytes). A disk that can hold 1.44 megabytes, for example, is capable of storing approximately 1.4 million ASCII characters, or about 3,000 pages of information.

**Bytecode**

Content encoding where the content is typically a set of low-level opcodes (that is, instructions) and operands for a targeted hardware (or virtual) machine

**Card**

A single WML navigational and user interface unit. A card may contain information to present to the user or instructions for gathering user input, for example.

**Character Encoding**

The process of mapping a character to a numeric value. The more bits available for the numeric value, the more characters can be mapped. The default encoding scheme used by XML processors is ISO 10646 UTF-8, which is 8-bit character encoding.

**Character Set**

A table of characters, mapped to numeric values, which are stored and used as a unit (see Character Encoding).

**Client**

A device or application that initiates a request for connection with a server.

**CGI (Common Gateway Interface)**

A programming language that enables you to use forms on your web site.

**Concatenation**

Appending one string to another to make a new string. For example, the string “foo” concatenated with the string “bar” gives the string “foobar”.

**Content Dispatcher**

The Content Dispatcher is a general purpose module designed to dispatch messages to the destination content handlers (WML, WMLScript, Multipart, SI, and SL handlers). Upon startup, the Content Dispatcher registers with the Application Dispatcher for all messages whose content type is supported by the WAE.

**Content Encoding**

When used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.

**Content Generator**

A service that generates or formats content. Typically content generators reside on origin servers.

**CSS (Cascading Style Sheet)**

A set of rules which determine the way in which a browser displays the elements composing an XML-based document.

**Deck**

A collection of one or more WML cards contained in a file of type wml. A WML deck is also an XML document.

**Device**

A network entity that is capable of sending and receiving packets of information and which has a unique device address. A device can act as both a client or a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server

**DRM (Digital Rights Management)**

A (still evolving) system used to express and enforce rights over digital content.

**DTD (Document Type Definition)**

A document type definition defines the elements that are permitted within an XML document, where these elements may appear within the document, which attributes and attribute values are permissible, and so on. Basically, a DTD is a set of rules that an XML parser uses to create a parse tree of a document, which is required for further processing.

**ECMA**

European Computer Manufacturer's Association. See http://www.ecma-international.org/ for more information.

**ECMAScript**

A script language developed by the ECMA and defined in the ECMA-262 specification. ECMAScript is based on JavaScript. See http://www.ecma-international.org/ for more information.

**Element**

An entity in XML that describes the structure and content of information. Elements may contain a start tag, content, an end tag, as well as one or more attributes.

**Extensible Markup Language (XML)**

The Extensible Markup Language is a World Wide Web Consortium (W3C) standard for Internet markup languages, of which WML is one such language. XML is a restricted subset of SGML.

**Flat DTD**

A DTD that includes all required element, attribute, or entity definitions in a single file and makes no references to external modules.

**GIS (Geographic Information System)**

Both the set of tools used to gather, transform, manipulate, analyze, and produce information related to the surface of the earth, as well as the delivery mechanism for such information (database, desktop software, etc.). GIS systems are usually tailored to the needs of certain types of users; for example, police, climatologists, and so on.

**GPS (Global Positioning System)**

A system whereby transmission of radio signals from a satellite to a receiving device is used to determine the position of that device.

**GSM (Global System for Mobile Communications)**

One of the leading digital cellular systems, GSM uses TDMA technology and allows eight simultaneous calls on the same radio frequency.

**GUI (Graphical User Interface)**

The set of graphical menus, dialogs, and windows through a which a user has access to the functions of a computer application.

**HTTP (Hypertext Transfer Protocol)**

HTTP is the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions web servers and browsers should take in response to various commands.

**IPC (Interprocess Communication)**

A capability supported by some operating systems that allows one process to communicate with another process. The processes can be running on the same computer or on different computers connected through a network.

**JavaHelp™**

An online help system generated by a set of Java class files.

**JavaScript™**

A standard language that can be used to add dynamic behavior to HTML documents.

**JVM (Java Virtual Machine)**

A utility that interprets and displays the results of Java code.

**Location Server**

A server that translates location information received from a mobile entity (via a WAP Gateway) to a format usable by an origin server.

**MIME (Multipurpose Internet Mail Extension)**

A specification, defined by the Internet Engineering Task Force (IETF), for formatting non-ASCII messages so that they can be sent over the Internet. Today, there are many pre-defined MIME types, which are supported by various browsers and mail clients.

**Modular DTD**

A DTD that references one or more external modules which contain required element, attribute, or entity definitions.

**MMS**

Multimedia Messaging Service. The capability within a WAP Client to receive and process multimedia message types such as those providing for sound and video.

**MSISDN (Mobile Subscriber Integrated Services Digital Network)**

A standard for sending voice or data over a cellular network.

**MTM (Message Type Module)**

A plug-in module containing a set of components that support a specific messaging protocol. Used within the extensible, multi-protocol messaging framework introduced in EPOC Release 5, the MTM handles all interaction with lower level communication protocols such as IP.

**Multipart Message**

A single message composed of an ordered list of messages (or parts), each of which has a set of HTTP headers and a content body. The parts may be of any content type. In WAP environments, the multipart format is used to enable the delivery of a set of related content parts to a device all at once in order to avoid multiple network

requests, thereby reducing latency and thus the user's perception of the application response time.

**Multipart User Agent (Mulitpart Handler)**

A handler to process s multipart message.

**Origin Server**

The server on which a given resource resides or is to be created, often referred to as a web server or an HTTP server.

**Push Access Protocol (PAP)**

A protocol for specifying control information and content that is to be pushed to a client via a Push Proxy Gateway (PPG).

**Push Framework**

A term describing the entire WAP push system and encompassing the protocols, service interfaces, and software entities used to push content to a WAP client.

**Push Initiator**

The origin server that submits content to the Push Framework for delivery to a user agent on a WAP client.

**Push OTA Protocol**

The Push "Over the Air" Protocol, which is the protocol used to convey push content from a Push Proxy Gateway to a WAP client.

**Push Proxy Gateway (PPG)**

A server responsible for accepting Push content from a Push Initiator and transmitting it to a WAP client.

**Push Session**

A WSP session that is capable of conducting Push operations.

**Reference Implementation**

Implementation of the Nokia WAP Client on a specific platform; Nokia provides a Windows NT reference implementation.

**Rendering**

Formatting and presenting information to the display of a mobile device.

**Resource**

A network data object or service that can be identified by a URL.

**Server**

A device (or application) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client.

**Service Indication (SI)**

One of the two content types used to develop a Push message.

**Service Loading (SL)**

One of the two content types used to develop a Push message.

**Short Message System (SMS)**

A protocol defined within the GSM standard that enables point-to-point transmission of short messages. A short message is 140 bytes (or 160 characters) in length and uses a special 7-bit character set; also, by means of concatenating messages, it can be used to transmit up to 31 KB of binary text or data.

**Synchronized Multimedia Integration Language (SMIL)**

An XML-based content type that may be included in MMS messages in order to specify the ordering, layout, and presentation of the included media parts.

**Standardized Generalized Markup Language (SGML)**

The Standardized Generalized Markup Language is a general-purpose language for domain-specific markup languages. SGML is defined in the *ISO8879* standard.

**Strictly Conforming Document**

A document that adheres both to the XML specification (and is therefore well-formed) and to its DTD (either contained within or referenced externally).

**Tag**

A command inserted in a document that specifies how the document or a portion of the document should be formatted. In XML-based languages, a tag is enclosed in a pair of angle brackets: < and >. Tags are generally used in pairs, one to start the element and one to end it.

**Terminal**

A device providing the user with user agent capabilities, including the ability to request and receive information. Also called a mobile terminal or mobile station.

**Transcode**

The act of converting from one character set to another, for example, conversion from UCS-2 to UTF-8.

**UAPROF (User Agent Profile)**

The UAPROF specification defines mechanisms for describing and transmitting information about the capabilities of client devices and the preferences of the users of such devices. It enables the end-to-end flow of a User Agent Profile (also referred to as Capability and Preference Information, or CPI) between the WAP Client, intermediate network points, and the Origin Server.

**Unicode**

An encoding scheme for written characters and text. Unlike ASCII, which uses 7 bits for each character, Unicode uses 16 bits, which means that it can represent more than 65,000 unique characters, a huge increase over ASCII's code capacity of 128 characters. Unicode was authored and is maintained by the Unicode Consortium, a group comprised of major corporations and institutions involved in international computing. The character repertory and the codes assigned in Unicode are identical to those specified by *ISO 10646*, the international Universal Character Set (UCS) standard.

The Unicode Standard, Version 2.0 defines codes for characters used in every major language written today. In all, the Unicode standard currently defines codes for nearly 39,000 characters from the world's alphabetic, ideographic and syllabic scripts and symbol collections. The Unicode repertory was derived from many pre-existing character set standards to which previously unstandardized characters have been added. In particular, the first 256 code values are identical to those of ISO 8859-1 extended to 16 bits. Unicode values are displayed as four hex digits preceded by U+. For example, U+0041 is Latin upper case A.

**URI (Uniform Resource Identifier)**

The generic name for all types of names or addresses that refer to objects on the Internet. A URI can refer to an Uniform Resource Locator (URL) or an Uniform Resource Name (URN).

**URL (Uniform Resource Locator)**

A URL specifies the address of a resource on the Web. The syntax of an URL consists of three elements: (1) the protocol to be used in fetching the resource, (2) the IP address or domain name where the resource is located, and (3) the name of the resource itself (for example, its file name).

**User**

A user is a person who interacts with a user agent to view, hear, or otherwise use a resource.

**User Agent (UA)**

A user agent is any software component running on a mobile device that interprets WAP content identified by content type such as WML, WMLScript, and so on.

**User Interface (UI)**

The presentation layer of a software/hardware system through which a user interacts with the system, for example, by entering or receiving data.

**VCAL**

vCalendar, a WAP content type representing an Electronic Calendaring and Scheduling Format.

**VCARD**

vCard, a WAP content type representing an Electronic Business Card.

**VM**

Virtual Machine, or bytecode interpreter.

**WAE (Wireless Application Environment)**

The Wireless Application Environment specifies a general-purpose application environment based fundamentally on World Wide Web technologies and philosophies. WAE specifies an environment that allows operators and service providers to build applications and services that can reach a wide variety of different platforms. WAE is part of the Wireless Application Protocol.

**WAP (Wireless Application Protocol)**

The Wireless Application Protocol specifies an application framework and network protocols for wireless devices such as mobile phones, pagers, and personal digital assistants (PDAs). The WAP specifications extend mobile networking technologies (such as digital data networking standards) and Internet technologies (such as XML, URLs, scripting, and various content formats).

**WBXML (Wireless Binary XML Content Format)**

The WBXML specification defines a compact binary representation of XML. It is designed to reduce the size of XML documents (such as WML cards) for more efficient transmission over a wireless network. The binary format encodes the parsed physical form of an XML document, that is, both the structure and content of document entitites within the document.

**WDP (Wireless Datagram Protocol)**

The WDP layer lies directly above the data-capable bearer services which are supported by the various network types. WDP provides a common interface to upper layer protocols such as the Security, Transaction, and Session layers, and thus enables these to function independent of the underlying wireless network.

**Web server**

The server on which a given resource resides or is to be created. Often referred to as an origin server or an HTTP server.

**Well-Formed Document**

A document that conforms to the XML specification but which may not conform to the rules of a DTD (either because the document does not reference a DTD or because its content does not conform to the rules of the DTD).

**Wireless Identity Module (WIM)**

A WIM is a tamper-resistant hardware module fitted to mobile client devices for the purpose of storing and processing user identification and authentication data such as

permanent private keys. It is used in WTLS and in application-level security functions.

**WML (Wireless Markup Language)**

The Wireless Markup Language is a markup language based on XML and is intended for use in specifying content and user interface for narrowband devices, including mobile phones and pagers.

**WMLScript**

A scripting language used to program a mobile device. WMLScript is an extended subset of the JavaScript™ scripting language.

**WPKI**

WAP Public Key Infrastructure Definition. This WAP specification describes the infrastructure and procedures that are required to establish and manage the trust relationships that make possible authentication between servers and clients. These include the functionality needed to manage CA Public Key Certificates used for WTLS Class 2 connections, Client Public Key Certificates used for WTLS Class 3 connections, and Client Public Key Certificates used in conjunction with WMLScript signText.

**WPS**

Wireless Protocol Stack (includes session management and all the layers: WPS, WTP, WTLS and WDP)

**WSP (Wireless Session Protocol)**

The Wireless Session Protocol provides the upper-level application layer of WAP with a consistent interface for two session services. The first is a connection-oriented service that operates above the transaction layer protocol, and the second is a connectionless service that operates above a secure or non-secure datagram transport service.

**WTA (Wireless Telephony Application)**

WTA specifies a framework for supporting wireless telephony applications: these are applications which interface between an in-device WTA user agent and the network telephony infrastructure.

**WTAI (Wireless Telephony Application Interface)**

WTAI provides an interface consisting of library functions that are used by WTA user agents to conduct telephony sessions with the telephony network. These functions enable such functions as call and resource control, and they are available through both WML and WMLScript execution paths.

**WTLS (Wireless Transport Layer Security)**

WTLS is a security level protocol operating above the transport layer protocol (WDP) and below the session layer protocol (WSP). WTLS specifies an interface for estab-

lishing a secure connection between client (mobile device) and server (WAP Gateway).

**WTP (Wireless Transaction Protocol)**

WTP is a "light weight" transaction-oriented protocol that is suitable for implementation on "thin clients," that is, mobile devices. It operates efficiently over wireless datagram networks and provides the services necessary for the "request/response" interchange between a client and a server, which is termed a "transaction." It is message oriented and especially suited to supporting "browsing," which is a typical request/response interchange.

**XML (Extensible Markup Language)**

A World Wide Web Consortium (W3C) standard for Internet markup languages, of which WML is one such language. XML is a restricted subset of SGML.


NOKIA


NOKIA