

## **Abstract**

High-quality Chinese, Japanese, and Korean (CJK) typography requires an open-ended class of Chinese-derived characters. Different CJK fonts provide a core selection of these characters as part of their standard character set. However, publishers need supplemental glyphs or characters that are known as “gaiji.” This presentation describes a new Adobe initiative to address the gaiji requirement: the SING architecture. SING answers the need for a flexible gaiji workflow on the desktop. SING enables you to extend your CJK fonts with individual new OpenType-based “glyphlets,” representing variant glyph shapes or symbols. These glyphlets are embedded in documents and move through the workflow. Adobe intends to include SING in future Adobe products.

Building on work presented at the 22nd IUC in September 2002, this presentation reviews why gaiji are important; it describes the SING architecture; and it looks at some implications for the Unicode character-glyph model. We will demonstrate SING Technology Preview software. The presentation assumes basic knowledge of the Japanese, Chinese, or Korean writing systems, and of text formatting and the Unicode character-glyph model. Anyone entering, processing, or displaying text that contains person or place names in CJK languages, be it for publishing, for corporate databases, or for the web and cell phones, will encounter the gaiji requirement, and would benefit from being aware of the SING approach.

## **Author**

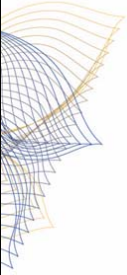
Jim DeLaHunt is an engineering manager at Adobe Systems, responsible for software related to Japanese font handling and to gaiji. He was engineering manager for the SING Gaiji Technology Preview. He was introduced the gaiji requirement when he first joined Adobe fifteen years ago, and still isn't satisfied with any gaiji mechanism he has found in the market.

**Jim DeLaHunt** ☎ +1-408-536-2690

✉ <delahunt@adobe.com>

✉ Core Technology Group, Adobe Systems Incorporated, M/S W-08, 345 Park Avenue, San Jose, CA 95110, USA

<<http://partners.adobe.com/asn/developer/type/gaiji.html>>




## Agenda

- **What are gaiji?**
  - Past gaiji-handling approaches
- **Adobe SING gaiji architecture**
- **Demo of SING gaiji technology preview**
- **Implications for Unicode**
- **Q&A**

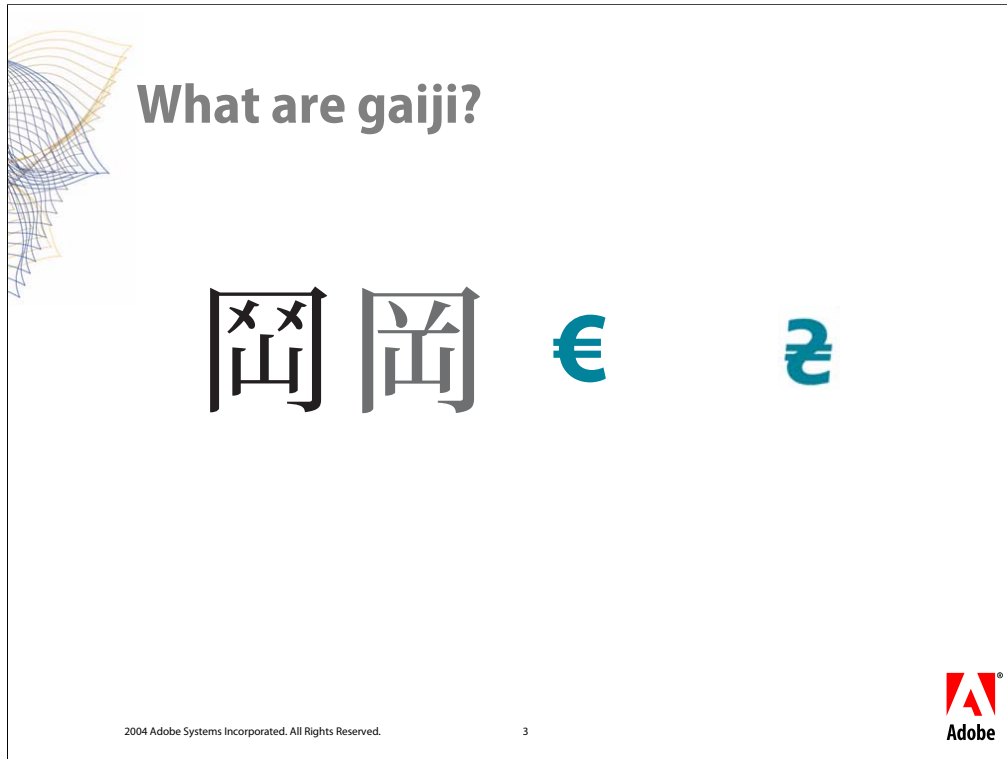
▪ Note: informal use of “character” and “glyph”

2004 Adobe Systems Incorporated. All Rights Reserved.

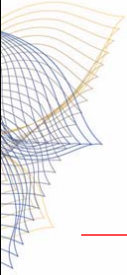


This paper begins with a brief definition of the term *gaiji*, and a historical review of how gaiji requirements were met by earlier generations of publishing technology. We then review Adobe's new SING gaiji architecture, which was designed to meet these requirements in a high-quality and high-productivity way. In the live presentation there will be a demonstration of the SING gaiji technology preview; in this paper there are some screen shots and brief explanations. We then move on to some interesting implications this work has for Unicode. There will be time for a Question and Answer (Q&A) period.

Note: in this talk, I reflect Japanese publishing industry practice, which is to not distinguish much between the concepts of “character” and “glyph”. Sharp eyes may notice me using one term rather informally, when the other might fit better with the Unicode character-glyph model.




In this section, we describe what *gaiji* are and where they are used.



## Gaiji, or supplemental characters

Supplemental glyphs or characters are known as “gaiji,” a term that refers to any glyph that’s valid in your written language but which is not in the font you are using.

2004 Adobe Systems Incorporated. All Rights Reserved. 4

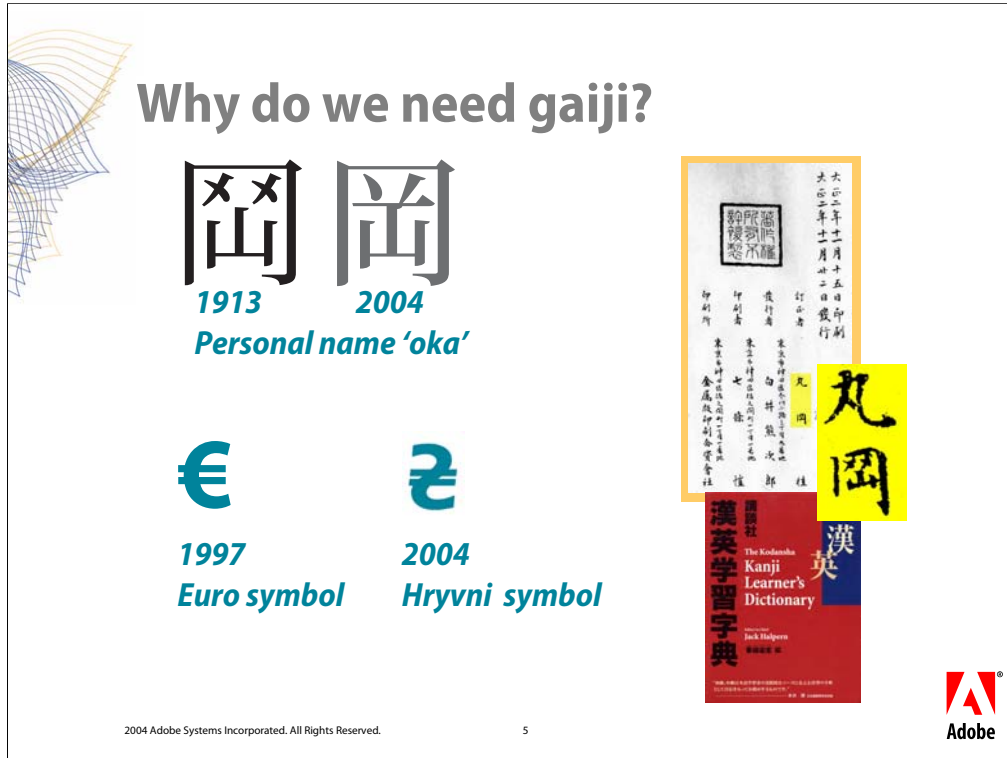


The term “gaiji” is a Japanese word meaning “outside character”. For the purpose of this paper, we define it as:

**Any character or glyph which is valid in your written language, but which is not in the font you are using.**

A good English translation for “gaiji” might be “supplemental character” or “supplemental glyph”.

Gaiji are particularly prominent in the CJKV ideograph script, *i.e.* the Chinese, Japanese, Korean, and Vietnamese languages.



**Why do we need gaiji?**

岡 岡  
1913 2004  
Personal name 'oka'

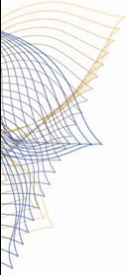
€ €  
1997 2004  
Euro symbol Hryvni symbol

2004 Adobe Systems Incorporated. All Rights Reserved. 5

Adobe

In the slide above, there are two variants of a character “oka”, used in personal Japanese names like “Maruoka”. The variant on the left is taken from a 1913 Japanese book, written by a person named “Maruoka”. The variant on the right is how that same character “oka” is written today. The 1913 variant is encoded in Unihan Extension B, but it is not available in typical fonts. It is a *gaiji*. Any publication that wanted to talk about the 1913 author “Maruoka”, using the characters they used, would have to find some way to reproduce the archaic character variant.


Western typography also has *gaiji*, on a much smaller scale. The Euro symbol was invented in the mid-1990s by the European Union. (The European Monetary Institute announced the symbol on 15. July 1997; the currency became official national currency on 1. January 1999, and Euro notes and coins started to circulate on 1. January 2002.) An upheaval resulted in the remainder of the 1990's, as the Euro character was encoded in Unicode, and fonts, keyboards, and printer drivers were updated to account for it. It was a rare event, but certainly not a unique one. On 3. March, 2004, the Ukrainian National Bank announced a symbol for the Hryvni, the Ukrainian currency. It is not clear how fast or fully the computer industry will support it, but it is clear that it is a *gaiji*, and some people will need a way to use it.



## Why do we need gaiji?

- **50,000 chars in dictionary, only 8-20,000 in Japanese font**
  - Historical variants, personal names, place names
  - Kanji writing system fundamentally open-ended
- **Printers want variant glyph designs which match font**
- **e-Government databases must sometimes store exact glyph**

2004 Adobe Systems Incorporated. All Rights Reserved. 6



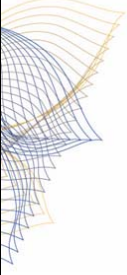
Gaiji are routine in the ideographic script used for Simplified Chinese, Traditional Chinese, Japanese, Korean, and Vietnamese (CJKV). This is because of the large and open-ended character repertoire for the script. Reference dictionaries for the Japanese language list about 50,000 characters. Standard Japanese personal computer fonts have only about 8,000 characters. Other fonts cover as many as 20,000 Han characters. Even such a comprehensive font as this still leaves 30,000 characters unrepresented. Any author wanting to use these characters would consider them gaiji.

But it's worse than that, because the CJKV ideographic script is fundamentally open-ended. Adding or removing a stroke, or changing one set of strokes for another, can be significant — either because it matches historical practice, or it changes the meaning of the character.

In CJKV ideographic scripts, historical forms are a rich source of gaiji. Countries periodically attempt to reform and simplify the writing system. Japan had a reform in the 1940's, and China in the 1950's. But the characters made obsolete by the reform are still important in a historical context, and so authors may wish to use them when writing about events or people or places of the pre-reform era.

Publishers and printing companies need gaiji support because they want a high-quality appearance for the gaiji in their printed text. This means that the glyph design for the gaiji must match the design of the parent font on which it is based. The gaiji must also be laid out in the text correctly, consistently with the rest of the text.


Government records in Japan are required to represent the exact characters for the personal names and birthplaces of people. As these records are computerised, the computer systems must be able to handle gaiji. Recently, several Asian governments have launched “e-government” initiatives that make this need more acute.



## Past gaiji-handling approaches

- **Optical typesetting (1960's)**
  - "Main plates" carry frequently-used characters
  - Other plates with special characters or symbols sold individually
- **Digital Typesetting Era (1970's-1980's)**
  - Character codes used as glyph ID codes
  - Gaiji in custom fonts or registered as gaiji codes
  - No portability of documents

2004 Adobe Systems Incorporated. All Rights Reserved. 7



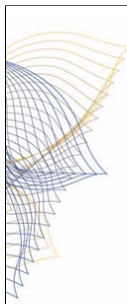
The gaiji requirement is not new. It is interesting to look at how past generations of publishing technology met this requirement. We focus on the Japanese publishing industry, but the same story applies elsewhere in East Asia.

Optical typesetting was a system where light was projected through glass plates carrying a mask shaped like a glyph, projecting onto photosensitive film. The result was the image of a glyph on the paper. A system of lenses and mechanism controlled the size and position of each glyph. Optical typesetting was adopted rapidly in Japan in the 1960's, together with offset printing.

Frequently-used characters provided on standard "main plates" from the system vendor. Other plates were sold individually, with needed variant glyphs or symbols. Customers placed special orders to the vendor for such plates. This made it fairly easy for customers to order gaiji characters from the font maker.

Next came the digital typesetting era, during the 1970's-1980's. Fonts were represented in software as curve outlines, in proprietary formats usable only on proprietary typesetting systems. Fonts, layout software, output devices, and gaiji handling was all supplied by the same vendor. On these systems, character codes were used glyph ID codes. (To this day, the publishing industry does not recognise the formal distinction made by the Unicode model between "character" and "glyph".)


Non-standard characters were put in custom fonts or registered as gaiji codes. These gaiji codes differed from installation to installation. As a result, documents which used custom fonts or gaiji were not portable to other systems. They could not be re-used, they had to be re-created on the other system.



## Gaiji in DTP, late 1980's – On

- **Ever-growing character collections and fonts**
  - (Adobe-Japan1-4 etc)
- **User makes Type 1 gaiji fonts**
  - Or special-orders fonts from font makers
- **Gaiji assigned special codes, as before**
  - When sharing documents with gaiji, high danger of dropped glyphs
- **No built-in support for gaiji**
  - Loss in quality and productivity

2004 Adobe Systems Incorporated. All Rights Reserved. 8



The Desktop Publishing (DTP) technology arrived in the Japanese market in the late 1980's, and in the other CKV markets in the following decade. In DTP technology, the mainstream of standard characters were made available through extending character collections. The Adobe-Japan1 series is a good example:

- Adobe-Japan1-1 represented the CID technology generation of font software
- Adobe-Japan1-3 represented the OpenType “Standard” generation of font software
- Adobe-Japan1-4 represented the OpenType “Pro” generation of font software
- Later, the series was extended to Adobe-Japan1-6.

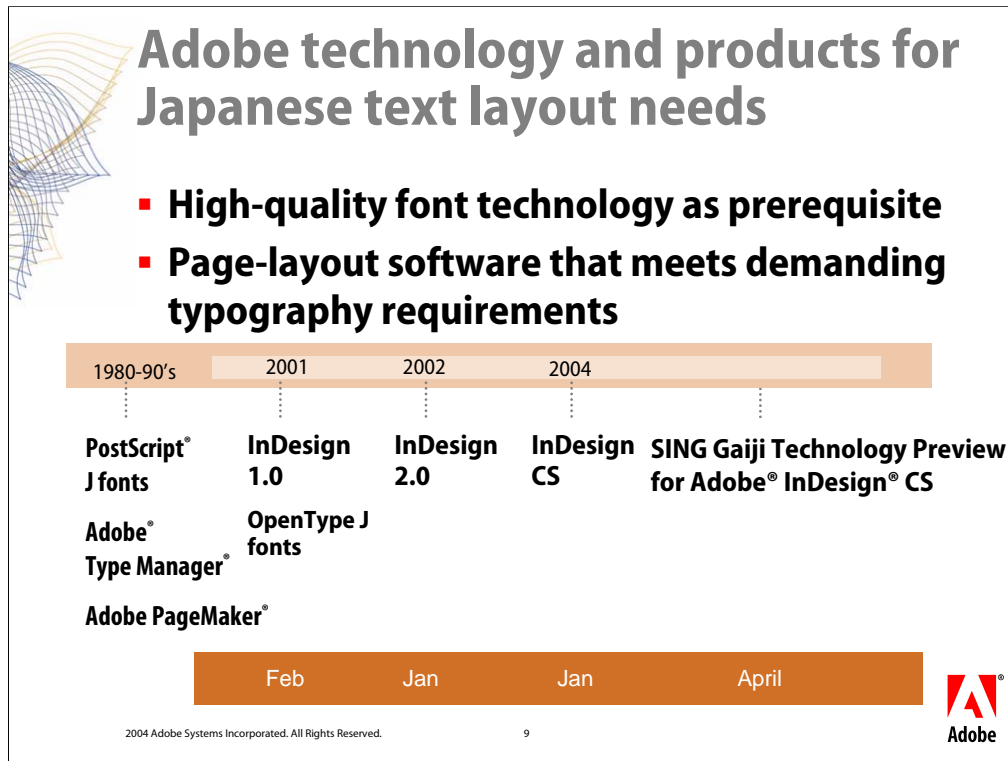
Each increment assigned Character Identifier (CID) codes to industry standard kanji, variant kanji (*itaiji*), and symbols. Similar character collections were defined for simplified Chinese, traditional Chinese, and Korean.

When users needed gaiji, they would produce Type 1 fonts containing the needed glyphs. In some circumstances they could order special purpose fonts from the parent font's maker. Gaiji were assigned special codes, as in the previous generation.

Because the gaiji were assigned special codes, it was difficult to share documents. There was a high danger of dropped glyphs. Special workflow steps needed to reduce this risk. The DTP production environments had no built-in support for operations involving gaiji such as text entry, editing, and layout. As a result, users suffered a loss in quality and productivity when attempting to use gaiji with DTP.

As a result, as of 2004 a major fraction of the Japanese printing industry still uses proprietary digital typesetting systems for text layout, especially when the text involves gaiji.





Adobe Systems has been a part of this history, starting with the advent of DTP in the Japanese (and later, Chinese and Korean) markets in the late 1980's.

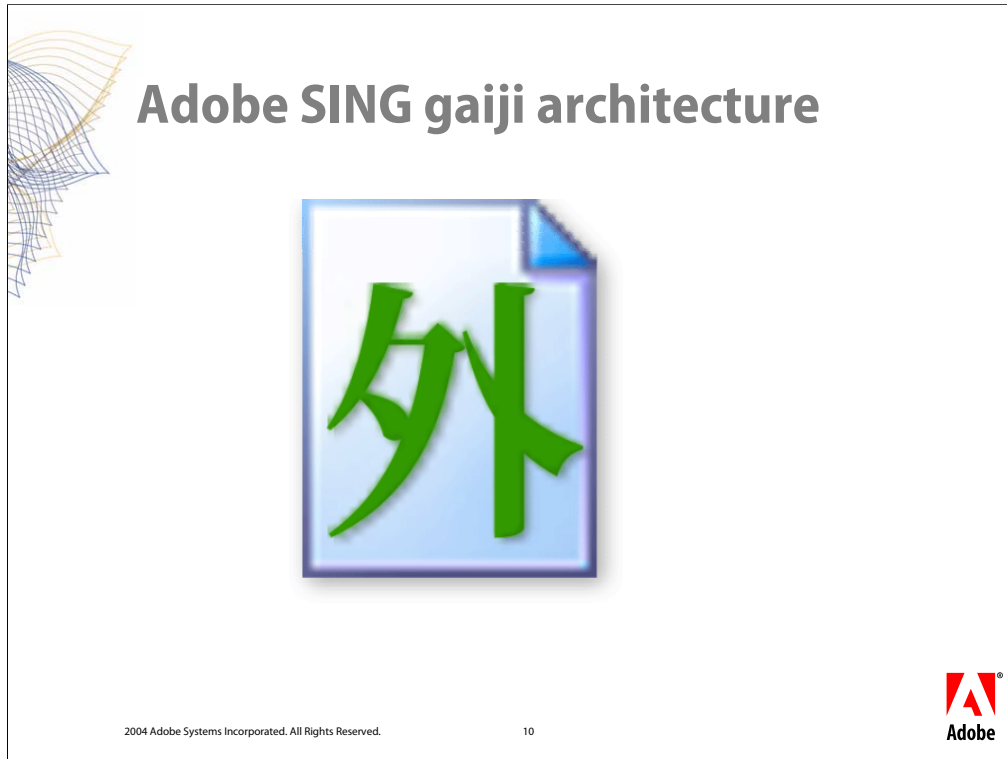
The overall technology trend was to put in place high-quality font technology, as a prerequisite to releasing page-layout software that used those fonts to meet demanding typography requirements.

Adobe produced the first Japanese fonts for use with PostScript® Japanese version output devices in 1988. Adobe® Type Manager® followed soon afterwards. Aldus Corporation (later merged with Adobe) released a Japanese version of PageMaker software in the same timeframe.

In 2001, OpenType Japanese fonts were released, along with Adobe InDesign® Japanese version layout software. InDesign J was designed with a typography model designed for CJK typesetting requirements, instead of one adapted from Latin script conventions of the North American and European markets. Updated versions of InDesign were released in 2002 and 2004.

In April, 2004, the SING Gaiji Technology Preview for Adobe InDesign CS was launched in Japan. We will discuss this product later in the paper.

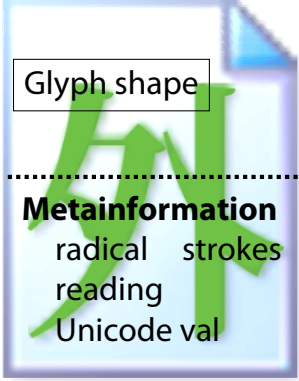
The overall message of this history is that Adobe Systems is not new to the issue of gaiji or to the requirements of CJK typography. The SING architecture we will discuss arises out of a long familiarity with the market, and several previous generations of experience meeting that market's needs.



Adobe has come up with a new cross-platform, general-purpose architecture to address the gaiji requirement in professional publishing and enterprise systems. It is known as the **SING gaiji architecture**. We will examine the architecture, its future course, and the ways in which it meets gaiji requirements.


## Overview of the SING gaiji architecture

- **SING: Smart INdependent Glyphlets**
  - Self-contained package of glyph shape and meta-information
  - Travels embedded in documents throughout workflow
- **“The OpenType approach to gaiji”**



1-2 KB

2004 Adobe Systems Incorporated. All Rights Reserved. 11



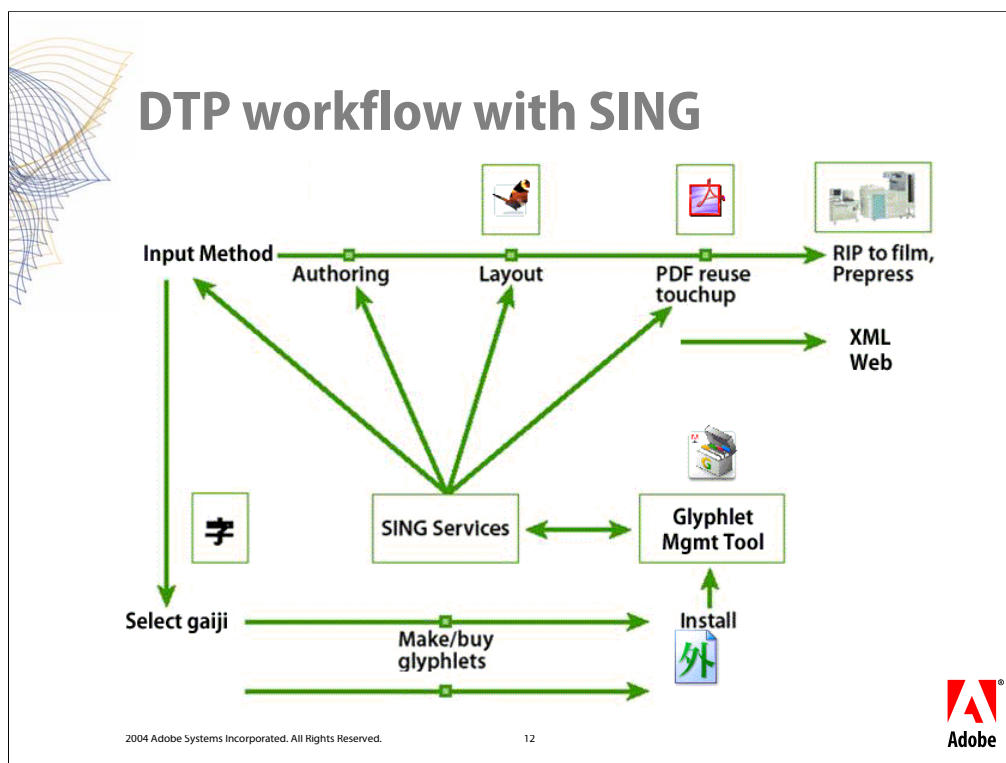
SING stands for “Smart INdependent Glyphlets”. Glyphlets are the centrepiece of the SING architecture.

A glyphlet is like a very small OpenType font that contains one glyph. It contains the glyph outline data for one glyph (plus a one or two alternate glyphs for different writing directions, if appropriate). This outline data is in the TrueType or CFF formats supported by OpenType. The glyphlet also contains meta-information, data that describes the character and glyph properties of the glyphlet. It omits several OpenType tables, so that an OpenType system will not accidentally interpret a glyphlet as an OpenType font.

A glyphlet is a small platform-independent file, only 1-2KB in size. When a glyphlet is used in a document, it is always embedded in that document, and travels with that document throughout the workflow. This means that whenever a document needs to be edited, laid out, printed, or converted to a different format, the glyphlet and all its character and glyph properties are available to support those operations. We escape the problem of having to install the glyphlet at multiple points in the workflow. Because the glyphlet is incorporated by value into the document, there is no need for it to have a global reference like a character code. SING does away with the requirement for globally known gaiji codes.

The metadata contains character properties like Unicode character code, and the reading of the character. It also contains glyph properties such as radical, stroke count, applicable Character ID (CID) values, and the font design which the glyphlet matches. A glyphlet can define itself as implementing an OpenType feature for the parent font. For example, it can supply a Printing Standard Glyph (or “NLC” glyph) for the ‘nlck’ feature.

By building on the OpenType specification and conceptual model, and on Unicode, SING represents an extension of the OpenType format, rather than a new font format. We believe it represents the “OpenType approach to gaiji”.



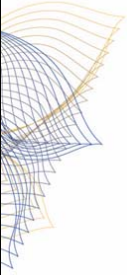
The diagram above shows the full SING architecture towards which Adobe is building. SING touches components throughout the workflow, and provides opportunities for many vendors, not just Adobe.

When a user selects a particular character for their document, the first step is to make that glyph available. Users can purchase the glyphlet from their font vendors. Since glyphlets are compact, standardised files, they are ideal for electronic distribution, and represent a new business opportunity for font makers. Or users can create a glyphlet using end-user oriented glyphlet creation tools. We anticipate opportunities for both very simple glyphlet creation tools, and more powerful tools based on font editors.

Once the glyphlet is bought or made, the user installs it via a **Glyphlet Management Tool**. This utility handles installation and removal of glyphlets, and lets users manage the collection of glyphlets installed on their system. The collection of glyphlets is under the control of a system-wide, shared SING Services Library. Other components in the system call this library for SING-related operations.

When a user enters text, the SING-enabled Input Method (IM) displays, in its list of candidate glyphs, glyphlets installed on the system as well as unencoded glyphs from OpenType fonts. The user can select the precise glyph variation they prefer at this stage, rather than waiting until the layout stage. Whether a glyph comes from SING or an OpenType font is transparent to the user. The authoring application maintains these glyph choices during the writing and text-editing process.


When the writer turns the article copy over to the layout step, all glyph choices and all necessary SING glyphlets are embedded in the article file. The layout station uses the embedded glyphlets to make high-quality layout, with no missing glyphs. It saves a final-form PDF file, which also embeds all the glyphlets (perhaps reduced to “dumb glyphs” with no gaiji properties). This PDF file can undergo all prepress operations. Thanks to the metadata, the laid-out text can also be exported as content marked up with XML or web languages.



## SING Glyphlet Format

- **Extension of OpenType format**
- **'glyf' or 'CFF' glyph outline tables**
  - Both TrueType and PostScript heritage supported
- **'cmap', 'BASE', 'GSUB' tables as in OpenType**
- **No 'name', 'head', 'OS/2', 'post' tables**
- **New SING tables**
  - 'SING': font-level data such as embedding info
  - 'META': tagged format for various metadata

2004 Adobe Systems Incorporated. All Rights Reserved. 13



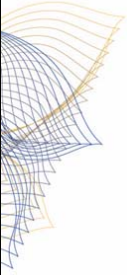
The SING format is based on OpenType architecture and conceptual model, which in turn relies on Unicode. SING is an extension to OpenType, rather than a new font format. Here is more detail about the glyphlet format.

The glyphlet contain a 'glyf' or a 'CFF' table, giving the glyph outline(s) in TrueType or CFF format respectively. These tables are identical to their OpenType counterparts, meaning it is easy to extend an OpenType rasteriser to support glyphlets. The SING architecture is agnostic about outline format.

The SING format calls for 'cmap', 'BASE', and 'GSUB' tables, with the same format and performing the same function as in OpenType. These tables refer to glyphs in the glyphlet or in the parent font. The SING format rules out the use of 'name', 'head', 'OS/2', and 'post' tables. These tables are omitted partly to be sure that no OpenType application mistakes a glyphlet for an OpenType font and displays it in a font menu – glyphlets are not to appear in font menus. Also, these tables contain information that is more relevant to full fonts than to glyphlets.

The SING specification adds two new OpenType tables. The 'SING' table contains information about the glyphlet as a font-like structure. Some of these values correspond to values from the 'head' or 'OS/2' tables in OpenType fonts. For instance, a **SING.embeddingInfo** field is defined as a subset of the values of an OpenType **OS/2.fsType** field.

The META table contains a series of tag-value pairs. The tags are integers which denote different classes of metadata, such as stroke count, parent font name, etc. The values can be various data types, typically UTF-8 strings. There is an extensible scheme for vendor-supplied metadata tags, so that glyphlet makers can add value through novel metadata tags.




## XML representation of glyphlet

- **Well-defined XML representation for glyphlet**
  - Two-way mapping between binary, XML forms
- **Based on RDF, XMP**
- **Metadata easily accessible as entities**
- **Glyphlet file enclosed as CDATA**
- **SING Services Library maps between XML and binary forms of glyphlet**

2004 Adobe Systems Incorporated. All Rights Reserved.

14

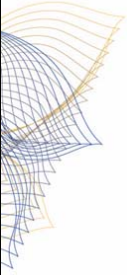


The SING architecture defines an XML language for representing SING glyphlets. Specifically, it represents the contents of the glyphlet's 'META' and 'SING' tables as entities, with a CDATA block which contains the entire content of the binary glyphlet file. This means that there is a standard format for embedding glyphlets in any larger XML entity. This gives SING access to the domain of XML languages for representing documents. How those languages will extend to incorporate SING glyphlets is a matter for the owners of those languages to determine, but at least there is a standard for the SING glyphlet entity itself.

There is a well-defined mapping from any glyphlet file to a corresponding XML entity. The reverse mapping is simple: discard the XML entities, and write the contents of the CDATA block out as a binary file. Thus, there is a lossless two-way mapping between the XML and binary forms of a glyphlet.

The XML representation of the glyphlet is based on Adobe's eXtensible Metadata Platform (XMP), which in turn is based on the Resource Definition Format (RDF). Most of the metadata values are easily accessible as entities by standard XML parsers.


The SING Services Library provides the service of returning the XML form of a glyphlet given the binary form, and vice versa.



## XML representation of glyphlet

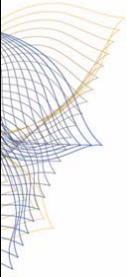
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:SING="http://ns.adobe.com/SING/1.0/">
<SING:UNIUnifiedBaseChars>&#x990C;;v</>
<SING:BaseFontName>KozMinStd-Medium</>
<SING:StrokeCount>14</>
<SING:IndexingRadical>&#x2FB7;</>
<SING:CIDBaseChars>Adobe-Japan1 ;1252 ;v</>
<SING:CIDBaseChars>Adobe-Japan1 ;13650 ;a</>
... [~30 entities omitted] ...
<SING:Self ><![CDATA[src:url("data;base64,\
T1RUTwAFAEAAAQgAQQkFTRYW\ ... igwV1NEZGMQAA")
]]> </SING:Self> </rdf:Description> </rdf:RDF>
```

2004 Adobe Systems Incorporated. All Rights Reserved. 15



In the slide above we show an abridged version of the XML form of a SING glyphlet, in particular the glyphlet ADBE\_KozMin-M\_00990C\_00.gai. The full XML form is 5,710 bytes long. To save space, we abbreviated the closing entity tags, removed line breaks, and left out about 30 metadata entities.

The list of metadata tags, and the semantics of the value associated with each tag, are defined in the META table specification. Note that the above XML entity was produced by the SING Gaiji Technology Preview, and the META table specification governing it is still a draft and subject to change.




## SING addresses problems of gaiji handling in DTP

- **Missing glyphs**
- **Hard to buy gaiji from maker of original font**
- **Hard to exchange documents that use gaiji**
- **Difficult to find and reuse gaiji glyphs**
- **Font selection is inflexible**

- **Embedded glyphlets**
- **Buy glyphlet files**
- **Embedded, Smart**
- **Glyphlet management**
- **Smart**

2004 Adobe Systems Incorporated. All Rights Reserved. 16



The SING architecture has been designed to bring high-quality, high-productivity gaiji handling to Desktop Publishing (DTP) and other document workflows, and resolve the problems associated with the current “Gaiji in DTP” approach as described above. This slide lists several pain points, and how the SING architecture addresses them.

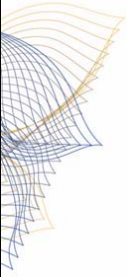
With Type 1 gaiji fonts or special fonts, missing glyphs were a frequent danger or failure mode. SING addresses this by having the glyphlets always be embedded in the document, and so always be available. Custom fonts represented a costly business model for font makers, so users typically had to create their own Type 1 fonts. Since glyphlets are standard, client-independent files, it is now practical for a font vendor to create glyphlets on demand, and then sell those glyphlets via ecommerce to many customers. This gives back to the industry a capability it lost in the transition from optical to digital typesetting.

Document exchange was always difficult and risky, because the glyph IDs for gaiji were site-specific and because it was difficult to propagate gaiji fonts to all parts of the workflow. Since glyphlets are embedded, they travel with the document and are always present. There is no longer any need for gaiji codes. And if exchanging documents involves a format conversion or different layout, the metadata that makes glyphlets “smart” provides the information needed for these operations.

Gaiji font management was a major pain point. Over years, service providers would accumulate dozens of gaiji fonts. Knowing which glyphs for which parent fonts were in which Type 1 gaiji font was difficult. The glyphlet management tool in SING provides a systematic way to know which gaiji a user has.

Gaiji fonts were typically an accumulation of glyphs with differing encodings, so changing font was nearly impossible if the text included gaiji. With the rich metadata of SING glyphlets, it is now practical for layout software to select correct gaiji glyphs when a user changes font. This could represent a dramatic freedom to experiment with design in Japanese graphic arts – now, the designer must commit to a choice of font early, even before the text is entered, and cannot change that choice later.






## SING Gaiji Technology Preview for Adobe® InDesign® CS J version

- **Introduces SING, shows Adobe's intentions**
- **Free download from [adobe.co.jp](http://adobe.co.jp)**
- **Plug-in to add SING to InDesign CS**
- **Glyphlet Management Tool**
- **501 SING glyphlets**
  - Standard Printing Forms (NLC) of kanji for Adobe's Kozuka Mincho typeface
  - Arrow, bullet, and dash symbols

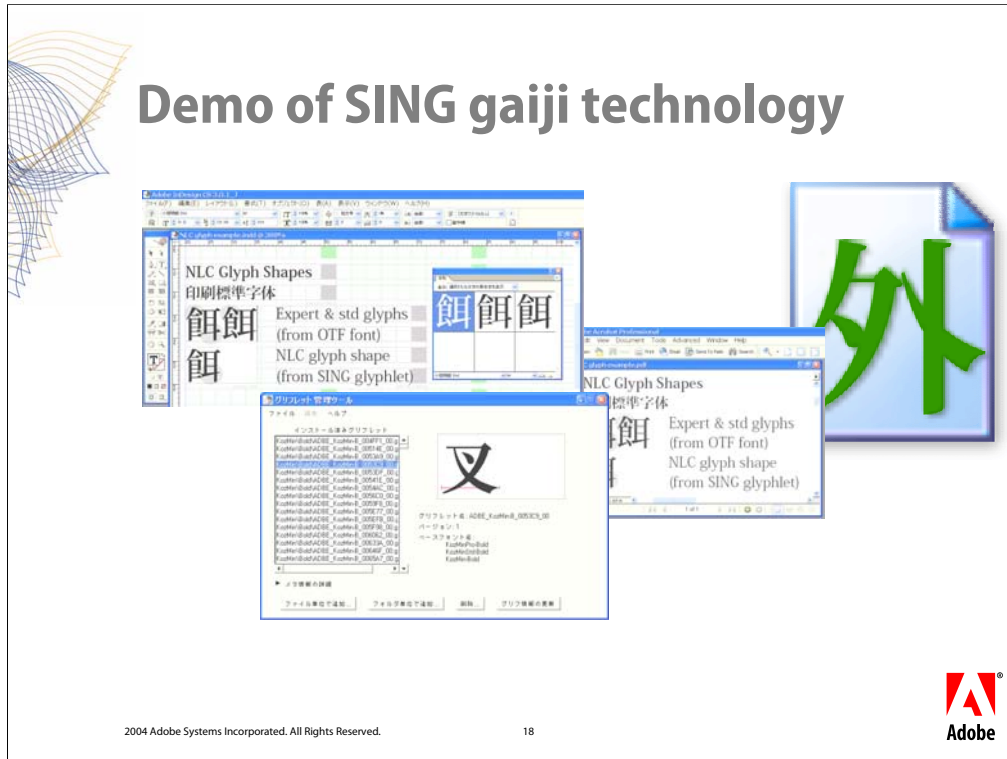
2004 Adobe Systems Incorporated. All Rights Reserved. 17



An architecture as wide-ranging as SING does not appear fully-formed with the first release. Adobe intends to support SING in future versions of its products as market needs dictate, but Adobe also relies on third parties to take up the opportunities offered by the SING architecture. In order to signal its intentions and to give third parties and customers a way to evaluate the SING architecture, Adobe released the **SING Gaiji Technology Preview for Adobe® InDesign® CS, Japanese version** in early 2004.

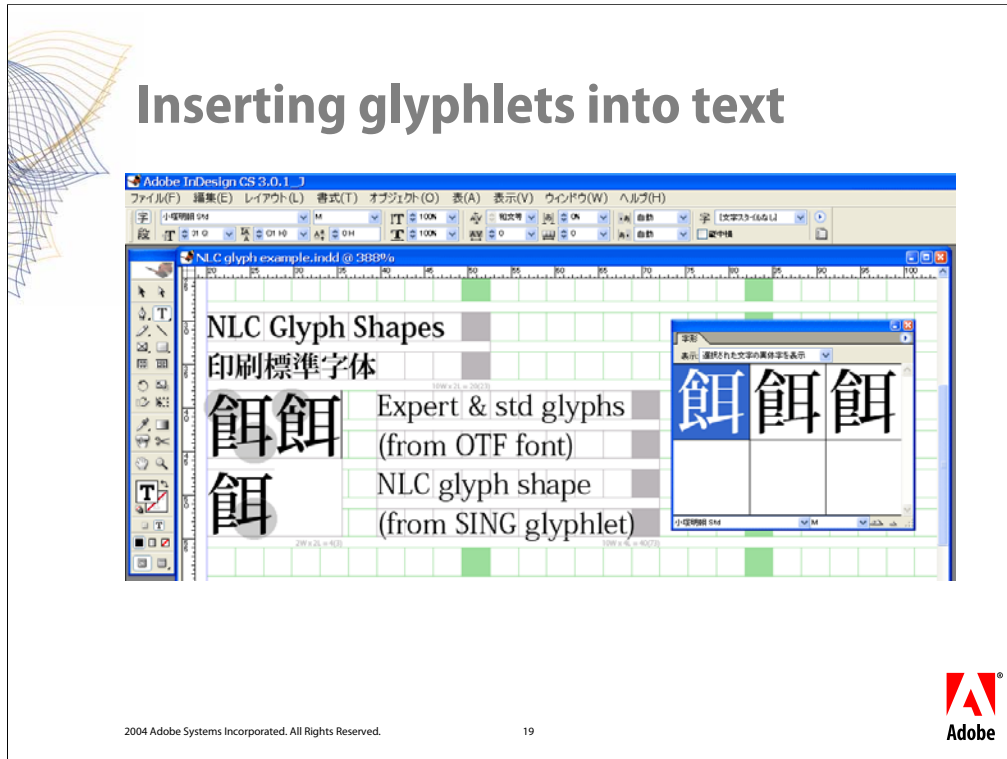
The Technology Preview is a free download from Adobe's Japanese and US web sites, <http://www.adobe.co.jp> and <http://www.adobe.com> respectively. It consists of a small but usable prototype of the SING architecture, which installs on top of Adobe InDesign CS Japanese version.

The Technology Preview consists of three main parts. First, it has an InDesign plug-in which adds SING functionality to standard Adobe InDesign CS (Japanese version). Second, it has a Glyphlet Management Tool utility application, which lets users see a list of the glyphlets they possess, and see some of the metadata for each glyphlet. Third, it comes with 501 glyphlets. Some of the glyphlets are a set of six weights of variant kanji that supplement Adobe's Kozuka Mincho Pro typeface, adding in variant glyph shapes that conform with the Standard Printing Forms (NLC) specification. The rest of the glyphlets are useful "pi" characters (symbols): arrows in various shapes and sizes, bullet marks in different shapes, and dashes of different weights and lengths, including two-Em-wide dashes.



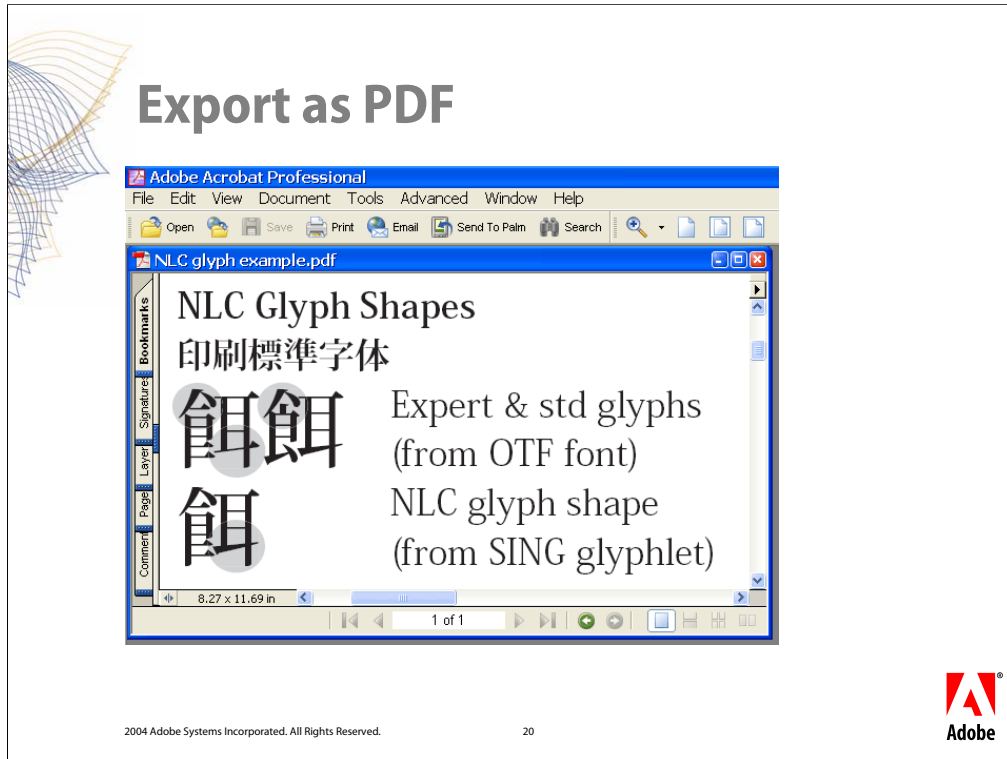
The presentation of this paper at the 26<sup>th</sup> Internationalisation and Unicode Conference will include a brief demonstration of the SING Gaiji Technology Preview. The following pages show screen shots that illustrate some facets of the Technology Preview.

## Inserting glyphlets into text



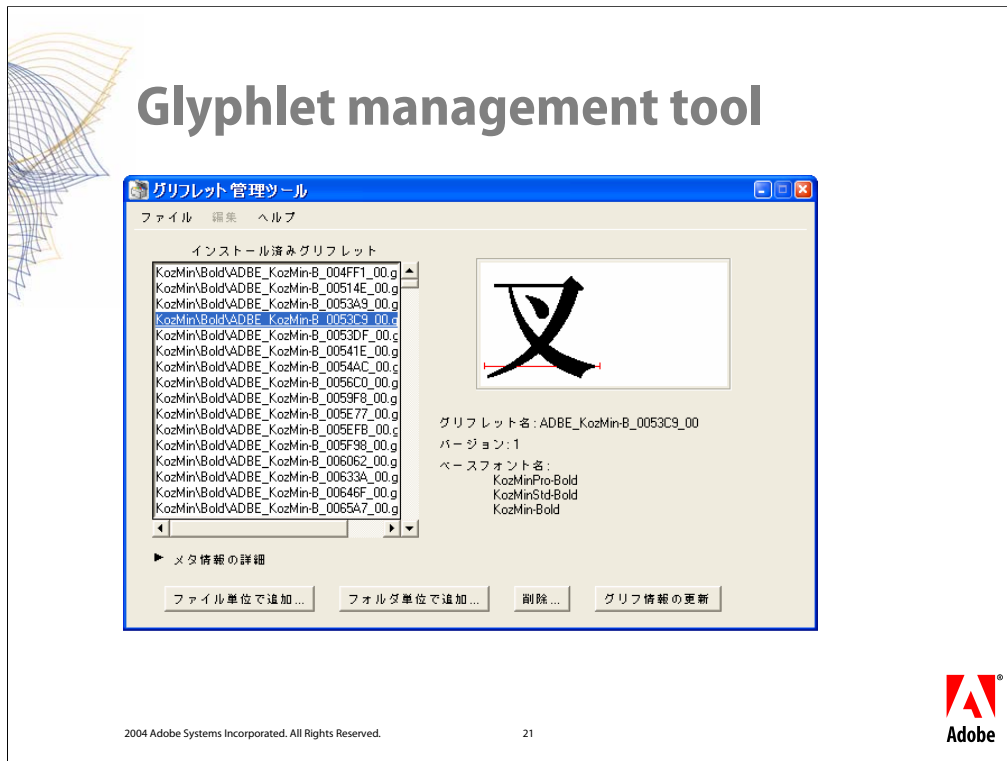
In this screen shot we see a sample text involving SING glyphs being entered into an Adobe InDesign document. In the right is InDesign's Glyphs palette. It shows three variant glyphs being displayed. All correspond to the same Unicode character point. The first two glyphs are provided by the OpenType Pro font, while the third is provided by SING. The user need not know or care whether their glyphs were supplied by the OpenType font or the SING glyphlet.

Grey circles highlight areas of the three glyphs, to show that the glyph shapes are in fact different.



In this document we see a PDF file exported from last slide's InDesign document. Note that the SING text appears in the highest quality, and that it is impossible to know by sight which glyphs were supplied by the SING system and which are part of OpenType fonts. Again, it is a goal of the SING architecture that the user need neither know nor care where their glyphs come from.

In the SING Gaiji Technology Preview, there are no extensions to the PDF format to support SING. When InDesign exports its document to PDF, it converts the SING glyphlets to CFF fonts and embeds them in the PDF page content. The gaiji become "dumb glyphs" and lose all their SING properties. The resulting PDF is completely standard, and compatible with all prepress workflows. The SING architecture calls for the PDF file format to eventually be extended to support SING, so that gaiji can exist in a PDF file and retain their SING properties.

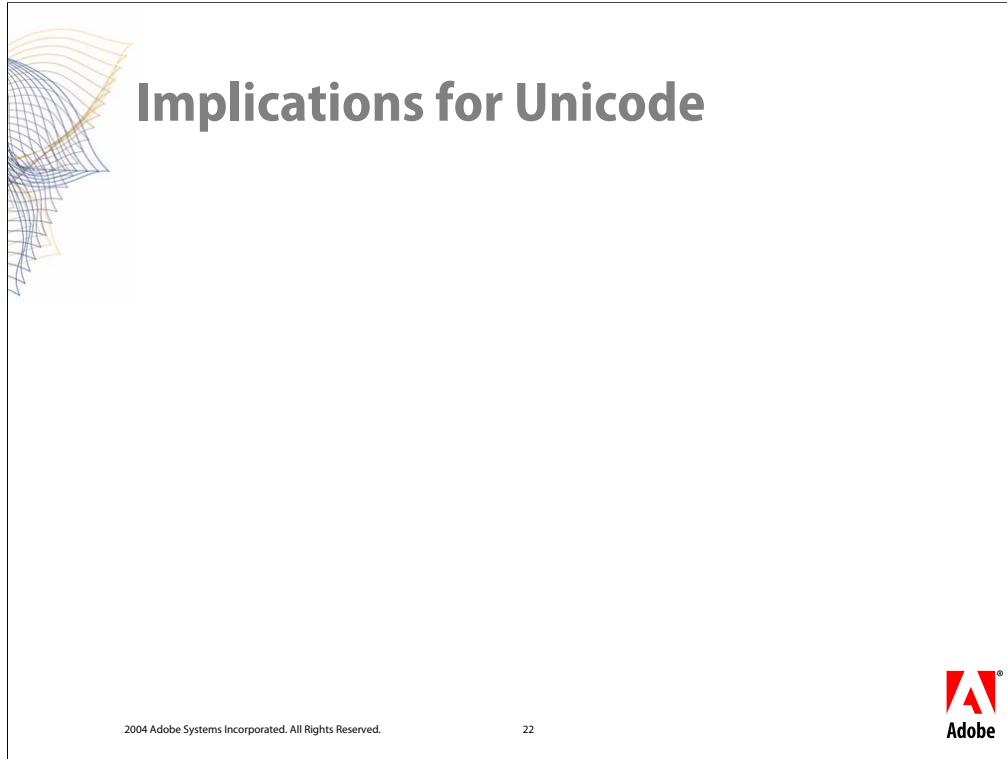


This is a screen shot of the prototype Glyphlet Management Tool (GMT), supplied in the Technology Preview.

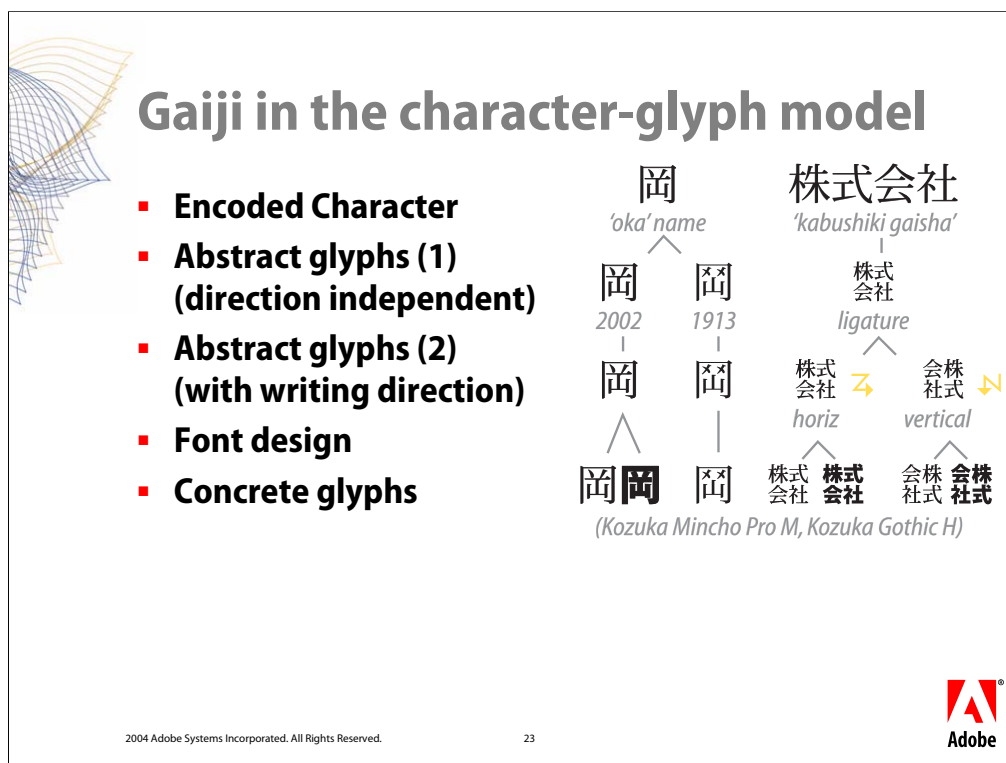
At the left are a list of all the glyphlets presently installed on the system. One glyphlet is selected. On the right is the image of that glyphlet, and some of its most important properties: the unique name of the glyphlet, the version number, and the names of the fonts it corresponds to. Note that the user interface is in Japanese; the Technology Preview is available in Japanese language only.

At lower left, below the list of glyphlets, is a right-facing triangle labelled “Metadata Details”. Clicking on this triangle opens up a rectangular area in the dialog where the GMT displays more of the metadata properties of the glyphlet.

The row of buttons along the bottom allow glyphlet installation (file by glyphlet file, or whole folders of glyphlets at a time); glyphlet deletion; and updating the SING system’s cache of metadata when glyphlets are added or removed without using the GMT.



The gaiji market requirements, and the experience of designing the SING architecture, have some implications for the Unicode architecture. In this section we briefly consider some of those implications. In this section of the paper, we will try to be rigorous about the terms “character” and “glyph”, as used by the Unicode character-glyph model.



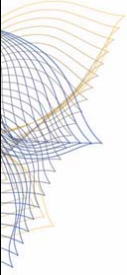
The Unicode Character-Glyph Model, as described in The Unicode Standard, defines the notion of “encoded character” and “glyph”. “Glyph” is defined as being the specific rendered shape corresponding to a character or characters, including all rendering options like OpenType layout, and the choice of font. But there are interesting intermediate layers of abstraction.

In the left example above, you see a character “oka”, as used in the personal name “Maruoka”. The first row shows the conventional modern glyph for this character. In the second row, we see to the right of the modern (2002) glyph, a 1913-era glyph variant. We believe this character is encoded in Unicode, but it certainly is not within the glyph complement of commonly-used publishing fonts.

In the right example, you see the common Japanese phrase “kabushiki gaisha”, meaning “stock company”. It is sometimes set as a ligature, a single glyph that represents all four characters together. Japanese text is commonly written horizontally (left-to-right, top-to-bottom) and vertically (top-to-bottom, right-to-left). So this ligature also appears in two forms, horizontal and vertical, with the parts of the ligature are arranged in different orders. By the way, the “oka” glyphs are the same in both directions.

In the last line of the diagram, you see each a font design applied to the glyphs, yielding different “concrete” glyphs. Three of the four glyphs are formatted with two fonts each: “Kozukua Mincho™ Pro M” and “Kozuka Gothic™ Pro H”. The 1913-era “oka” character is formatted only in Kozuka Mincho.


Looking at this diagram, it is clear that each succeeding line represents a step from character to glyph, from abstract to concrete. The first line is pure character data. The line “Abstract glyphs (1)” specifies ligature formation and historical glyphs, but not writing direction or font design. The line “Abstract glyphs (2)” adds a specification of writing direction. Only in the last line do we specify font design and unambiguously arrive at what the Unicode character-glyph model terms a “glyph”.



## Abstract Glyph is a useful concept

- **It is meaningful to identify glyphs as “the same” except for font variations**
- **Glyph identification schemes are widely used**
  - CID-Keyed Character Collections, e.g. Adobe-Japan1-4
  - Adobe Glyph List naming conventions
- **Practical uses of “abstract glyph” codes**
  - Glyph to character mapping, font change of text
- **Use Unicode Han Variation Selectors?**

2004 Adobe Systems Incorporated. All Rights Reserved. 24



The diagram on the previous page illustrates two intermediate levels of abstraction: before and after the specification of writing direction, but not yet specifying font design. It is meaningful to speak of the glyphs on the bottom row as being “the same” as each other except for font design variation. It is meaningful to speak of the glyphs on the “Abstract glyphs (2)” row as being the same except for writing direction variation. There are other layers that can meaningfully be defined.

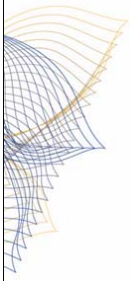
The Japanese desktop publishing and type industry has extensive practical experience with a layer of abstract glyphs, namely the glyphs specified by the CID-Keyed Character Collections defined by Adobe Systems, Incorporated and others. For instance, the “Adobe-Japan1-4” collection describes 15,444 abstract glyphs that differentiate glyph variants, horizontal and vertical forms, and more, but are independent of font design. The CID level of abstraction has proven a useful foundation upon which multiple font vendors could design commercially useful, high-quality typefaces.

One practical use for abstract glyph names in layout software is font changing when formatting text. Layout software can use one font to perform a character-to-glyph mapping, then store the abstract glyph code. If the formatting changes to use a different font which uses the same abstract glyph codes, it is simple and reliable to apply the stored abstract glyph codes to the new font, and display correct glyphs. Without shared abstract glyph codes, this operation is much more difficult.

This leads us to believe that defining Unicode Variation Selectors for the CJKV ideographic script characters to operate at roughly this level of abstraction would prove very useful. It would allow many gaiji to be processed at a character level, using standard Unicode mechanisms, saving many layers of software from having to add specific gaiji support.

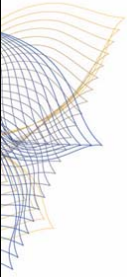
The question of how Variation Selectors will be put to use is, in our opinion, the most interesting open question concerning Unicode support for gaiji.





## Conclusion


- **Gaiji is a strong requirement for CJKV text, and a useful concept globally**
- **Adobe's new SING architecture**
  - Smart, INline Glyphlets
  - "the OpenType approach to gaiji"
- **SING Gaiji Technology Preview for InDesign CS**
- **Implications for Unicode**
  - Abstract Glyph is a useful concept



## For more information

- **SING Gaiji Technology Preview for InDesign CS**
  - [http://www.adobe.com/products/indesign/sing\\_gaiji.html](http://www.adobe.com/products/indesign/sing_gaiji.html) (Eng)
  - [gaiji@adobe.com](mailto:gaiji@adobe.com) for public comments (Jp or Eng)
- **"SING: The Final Frontier For Japanese DTP"**
  - **The Seybold Report** • Vol. 4, No. 10 • August 18, 2004
- **"Gaiji: Characters, Glyphs, Both, Neither?"**
  - 22<sup>nd</sup> IUC, <http://partners.adobe.com/asn/developer/type/gaiji.html>
- **"Japan's 'Second Wave' of DTP ..."**
  - **The Seybold Report** • Vol. 3, No. 9 • August 18, 2003

2004 Adobe Systems Incorporated. All Rights Reserved. 26



Here are some references to useful background information on the gaiji market requirements and SING.

The **SING Gaiji Technology Preview for InDesign CS** is available from the following locations.

- [http://www.adobe.co.jp/products/indesign/sing\\_gaiji.html](http://www.adobe.co.jp/products/indesign/sing_gaiji.html) (Japanese)
- [http://www.adobe.com/products/indesign/sing\\_gaiji.html](http://www.adobe.com/products/indesign/sing_gaiji.html) (English)
- Public comments welcomed to [gaiji@adobe.com](mailto:gaiji@adobe.com) (Japanese or English)

**The Seybold Report** published an article on SING (just in time for the 26<sup>th</sup> IUC). Exactly a year earlier, they published a survey of the Japanese DTP market, with good background on the gaiji requirement.

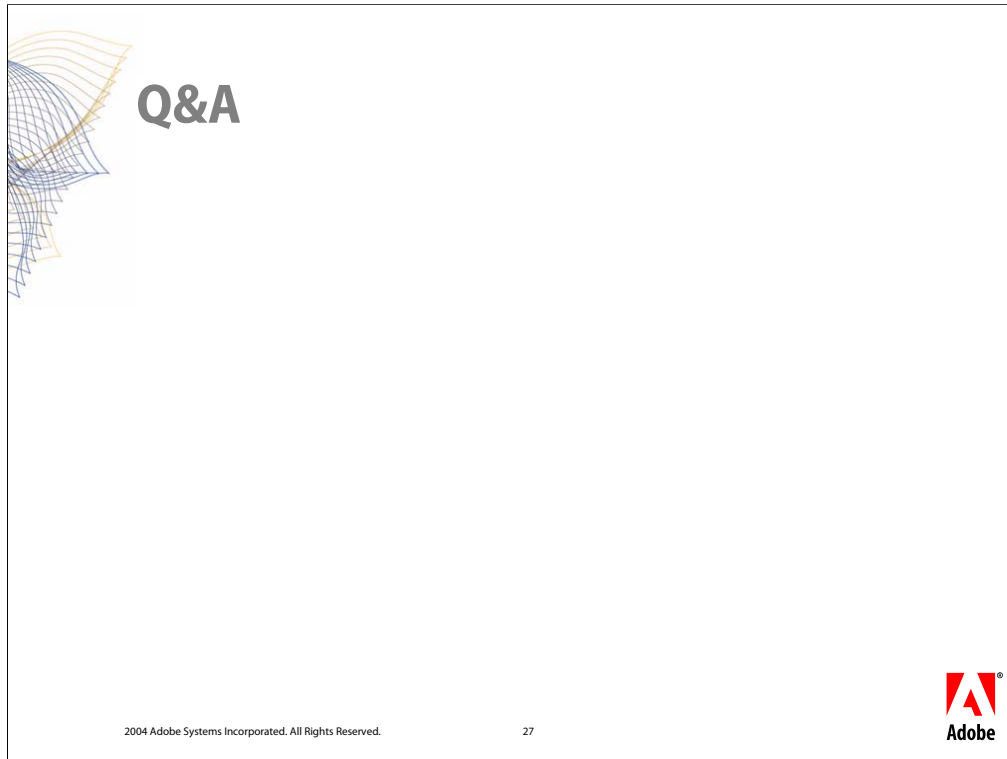
- **SING: The Final Frontier For Japanese DTP**, Joel Breckenridge. Vol. 4, No. 10 • Aug 18, 2004
- **Japan's 'Second Wave' of DTP Sets Stage for CTP Savings**, J. Breckenridge. • Vol. 3, No. 9 • Aug 18, 2003

This paper builds on ideas from my earlier paper, **Gaiji: Characters, Glyphs, Both, Neither?**, by Jim DeLaHunt. Presented to the 22<sup>nd</sup> IUC, September 2002. Abstract: "Unicode encodes Han characters by the tens of thousands, but fonts typically have only thousands of glyphs. Some fonts may have more glyphs, some may have fewer. And since the Han character repertoire is fundamentally open-ended, there will always be characters which are not encoded. The characters legal for the script, but not in your font, are known as "gaiji". Writers and publishers insist on being able to use gaiji, so the Japanese publishing and computer industries have come up with a number of gaiji mechanisms. Looking from the viewpoint of a publishing software and font developer, we describe and evaluate a few of the most important gaiji mechanisms. Finally, we look at gaiji in terms of the Unicode character-glyph model. Are they glyph variants, or characters, or both, or neither?"

- <http://partners.adobe.com/asn/developer/type/gaiji.html>, <http://www.unicode.org/iuc/iuc22/a358.html>

The present paper, **SING: Adobe's New Gaiji Architecture**, will be published at the same URL.

- <http://partners.adobe.com/asn/developer/type/gaiji.html>, [http://www.unicode.org/iuc/iuc26/abstracts\\_b.html#a341](http://www.unicode.org/iuc/iuc26/abstracts_b.html#a341)





**Adobe**

**Tools for the New Work™**