

CAVEATS ON EARLY ODMJNI100 OPERATION

This is information on use of the early odmjni100 implementation releases:

1. Binding to ODMJNI and Using the info.odma.practical100 package
2. Choosing an Application ID.
3. Avoiding Text Characters outside of ISO 646 (7-bit ASCII/ISO).

[These are preliminary notes that will be blended into an update of the "Guide and Usage Scenarios" document at some point.]

[IMPORTANT: The restrictions on text characters (3) may be a problem working with a production DMS such as GroupWise in a European or other international setting. PLEASE WATCH FOR THIS.]

1. BINDING TO ODMJNI AND USING INFO.ODMA.PRACTICAL100.

1.1 Connection to the ODMJNI Implementation

Java-based ODMA-aware applications can connect to the ODMJNI implementation using code of this form:

```
import info.odma.practical100.*;
import info.odma.odmjni100.*;

OdmConnection MyConnection = OdmJniBind.application("MyAppId");
```

This is slightly different than how the current null implementation is chosen:

```
OdmConnection MyConnection = new OdmNullConnection("MyAppId");
```

But the use of the connection-interface reference (here, MyConnection) is the same from that point on.

1.2 Background on the Two Packages and the Approach

[Preliminary notes for an update to the "Guide and Usage Scenarios."]

The info.odma.practical100 package provides the interfaces that are always used in providing ODMA-aware operations in Java-based desktop applications.

info.odma.practical100 (currently at 0.05alpha version) will evolve to beta and then final versions. In addition to the interface, there will be some helper classes, including versions of the null implementations that are already included in 0.05alpha.

The implementations of these interfaces for full ODMA-aware operation are in the package info.odma.odmjni100. That package implements access to the ODMA Connection Manager and to ODMA-compliant DMS integrations

available to the Java-based application.

The odmjni100 package contains private material. There is only one operation available to the ODMA-aware Java-based application:

```
info.odma.practical100.OdmConnection  
    info.odma.odmjni100.OdmJniBind.application(String appId)
```

info.odma.odmjni100.OdmJniBind is a static class. It has one public method, the static application(String appId) operation.

This operation always returns an OdmConnection interface.

It is important to notice that there is no constructor for OdmJniBind. The class is simply used. Compare that with the use of the current info.odma.practical100.OdmNullConnection class:

```
OdmConnection MyConnection = new OdmNullConnection("MyAppId").
```

This OdmJniBind approach hides implementation details a little better. More importantly, it allows for installation of compatible upgrades without disturbing the ODMA-aware application and without requiring recompilation of the Java code for the application. It is a little less disruptive for introducing updates and fixes.

For harmony, we will add OdmNullConnection.application(String appId) later.

2. CHOOSING AN APPLICATION ID

Every ODMA-aware application has a distinct ODMA Application ID. The Application ID is a text string of at most 15 characters:

- * The Application ID is communicated to each DMS so that it may adjust its behavior based on knowledge of the application.
- * The Application ID is checked by the ODMA Connection Manager to determine the availability of a Default ODMA DMS for this application.
- * The Application ID is used as a key in the Windows Registry for information about treatment of the Application by the ODMA Connection Manager and by ODMA DMS integrations.
- * Different Application IDs should not differ only in case. Some uses may be case insensitive and this should be allowed for. Always use the same cased form of an Application ID so that case-sensitive uses will not lead to mismatching of IDs.

The registry at <<http://odma.info/faq/ODMAapps.htm>> lists known Application IDs. Not all Application IDs are known,. There is no obligation to register applications.

ODMJNI enforces the following restrictions on Application ID values:

- * The Application ID cannot be a null String and it must not

contain more than 15 characters.

* The Application ID is limited to the following Java characters:

digits '0' to '9' (codes '\u0030' to '\u0039')
letters 'A' to 'Z' (codes '\u0041' to '\u005a')
letters 'a' to 'z' (codes '\u0061' to '\u007a')

[IMPLEMENTER NOTE: Early versions do not provide full validation.
By the beta release, the implementation will be updated to throw
an unchecked exception if this format is violated.]

[COMPATIBILITY NOTE: There are some existing Application IDs with
space characters in them (e.g., "MS Word"). The current restriction
avoids problems of confusion in the creation of registry entries
and in communication of the IDs.]

The ODMJNI Test programs that behave as applications use the Application
ID "OdmNativeTest" and "MyAppId" appears in many examples.

Application IDs tend to distinguish desktop-application products. The
Application ID does not need to change when there are new releases of
the product unless there is some major change in how the documents
must be treated by the ODMA Connection Manager and ODMA-compliant DMS.

3. AVOIDING TEXT CHARACTERS OUTSIDE OF ISO 646 (7-bit ASCII/ISO)

Many Java String values are used in coordination between the Java
application and ODMA. Some of these strings are returned from
ODMJNI for use by the application. Other strings (such as Application
IDs and content-format identifiers such as ".doc") are provided by
the application.

All of the Java Strings are translated from or to single-byte character
strings as part of communicating with the ODMA Connection Manager and
with the ODMA DMS integrations via ODMA. No form of Unicode is used on
the ODMA side of these operations, and multibyte coding schemes (such as
shift-JIS) are also not supported. (The illustrative statement in the
ODMA 2.0 specification is incorrect on this subject.)

For all String values communicated between the application and methods
of the info.odma.practical100 interfaces, all characters must be chosen
from the printable characters of the Basic Latin character set (codes
'\u0020' to '\u007e'). These are the same as the printable 7-bit ASCII
codes.

[IMPORTANT NOTE: This means that additional characters, such as those of
the Latin-1 supplement and beyond are not available for use in
communication of ODMA-significant text. This also constrains the
behavior of DMS Integrations, which must not use any character codes
in the range '\0x80' to '\0xff' because there is no reliable way
to ensure their correct conversion to Unicode characters for use
in Java. This impacts the use of characters in

and file-system full path names for files

that are specified by a DMS and delivered to the application by ODMA.]

[IMPLEMENTER NOTE: There will be static-method filters for verifying the acceptability of java String values for particular uses by ODMA. When an unacceptable string is submitted to an ODMJNI operation, an unchecked exception will be thrown.

Early versions do not provide the static-method filters and full validation of input strings. By beta release, the filters will be in place and any exceptions to this restriction will be added.]

[IMPLICATIONS: If an existing DMS returns character codes outside of the allowed set, the operation will fail in alpha-level operations. The idea is not to deliver anything from a DMS that can't be converted and/or accepted back correctly.

If we find that there are additional characters actually used, we will need to implement a codepage-sensitive conversion from the returned characters to Unicode (and vice versa). This is not a perfectly reliable situation, but we will handle it if necessary to accomodate existing production DMS systems.

In addition, there are text values returned by a DMS that are not important to ODMA but can be important to the user's experience of the application (e.g., the title of a document, or the author's name). If those currently use additional characters, those characters will be translated incorrectly to Java String if they are translated at all.

These two situations are cause to accept expanded character sets returned in results from a DMS. But this situation may not be cured until after initial beta release.]

0.00 2006-11-15-08:30 Provide initial information and warnings about the limitations on character codes in the first implementations.

\$Header: /MyProjects/java/ODMdev/info/odma/odmjni100/odmjni100-caveats.txt 1
06-11-15 11:44 Orcmid \$

*** end of odmjni100-caveats.txt ***