Omnis für Unicode

Ein Weißbuch

TigerLogic

Januar 2009

Inhalt

Omnis für Unicode

Einführung: Unicode oder Nicht-Unicode?

Wussten Sie schon, dass das nächste größere Release von Omnis Studio nur noch in Unicode sein wird? Zur Zeit, in Omnis Studio 4.3.x, liefern wir sowohl eine Unicode- als auch eine Nicht-Unicode-Version von Omnis, und jeder Entwickler kann entscheiden, welche Version er verwenden möchte. Die Unicode-Version von Omnis kam zusammen mit Studio 4.1 im Jahr 2005 heraus und wird bereits von zahlreichen Entwicklern verwendet, die Daten mit einem erweiterten Zeichensatz speichern und verarbeiten müssen, meist um andere Sprachen zu unterstützen. Die Umstellung auf eine ausschließliche Unicode-Version der Omnis.exe bedeutet, dass die volle Bandbreite erweiterter und fremdsprachlicher Zeichensätze unterstützt wird, die in modernen Betriebssystemen und im Internet verfügbar sind. Nichts anderes wird sich ändern. Bei dieser Umstellung handelt es sich um eine notwendige Änderung, um moderne Plattformen und dazugehörige Softwareprodukte wie Server-Datenbanken und Programmiersprachen anderer Anbieter zu unterstützen, von denen wir viele in Omnis integrieren können oder unterstützen.

Das vorliegende Weißbuch fasst alle Aspekte hinsichtlich der Umstellung auf ausschließlich Unicode zusammen und beschreibt wichtige technische Details - keines davon stellt eine größere Hürde dar, im Gegenteil wird Unicode eine ganze Reihe von Vorteilen bringen. Zum Beispiel wird es für Ihre Omnis-Applikationen die Möglichkeit bedeuten, in neue internationale Märkte vorzustoßen, die eine Speicherung von Daten in erweiterten Zeichensätzen erfordern. Selbst wenn Unicode für Ihren Markt nicht erforderlich sein sollte, wird die Tatsache, dass Omnis auf ausschließlich Unicode umstellt, auch Ihnen Vorteile bringen, denn damit wissen Sie über Unicode besser Bescheid und kennen die damit verbundenen Fragen.

Entwickler, die schon länger mit Omnis arbeiten, seien darauf hingewiesen, dass viele der in diesem Weißbuch enthaltenen Informationen bereits früher publiziert worden sind. Wir haben sie hier für Sie zusammengefasst und aktualisiert.

Was ist Unicode?

Das Unicode Consortium (www.unicode.org), das den Unicode Standard pflegt, liefert folgende Defintion:

> *“Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.” –*
>
> *(Unicode bietet eine eindeutige Zahl für jedes Zeichen, unabhängig von der Plattform, unabhängig vom Programm, unabhängig von der Sprache.)*

Eine ausführlichere Erläuterung durch das Unicode Consortium finden Sie auf:
http://www.unicode.org/standard/WhatIsUnicode.html

Und noch ein weiteres Zitat des Unicode Consortiums:

> *“Im Grunde arbeiten Computer nur mit Zahlen. Sie speichern Buchstaben und andere Zeichen, indem sie jedem eine Zahl zuordnen. Bevor Unicode erfunden wurde, gab es hunderte unterschiedlicher Codierungssysteme, um diese Zahlen zuzuordnen. Kein einzelnes Codierungssystem konnte ausreichend viele Zeichen enthalten... Die Aufnahme von Unicode in Client-Server- oder mehrschichtige Applikationen und Web-Seiten bringt erhebliche Kosteneinsparungen verglichen mit der Verwendung überkommener Zeichensätze. Unicode ermöglicht es, dass ein einzelnes Softwareprodukt oder eine einzelne Web-Seite über verschiedene Plattformen, Sprachen und Länder verwendet werden kann, ohne neu- oder umprogrammiert*

werden zu müssen. Damit können Daten ohne Korruptionsgefahr durch viele unterschiedliche Systeme transportiert werden.." (www.unicode.org, Juli 2008)

Und deshalb ist es wichtig, dass Omnis und – noch viel wichtiger – *Ihre Applikationen* zukünftig Unicode-Daten unterstützen. Unter kommerziellen Gesichtspunkten sollten Sie in Ihren Applikationen eine Unicode-Unterstützung bereitstellen, um zumindest Ihre gegenwärtige Marktposition *zu wahren* (andernfalls könnten Sie gegenüber Ihren Mitbewerbern an Boden verlieren) und um *neue Kunden zu gewinnen* in neuen und sich entwickelnden Märkten, die die Unterstützung von Daten mit fremdsprachlichen und erweiterten Zeichen erfordern.

Das Unicode Consortium bietet Informationen und Ressourcen zum Thema Unicode einschließlich der Standarddefinition, Wartung, Zeichentabellen, ein Repository mit lokalen Indentifizierungen sowie eine Liste mit Unicode-fähigen Produkten.

Die neueste Version von Unicode ist Version 5.1, die mehr als 100.000 verschiedene Zeichen, die in vielen verschiedenen Sprachen auf der ganzen Welt verwendet werden, darstellen kann. Viele Betriebssysteme und Softwareprodukte unterstützen Unicode standardmäßig; Unicode ist mittlerweile allgemein als Standard für die Darstellung von Zeichen akzeptiert. So bieten beispielsweise Microsoft Windows NT, 2000, XP und Vista sowie die aktuellen Mac OS X Versionen Unicode-Unterstützung. Auch alle neuen Web-Standards wie z. B. die aktuellen Versionen von HTML und XML unterstützen Unicode, ebenso wie die neuesten Versionen des Internet Explorers und alle Mozilla-basierten Browser. Und auch die aktuellen Versionen von Sybase, Oracle und DB2 bieten Unicode-Unterstützung.

Unicode-Unterstützung in Omnis

Omnis Studio 4.1 war die erste Omnis-Version, die Unicode unterstützte. Als sie herauskam, gab es eine gesonderte Unicode-kompatible Version von Omnis Studio neben der Nicht-Unicode-Version von Omnis. Zur Zeit ist die Unicode-Version noch nur für Windows und Mac OS X erhältlich. Die gegenwärtige Version Omnis Studio 4.3.1 bietet ebenfalls eine Unicode- und eine Nicht-Unicode-Version von Omnis, aber ab dem nächsten größeren Release von Omnis werden wir nur noch die Unicode-Version von Omnis Studio ausliefern und selbstverständlich auch eine Linux-Version zu den bereits vorhandenen Unicode-Versionen für Windows und Mac OS X hinzufügen.

Neben der Darstellung von Textelementen und Character-Daten in unterschiedlichen Sprachen in den Omnis-Entwicklungs- und –Runtime-Umgebungen betrifft die Verwendung der Unicode-Codierung hauptsächlich die Sortierreihenfolge dynamischer Daten, zum Beispiel in Listen, sowie das Abfragen und Aufrufen von Daten aus Unicode-kompatiblen Server-Datenbanken und der Omnis-Datenbank selbst.

Zeichen-Normalisierung

Ursprünglich war Unicode ein 16-bit Zeichensatz. In der Folgezeit wurde es laufend erweitert und umfasst heute Unicode-Zeichen, auch „Code Points“ genannt, bis einschließlich U+10FFFF. Es ist nicht davon auszugehen, dass es noch mehr erweitert wird. Windows und Mac OS X stellen nach wie vor Unicode-Textelemente mit Hilfe von Arrays von Short (16-bit) Integers dar. Das ist auch kein Problem, da der UTF-16 Standard es erlaubt, Unicode-Zeichen U+10000 und größer als Paare von 16-bit Werten (jeder Teil des Paares nimmt den Platz im 16-bit Bereich ein, der nicht von Code Points beansprucht wird) darzustellen. Diese Wiedergabe wird als „*surrogate pair“* bezeichnet.

Intern verwendet Omnis UTF-32 zur Darstellung von Code Points, das heißt jedes Unicode-Zeichen nimmt 32 bits ein, und der Wert jedes Code Points liegt zwischen 0 und U+10FFFF einschließlich. Dieses ermöglicht eine direkte Übergabe von Textelementen, da jedes Unicode-Zeichen im Speicher den gleichen Platz einnimmt.

Unicode erlaubt, dass eine erhebliche Zahl von Zeichen durch mehr als eine Folge von Code Points wiedergegeben werden. Ein Beispiel dafür ist der Buchstabe E mit einem Accent circumflex darüber und einem Punkt darunter (Ệ), ein Zeichen im Vietnamesischen. Dieses Zeichen hat fünf mögliche Darstellungsweisen in Unicode:

1. U+0045 lateinischer Großbuchstabe E
   U+0302 kombiniert mit einem Accent circumflex
   U+0323 kombiniert mit einem Unterpunkt
2. U+0045 lateinischer Großbuchstabe E
   U+0323 kombiniert mit einem Unterpunkt
   U+0302 kombiniert mit einem Accent circumflex
3. U+00CA lateinischer Großbuchstabe E mit einem Accent circumflex
   U+0323 kombiniert mit einem Unterpunkt
4. U+1EB8 lateinischer Großbuchstabe E mit einem Unterpunkt
   U+0302 kombiniert mit einem Accent circumflex
5. U+1EC6 lateinischer Großbuchstabe E mit einem Accent circumfl und einem Unterpunkt

Ein Zeichen, das durch mehr als ein Unicode-Zeichen repräsentiert wird, wird *composite character* genannt, während ein Zeichen, das nur durch ein Code Point repräsentiert wird, als *pre-composed character* bezeichnet wird.

Was den Endanwender betrifft, so werden normalerweise alle diese Zeichenrepräsentierungen gleich behandelt. Das führt zu einigen interessanten Konsequenzen für Omnis, die in den folgenden Kapiteln beschrieben werden. Bitte beachten Sie dabei, dass der Begriff *Anwender-Zeichen (end-user character)* das Zeichen bezeichnet, mit dem der Anwender arbeitet – im obigen Beispiel ist das Anwender-Zeichen also das Ệ.

Die Normalisierung eines Unicode-Textelements konvertiert dieses Element in ein definiertes Standardformat. Einmal normalisiert, hat das Unicode-Textelement nur eine mögliche Repräsentation, wodurch es möglich wird, es mit anderen Textelementen zu vergleichen und daraus für den Anwender nützliche Ergebnisse zu erzielen. Der Unicode-Standard empfiehlt zwei Formen der Normalisierung, nämlich:

- **Kanonische Dekomposition, bezeichnet als NFD:**
  Pre-composed Characters werden durch ihre äquivalenten Composite Characters ersetzt; Composite Characters werden ersetzt durch eine einzige feste Composite-Repräsentierung.
- **Kanonische Dekomposition gefolgt von kanonischer Komposition, bezeichnet als NFC:**
  Nach der Durchführung der NFD werden alle Composite Characters durch ihr Pre-composed Äquivalent ersetzt, sofern ein solches existiert.

Omnis bietet zwei Funktionen, um Zeichenfolgen zu normalisieren:

- nfd(string)
  führt für die Zeichenfolge eine kanonische Dekomposition aus und gibt den normalisierten String zurück.
- nfc(string)
  führt für die Zeichenfolge eine kanonische Dekomposition und anschließend eine kanonische Komposition durch und gibt den normalisierten String zurück.

Diese Funktionen sind **nicht** in Client-seitigen Web Client-Methoden verfügbar.

Textvergleich

Omnis verwendet zwei Arten von Vergleichen für Textelemente:

- Vergleich der UTF-8 Werte des Strings. Dieses wird als “Character Comparison” bezeichnet.
- Vergleich entsprechend der für den Anwendungsbereich geltenden Regeln, wie sie in der Lokalisierungs-Datendatei spezifiziert sind; vor dem Vergleich werden die Eingabedaten normalisiert. Dieses wird als “National Comparison” bezeichnet. National Comparison wird meistens eher die Resultate erzielen, die der Anwender erwartet. Beachten Sie hierbei, dass Großbuchstaben in Verbindung mit “National Comparison” unter Umständen keinen Effekt haben, da manchmal die Regeln der nationalen Sortierreihenfolge die Groß- und Kleinschreibung ignorieren.

Die natcmp() Function verwendet “National Comparison”. Beachten Sie bitte, dass die Funktion natcmp() in Client-seitigen Web Client-Methoden **nicht** verfügbar ist.

Omnis vergleicht Text aus vielen verschiedenen Gründen und an vielen unterschiedlichen Stellen. Schlüsselbereiche dafür sind:

- Sortieren von Listen
- Suchen in Listen
- Manipulieren von Datendatei-Indizes
- Expressions, zum Beispiel zum Testen eines If-Statements

Omnis unterstützt zwei Arten von Character Variablen – Character und National

Sortieren von Listen

Wenn der Typ Character benutzt wird, verwendet Omnis „Character Comparison“.

Wenn der Typ National benutzt wird, verwendet Omnis „National Comparison“.

Suchen in Listen

Wenn der Typ Character benutzt wird, verwendet Omnis „Character Comparison“.

Bei Suchabfragen, die direkt eine Zeichenspalte vom Typ National benutzen, verwendet Omnis „National Comparison“.

Andere Suchen, zum Beispiel solche, die eine Berechnung enthalten, verhalten sich so als wenn sie mit normalen Zeichen-Daten arbeiten. Sie können aber natcmp() als Teil der Berechnung verwenden, damit „National Comparison“ benutzt wird.

Manipulieren von Datendatei-Indizes

Indizes für Felder vom Typ National verwenden “National Comparison“.

Expressions

Um das korrekte Verhalten von Expressions, die den Wert von Character-Variablen testen, sicherzustellen, müssen Sie entweder zuerst ihren Wert mit Hilfe von nfc() oder nfd() normalisieren, oder Sie müssen die Funktion natcmp() verwenden.

Zeichnen von Text

Je nachdem, welchen Font und welches Betriebssystem Sie verwenden, werden verschiedene Repräsentationen desselben Anwender-Zeichens nicht immer in derselben Weise gezeichnet. Das gleiche gilt für die Verwendung von Strings, die „surrogate pairs“ erfordern. Allgemein gesagt: Sie werden die besten Ergebnisse erzielen, wenn Sie den Text mit Hilfe von nfc() normalisieren, denn Probleme tauchen generell mit Composite Characters auf.

Eingeben von Text

Wir empfehlen, wo immer möglich die nfc() Normalisierung für Daten zu verwenden, die editiert werden sollen. Wenn in den Daten Composite Characters enthalten sind, kann ein mehrmaliges Drücken der linken oder rechten Cursortaste notwendig sein, um ein solches Zeichen zu überspringen; außerdem können beim Markieren und Auswählen eines Textbereichs Fehler auftreten, das heißt ein bestimmter Textbereich wird optisch hervorgehoben, aber wenn diese Zeichenfolge in die Zwischenablage kopiert wird, wird möglicherweise nicht exakt oder vollständig der zuvor hervorgehobene Text kopiert.

Omnis führt eine NFC-Normalisierung auf Character-Daten, die aus der Zwischenablage eingefügt wurden, durch, wenn es im Thick Client-Modus (Runtime) läuft; es wird keine Normalisierung durchgeführt, wenn Zeichen in ein Remote Form im Web Client eingefügt werden.

Übersetzen von Zeichen

Die folgenden Funktionen ermöglichen Ihnen die Übersetzung eines spezifischen Zeichens innerhalb eines Strings in seinen Unicode-Wert und umgekehrt:

- *unicode*(string,position[,returnhex])
  gibt den Unicode-Wert des Zeichens an der angegebenen Position innerhalb des Strings zurück. Die erste Position im String ist 1. Wenn Boolean returnhex true/wahr ist (Standardeinstellung ist false), wird ein Hex-String, der den Wert repräsentiert, in der Form 'U+h' zurückgegeben.
- *unichr*(num1[,num2]...)
  gibt einen String zurück, der durch die Verknüpfung der übergebenen Unicode-Zeichencodes gebildet wird. Jeder Code ist entweder eine Zahl oder ein String der Form 'U+h', wobei h 1-6 Zeichen, die einen hexadezimalen Wert repräsentieren, ist.

Diese Funktionen sind in Client-seitigen Methoden ebenso wie im Thick Client verfügbar. (Solche Funktionen generieren eine Fehlermeldung, wenn sie in der Non-Unicode-Version von Omnis verwendet werden, wodurch alle Ihre Libraries, die in der Unicode-Version von Omnis entwickelt wurden, imkompatibel werden mit der Non-Unicode-Version von Omnis.)

Locale Identifier

Die Funktion *locale()* gibt den Locale Identifier (LCID) für den aktuellen Rechner/das aktuelle Betriebssystem zurück. Neben der Sprache des Systems spezifiziert der Locale Identifier die Dezimal-, Tausender- und Listen-Separatoren, Währungswerte, Maßeinheiten, Datumsformate und die Reihenfolge des Zeichensatzes. Sie ist spezifiziert auf Betriebssystemebene und hat die Form Sprache_Land, wobei *Sprache* der Sprachname entsprechend ISO639 und *Land* der Landesname entsprechend ISO3166 ist. Beispielsweise ist die Locale für Großbritannien “en_GB”. Auf Mac OS X können sich zusätzliche Informationen, zum Beispiel Script-Code, zwischen der Sprache und dem Land befinden (das liegt daran, dass OS X ja ICU Locale verwendet).

Beachten Sie, dass *locale()* in Methoden, die im Web Client unter Mac OS X ausgeführt werden, nicht funktioniert.

Test auf Unicode-Version

Die Funktion *isunicode()* gibt ‘true’ zurück, wenn (und nur wenn) der aktuelle Code in der Unicode-Version von Omnis Studio ausgeführt wird. *isunicode()* funktioniert auch in Client-seitigen Methoden und zeigt an, ob der Web Client Unicode unterstützt.

Unicode Datenkonvertierung

Die Funktion *uniconv()* ermöglicht Ihnen die Übersetzung von Unicode Character-Daten eines Typs in einen anderen. Die Syntax ist dabei:

```
uniconv(srctype,src,dsttype,dst,bom,errtext)
```

konvertiert *src* und speichert das Ergebnis in *dst*. Im Erfolgsfall wird null zurückgegeben und bei Nicht-Erfolg ein ‚non-zero’ Fehler-Code mit dem Fehler-Text in *errtext*. *Src* und *dst* sind entweder binäre oder Character-Variablen, abhängig von den Werten von *srctype* und *dsttype*.

srctype und **dsttype** sind Konstanten vom Typ kUniType... (siehe unten).

Bom ist Boolean: Wenn true, hat **dst** einen Unicode Byte Order Marker (BOM), wenn es relevant für die Art des Ziels ist.

Die kUniType...-Konstanten sind folgende:

kUniTypeAuto: Die kodierende Quelle wird automatisch aus der Konvertierungsquelle ermittelt; mögliche Kodierungen werden aus den verbleibenden kUniType...-Konstanten identifiziert (nur für den Quelltyp erlaubt).

kUniTypeUTF8: Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF-8 kodiert wurden.

kUniTypeUTF16BE: : Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF-16BE kodiert wurden.

kUniTypeUTF16LE: : Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF-16LE kodiert wurden.

kUniTypeUTF16: : Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF-16LE kodiert wurden, wenn der Rechner Little Endien ist, oder mit UTF-16BE, wenn der Rechner Big Endian ist. Dieses ist nützlich, wenn man Cross Plattform-Code schreiben möchte, der mit dem Betriebssystem interagiert.

kUniTypeUTF32BE: : Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF32BE kodiert wurden.

kUniTypeUTF32LE: : Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF32LE kodiert wurden.

kUniTypeUTF32: Die Daten werden in einer Binärvariablen gespeichert und enthalten Unicode Character-Daten, die mit Hilfe von UTF-32LE kodiert wurden, wenn der Rechner Little Endien ist, oder mit UTF-32BE, wenn der Rechner Big Endian ist. Dieses ist nützlich, wenn man Cross Plattform-Code schreiben möchte, der mit dem Betriebssystem interagiert.

kUniTypeNativeCharacters: Die Daten werden in einer Binärvariablen gespeichert und enthalten eine Kette von Bytes, wobei jedes Byte ein Zeichen im Latin 1 Zeichensatz für die jeweilige Maschine ist (Ansi auf Windows, MacRoman auf dem Mac, ISO-8859-1 auf Unix)

kUniTypeCharacter: Die Daten werden in einer Character-Variablen gespeichert. Bitte beachten Sie: diese Konstante wurde seit dem letzten Unicode-Build verlegt, Sie müssen sie also in Ihrem Code neu eingeben.

kUniTypeAnsiThai, kUniTypeAnsiCentralEuropean, kUniTypeAnsiCyrillic, kUniTypeAnsiLatin1, kUniTypeAnsiGreek, kUniTypeAnsiTurkish, kUniTypeAnsiHebrew, kUniTypeAnsiArabic, kUniTypeAnsiBaltic und kUniTypeAnsiVietnamese: Die Daten werden in einer Binärvariablen gespeichert und enthalten Character-Daten, wobei jedes Byte mit Hilfe der identifizierten ANSI Code Page kodiert ist.

Formfile

Die Properties $filereadencoding und $filewriteencoding in der Formfile-Komponente, die in Studio 4.0 eingeführt wurden, wurden in Studio 4.1 (und höher) dahingehend verändert, dass die kFFEncoding… Konstanten abgelehnt und durch die kUniType… Konstanten ersetzt wurden, um die kodierende Datei zu identifizieren. Formfile verwendet nun alle kUniType... Konstanten außer kUniTypeCharacter für die Eigenschaft $filereadencoding und alle kUniType... Konstanten außer kUniTypeAuto und kUniTypeCharacter für die Eigenschaft $filewriteencoding.

Zusätzlich gibt es eine kUniTypeBinary Konstante, um Dateien zu identifizieren, die als Raw Binary Data behandelt werden sollten. Code, der die alten Konstanten verwendet, sollte weiterhin funktionieren.

Fileops

Die Fileops Komponente hat zwei neue Methoden, nämlich $readcharacter() und $writecharacter(), mit deren Hilfe Sie Unicode Character-Daten aus einer Datei lesen und in eine Datei schreiben können

- $readcharacter(encoding,variable)
  liest alle Daten aus einer Datei, die Character-Daten enthält, in eine Variable; Kodieren ist eine der oben aufgelisteten kUniType… Konstanten, die die Kodierung der Datei identifizieren.
- $writecharacter(encoding,variable)
  ersetzt die Inhalte der Datei durch die Character-Daten, die in der Variable gespeichert sind; Kodieren ist eine der oben aufgelisteten kUniType… Konstanten, die die Kodierung der Datei identifizieren.

Für $readcharacter kann die Kodierung als eine der kUniType... Konstanten mit Ausnahme von kUniTypeBinary und kUniTypeCharacter spezifiziert werden. Die kUniTypeAnsi... Konstanten werden nur in Unicode Studio unterstützt.

Für $writecharacter kann die Kodierung als eine der kUniType... Konstanten mit Ausnahme von kUniTypeAuto, kUniTypeBinary und kUniTypeCharacter spezifiziert werden. Die kUniTypeAnsi... Konstanten werden nur in Unicode Studio unterstützt.

Import/Export und Report File Kodierung

Sie können die Kodierung von Import-Textdateien, Export-Dateien und Berichtsdaten, die in Textdateien geschrieben werden, und den Port kontrollieren, indem Sie die Properties $importencoding und $exportencoding verwenden:

- $importencoding
  Die Kodierung, die für importierte Daten bei Importen über den Port oder wenn die importierte Datei kein Unicode BOM hat, verwendet wird. Alle kUniType... Konstanten mit Ausnahme von kUniTypeAuto, kUniTypeCharacter, kUniTypeBinary und den kUniTypeUTF32… Werten.
- $exportencoding
  Die Kodierung, die für den Export von Daten und das Drucken an den Port oder in eine Textdatei verwendet wird. Alle kUniType... Konstanten mit Ausnahme von kUniTypeAuto, kUniTypeCharacter und kUniTypeBinary.
- $exportbom
  Wenn ‘true’ und die $exportencoding Preference eine Unicode-Kodierung identifiziert, wird ein Unicode BOM am Beginn der Ausgabedatei ausgegeben.

Diese Properties sind in den Omnis Preferences ($root.$prefs) zu finden. Auf einem Multi-threaded Server gibt es einen separaten Wert für jede dieser Properties für jeden Thread.

Library-Konvertierung

Wenn Sie eine Omnis-Library in der Unicode-Version öffnen, werden Sie aufgefordert, diese Datei zu konvertieren. Dabei handelt es sich um eine volle Konvertierung einschließlich Konvertierung aller Zeichen >= 128. Wurde eine Library einmal in Unicode konvertiert, kann sie nicht mehr in der Non-Unicode-Version von Omnis geöffnet werden.

Versionskontrolle

Die Unicode-Version des Omnis VCS kann Libraries erstellen, die nur in der Unicode-Version von Omnis Studio laufen. Die Non-Unicode-Version des VCS kann Libraries erstellen, die in der Unicode- und der Non-Unicode-Version von Omnis Studio laufen.

Editieren von Feld- Scrollbars

Die Eigenschaft $righttoleft ist in einer vorangegangenen Version eingeführt worden, um bei der Texteingabe in Eingabefeldern von rechts nach links zu scrollen und damit beispielsweise die Dateneingabe auf arabischen Systemen zu unterstützen. Diese Eigenschaft ist dahingehend verbessert worden, dass in einem mehrzeiligen Feld der vertikale Rollbalken auf der linken Seite des Feldes angezeigt wird.

DAMs

Oracle8 DAM (nur Unicode)

Es gibt eine neue Session Property für das Oracle8 DAM (DAMORA8), die nur im Unicode DAM verfügbar ist.

- $nationaltonclob
  $nationaltonclob wird verwendet, um das Standard-Mapping von Omnis Character- und National-Zeichen zu ändern. Standardmäßig werden Omnis Character- und National-Felder mit einem Subtype größer als $maxvarchar2 zum NCLOB Datentyp zugeordnet. Wenn $nationaltonclob auf kTrue gesetzt ist, werden nur National-Felder mit einem Subtype >$maxvarchar2 als NCLOBs zugeordnet. Character-Felder mit Subtype >maxvarchar2 werden als Non-Unicode CLOBs zugeordnet. Character-Felder, die in dieser Weise zugeordnet werden, bieten die Gefahr eines Datenverlusts/-abschneidung, wenn diese Felder Unicode-Zeichen enthalten.

Omnis-Datendatei-Konvertierung

WARNUNG: Sie sollten unbedingt eine Sicherheitskopie Ihrer Omnis-Datendatei machen, bevor Sie diese in die Unicode-Version von Omnis Studio konvertieren.

Die Unicode-Version von Studio 4.3 (und 4.3.x) besitzt einen vollständigen Dateidatei-Converter, der die Daten in Ihrer Omnis-Datendatei konvertiert und die Indizes neu erstellt. In der Unicode-Version von Studio 4.1 und 4.2 wurden die Daten nicht konvertiert, nur die Indizes wurden herausgenommen und neu erstellt, und die Datei wurde als Unicode markiert.

Bei dieser kompletten Konvertierung in Studio 4.3 oder höher werden alle Daten, die als Character markiert sind, konvertiert. In Fällen, in denen Character-Daten binär gespeichert sind – zum Beispiel Text, der in einer Dokumentdatei gespeichert ist – findet *keine* Konvertierung statt.

Wenn Sie auf eine Datendatei in der Unicode-Version von Omnis Studio zugreifen, werden Sie gebeten zu bestätigen, dass Sie die Daten konvertieren wollen. Wenn Sie mit Yes antworten, erscheint ein Dialog, der Ihnen zwei verschiedene Konvertierungsarten anbietet:

- Quick
  Hierbei werden die Indizes herausgenommen und neu erstellt, aber die Daten werden nicht konvertiert (es handelt sich also um dieselbe Konvertierung wie in Studio 4.2). Dieses Verfahren ist akzeptabel für Dateien, die nur 7 bit-Daten enthalten: Omnis überprüft nicht, ob die Datei ausschließlich 7 bit-Daten enthält, es liegt also in Ihrer Verantwortung zu entscheiden, ob es sicher ist, diese Art des Konvertierungsprozesses anzuwenden.
- Full
  Hierbei handelt es sich um den neuen Konvertierungsprozess, bei dem eine vollständige Konvertierung der Character-basierten Daten stattfindet

Datendatei-Befehle

Die Befehle *Open data file* und *Prompt for data file* besitzen eine Option “Convert without user prompts”. Wenn diese Option ausgewählt ist, wird der neue Konvertierungs-Dialog nicht angezeigt. Es gibt eine neue Option für diese Befehle, nämlich “Quick Unicode conversion” oder “Full Unicode conversion”, die es Ihnen ermöglicht auszuwählen, welcher Level an Konvertierung durchgeführt werden soll.

Datendatei-Konvertierung

Die Unicode-Version von Omnis Studio 4.3 führt eine vollständige Konvertierung der Omnis-Datendatei in Unicode durch wie oben beschrieben. Angesichts der Tatsache, dass das nächste größere Release von Omnis Studio nur noch Unicode sein wird, empfehlen wir Ihnen, Ihre Omnis-Datendatei mit der Unicode-Version von Omnis Studio 4.3 (oder 4.3.x) in Unicode zu konvertieren und uns alle evtl. auftretenden Probleme so schnell wie möglich mitzuteilen, damit wir ggf.

vorhandene Probleme beseitigen können, bevor die Unicode-only Version von Omnis Studio herauskommt.

WARNUNG: Noch einmal raten wir Ihnen dringend, eine korrekte Sicherheitskopie Ihrer Omnis-Datendateien zu machen, bevor Sie diese in der Unicode-Version von Studio 4.3 öffnen und/oder konvertieren. Wir empfehlen, eine Kopie Ihrer Daten zu erstellen, damit die Unicode-Konvertierung in Studio 4.3 zu testen und Probleme, falls welche auftauchen, umgehend an uns zu melden.

Entwickler, die Unicode Studio 4.3 verwenden, sollten bitte die Ergebnisse der vollen Konvertierung sorgfältig prüfen und Ihr Backup der Non-Unicode-Datendatei nicht vernichten, bevor Sie sich davon überzeugt haben, dass die Daten erfolgreich konvertiert worden sind. Sie sollten eine Reihe von Regression-Tests in Ihrer Applikation und Ihren Daten durchführen – normalerweise sollte man dieses mit einer neuen Version von Studio tun, aber bei der Umstellung auf Unicode Omnis und der Konvertierung Ihrer Datendateien sollten Sie ganz besonders sorgfältig auf eventuelle Datendatei- und Index-Probleme achten.

Behandlung von Char- & Binary-Daten unter Unicode

Sie können in der Unicode-Version von Omnis Studio eine Character-Variable nicht mit einer Binary-Variablen verknüpfen. Die korrekte Methode dafür ist es, $readfile zu verwenden, um die Datei in eine Binary-Variable einzulesen und dann die Binary-Variable zu analysieren. Bei der Zuordnung von Character zu Binary und umgekehrt können möglicherweise Probleme entstehen, besonders in der Unicode-Version, und dieses sollte deshalb vermieden werden.

Lokalisierung

Bei der Lokalisierung handelt es sich um den Prozess, Ihre Applikation zum Einsatz bei Kunden bereitzustellen, die die Software in einer anderen Sprache benötigen. In der Praxis bedeutet das die Notwendigkeit, jeglichen Text, der für den Anwender sichtbar ist, in die entsprechenden Fremdsprachen zu übersetzen, aber es kann auch sein, dass Sie zusätzlich die Art, wie Daten verwaltet, sortiert oder angezeigt werden, für manche Regionen ändern müssen.

Um Ihre Applikation zu lokalisieren, müssen Sie bestimmte Dinge im Omnis-Programm selbst (die Runtime Omnis.exe) und alle Textfolgen und Etiketten in Ihrer Omnis-Library ändern. Wenn Sie Ihre Applikation über das Web einsetzen, brauchen Sie nur den Text in Ihrer Library oder, genauer gesagt, alles was die Anwender in ihrem Browser sehen können, zu ändern..

Omnis Studio bietet Ihnen verschiedene Werkzeuge, um das Omnis-Programm und die Library-Dateien zu übersetzen. Diese Werkzeuge werden detailliert in der Omnis Studio-Programmdokumentation beschrieben und werden hier nur kurz zusammengefasst:

Omnis-Programm-Lokalisierung

Bei der Lokalisierung des Omnis Runtime-Programms können Sie die folgenden Teile ändern; einige davon sind für den Anwender sichtbar, während andere beeinflussen, wie Buchstaben- und Zahlen-Daten verwaltet werden:

- Bezeichnungen der Wochentage und Monatsnamen,
- Separatoren-Zeichen,
- Text für Ja/Nein, OK/Abbrechen, True/False, Am/Pm und An/Aus,
- die nationale Sortierreihenfolge.

Alle Omnis-Libraries teilen sich dieselben Daten, die in einer Omnis-Datendatei mit der Bezeichnung omnisloc.df1 gespeichert sind, oder die Lokalisierungs-Datenbank, die sich im Local-Ordner unter dem Omnis-Hauptordner befindet. Mit Hilfe einer Omnis-Library mit der Bezeichnung omnisloc.lbr können Sie Sprachinformationen in der Lokalisierungs-Datenbank erstellen und editieren.

Die Lokalisierungs-Datenbank (omnisloc.df1) beinhaltet einen Daten-Slot für Konfigurationsdaten; jeder Datensatz in diesem Slot enthält einen kompletten Satz an Daten entsprechend der jeweiligen Sprache. Außerdem ist ein Daten-Slot mit einem einzelnen Datensatz enthalten, der die aktuelle Sprache, das heißt den aktuellen Satz an Konfigurationsdaten, identifiziert.

Die Lokalisierungs-Library (omnisloc.lbs) ist für die Unicode-Version von Omnis Studio aktualisiert worden. Das Sortierreihenfolge-Feld ist nun mit dem Etikett ‚Locale' versehen, wenn die Unicode-Version ausgeführt wird. Nur in der Unicode-Version gibt es außerdem eine neue Checkbox unterhalb des Locale-Feldes: "Use Locale For Defaulted Items". Diese bestimmt die Locale für Sprach-Teile, die leer gelassen wurden, so dass diese einen Default-Wert vom System erhalten:

- Wenn die Checkbox Use Locale angeklickt wurde, kommen die Default-Werte von der Locale, die im Sprach-Datensatz gespeichert ist.
- Wenn diese Checkbox nicht ausgewählt ist, kommen die Standardwerte aus der Default Locale des Betriebssystems.

Lokalisierung der Omnis-Library

Ihnen stehen zwei Möglichkeiten zur Übersetzung des Texts und der Etiketten in Ihrer Omnis-Applikation zur Verfügung:

- **Die Translation Library**
  Diese Omnis-Library (trans.lbs), die sich im Local-Ordner unterhalb des Omnis Hauptordners befindet, ermöglicht Ihnen die Übersetzung des Texts und der Etiketten in Ihrer gesamten Library; dabei exportieren Sie alle Textzeichen in Ihrer Omnis-Library, übersetzen sie und importieren sie anschließend zurück in Ihre Library.
- **String Tables**
  Wenn Sie Tabellen übersetzter Textstrings verwenden, können Sie den Text und die Etiketten in Ihrer Omnis-Applikation dynamisch ändern. Mit Hilfe des String Table Editors in Omnis können Sie Tabellen erstellen, die eine Matrix von Worten für eine beliebige Zahl von Sprachen enthalten, und dann die String Table-Funktionen verwenden, um den übersetzten Text wenn benötigt zu laden. Der String Table Editor befindet sich unter Tools>>Add Ons.

Wir würden gern von Ihnen hören

Wir würden gern von Ihnen hören, wenn Sie irgendwelche Fragen zur Umstellung auf Unicode in Omnis haben. Bitte senden Sie Ihre Fragen, Kommentare usw. per Email an: theeditor@omnis.net. Wir werden versuchen, alle Ihre Fragen zu beantworten und entsprechende Lösungen im nächsten größeren Release von Omnis Studio zu realisieren.

Und wenn Sie die Unicode-Version von Omnis Studio bereits benutzt haben, um eine Applikation zu entwickeln, die andere Sprachen als englisch und deutsch unterstützt, dann würden wir ebenfalls gern von Ihnen hören. Vielleicht ergibt sich daraus eine Möglichkeit, Sie in unserer Section ‚Success Stories' (Case Studies und Kundenreferenzen) auf unserer Web-Seite vorzustellen und zu unterstützen. Bitte benutzen Sie auch dafür die genannte Email-Adresse, um mit uns in Kontakt zu treten.

