21ST CENTURY 4GL

From green screens to smartphones

Genero®

At the height of its glory, IBM® Informix® 4GL (4GL) dominated the retail, finance, manufacturing, and government sectors due to its talent for creating world-class, reliable business applications both quickly and cost-effectively.

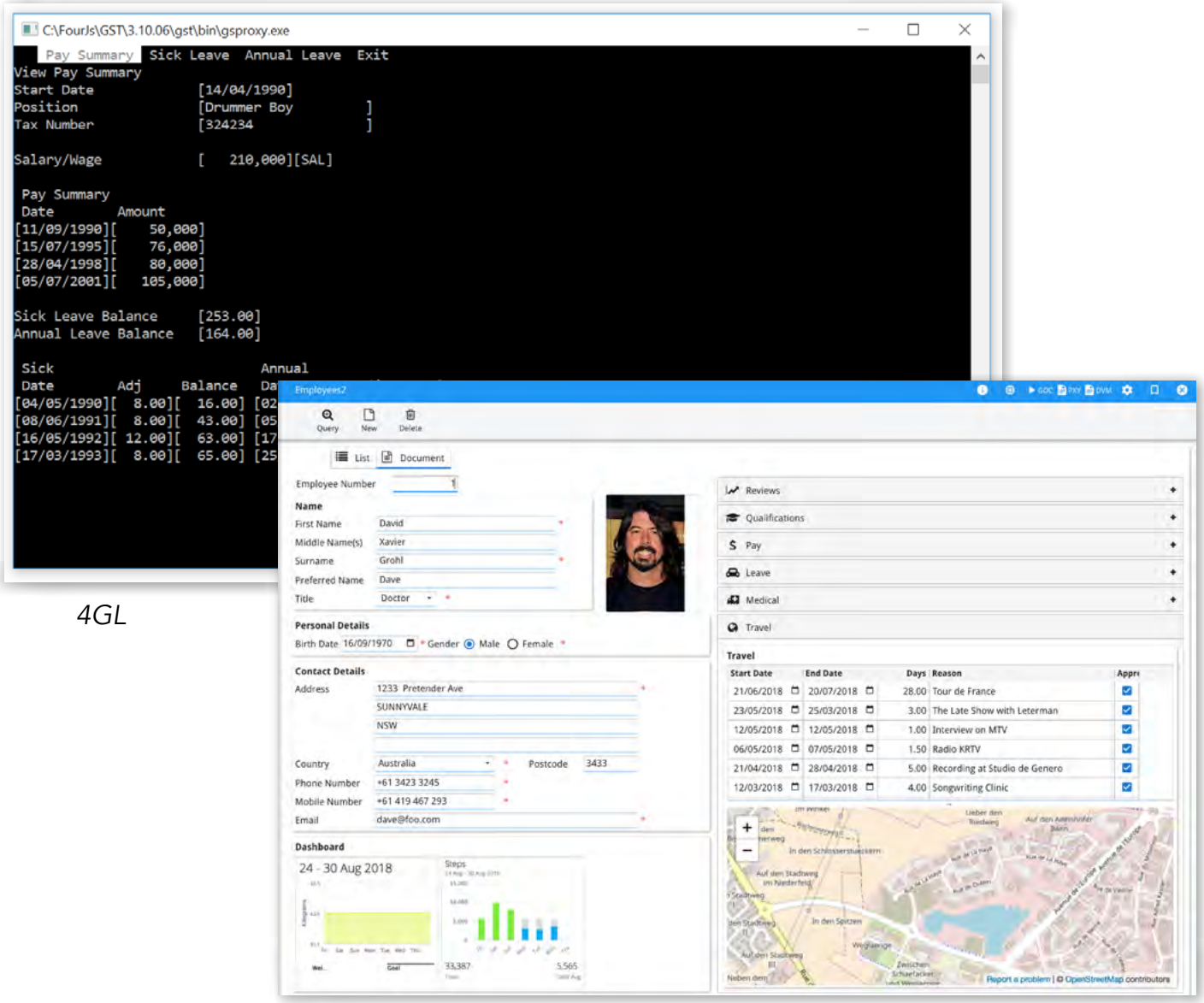
In the mid-nineties however, the advent of client-server computing and object-oriented programming languages put paid to that glory – even if those trends could never match 4GL for developer productivity.

Nevertheless, in a world where IT budgets are constantly under pressure, it remains relevant and pervasive to this day. It is still the most productive way to develop business applications.

Over the last 30 years, technology advances have revolutionized the way we work. During that time, 4GL struggled to keep pace, which is why Genero® came into being. Genero was written from scratch using the latest techniques and technologies, while remaining 100% compatible. Genero respects the significant investment in code and skills made over the last few decades. It is one of the most portable, extendable, and scalable development environments that enables applications to evolve effortlessly in-step with technology trends.

In their Genero iteration, 4GL applications run everywhere – irrespective of client, database or operating system – and with little to no modification. Be it on a green screen, on a macOS® or MS® Windows laptop, in a browser or on a smartphone, applications adapt at runtime from a single source tree. This explains why Genero continues to further the 4GL cause and prosper as the linchpin of so many mission-critical information systems running big business and government institutions.

> Genero protects decades of investment in code and skills.



4GL

Genero

As 4GL never evolved from its 80's 'green screen' roots, it has never provided a contemporary user experience to its customers. As a consequence, it doesn't support any of the ergonomic features provided by today's modern clients: MS Windows, macOS, browsers or mobile devices.

In the absence of tabs, widgets, and toolbars, applications developed with 4GL tend to squeeze as much information as possible into a single form that is split into 'master' and 'detail' windows.

Genero clears the clutter, revolutionizes the user experience and makes applications indistinguishable from the rest.

> Genero is unique in its ability to combine code written 25 years ago with new code that exploits the very latest computing paradigms.

CONTEMPORARY ERGONOMICS

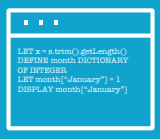
In 4GL, pop-up dialogs provide a mechanism to access more information, but users must exit one before entering another. Users can 'jump' from field-to-field, but only sequentially and at one dialog at a time. 4GL applications are not browser friendly either, where it is easy to lose context through the 'back' button. As a result, they are cumbersome and frustrating to use for those experienced with modern systems. Genero resolves these issues with a state-of-the-art client rendering technology.

INTEROPERABILITY

4GL is also notoriously introvert. It talks to itself very well, and ignores everyone else. It's not that it doesn't want to communicate – it simply doesn't know how to. Modern applications – irrespective of how they were written – interoperate via SOAP and RESTful Web services using XML and JSON data structures. These features have been embedded into Genero to make applications gregarious – able to communicate seamlessly with the outside world and to be used without in-depth knowledge of the underlying protocols.

SECURITY

Applications exposed to the outside world are extremely vulnerable to attack. Developing secure applications that dovetail seamlessly into this complex world is therefore critical. Genero provides classes and methods that perform cryptography, digest algorithms, authentication, certification, and single sign-on so that apps may interact with social media and other Internet oriented systems.



LANGUAGE IMPROVEMENTS

The language has been improved to incorporate features such as named parameters, passing function parameters by reference, event handling, internationalization, and interfacing to Java®. A class of abstract operating system methods covering filesystem management greatly enhances application portability.

AN INTEGRATED DEVELOPMENT ENVIRONMENT

Every modern development language needs a rich environment within which to effectively create. Genero Studio, Genero Report Writer and Genero Ghost Client serve that purpose – to improve developer productivity and test applications for validity, reliability and scalability.

ARCHITECTURE

Genero frees the developer from the shackles of 4GL – namely its mono-database, mono-operating system, monolithic and 'green-screen' culture. By liberating the developer from these constraints, Genero breathes a new lease of life into legacy applications preparing them for today's and tomorrow's deployment spectrum. It does this from within a single development environment, creating applications that are portable, secure, and that scale to thousands of concurrent users without the need for expensive middleware.

CONTEMPORARY ERGONOMICS

It is possible to recompile your 4GL application with Genero 'as is' using a Text User Interface (TUI) and continue to develop as usual in 'ASCII' mode. Alternatively, recompiling in 'traditional mode' renders a graphical user interface, soft function keys, scrollbars and a mouse 'automagically' with no changes to the code needed. This approach is quick, but the user interface retains the constraints of its ASCII geometry. That would be a shame, because the benefits of Genero's Graphical User Interface (GUI) extensions would not be realized.

Perhaps the most important change to the language made within Genero is the introduction of the LAYOUT keyword for .per files. When used instead of SCREEN, the form is rendered through the activation of a superset syntax comprising graphic verbs and tokens.

In many cases, the biggest visual impact can be achieved by just modifying the most important .per files. Genero extends the .per form syntax with the following abstract graphical elements that render natively across Windows, Linux®, macOS, browser and mobile APIs:

Graphical widgets define how values are displayed and edited in a GUI form:

- BUTTON – present a clickable or tappable zone on the screen that triggers an action
- BUTTONEDIT – an edit with an associated button used to trigger an action for editing or providing more detail about a value
- CHECKBOX – user may toggle between two (or three including NULL) values
- COMBOBOX – user may select from a scrollable list of values
- DATEEDIT – user may select a date from a calendar widget
- DATETIMEEDIT – user may select a date and time from a specialised widget
- EDIT – default rendering and editing of a value
- IMAGE – render an image
- LABEL – display text that is not editable e.g. titles, descriptions
- PROGRESSBAR - display an integer as a bar that increases to the maximum value
- RADIOGROUP – user may select one value from multiple choices
- SLIDER – user may increase or decrease an integer value by sliding a widget
- SPINEDIT – user may adjust an integer value by selecting up/down buttons
- TEXTEDIT – user may edit multiple lines of text
- TIMEEDIT – user may select a time from a specialised widget

Layout containers define how GUI widgets are arranged relative to one another

- GRID – position child widgets with a given X, Y, width, and height parameter
- SCROLLGRID – view and scroll through multiple instances of a GRID
- TABLE – arrange child widgets into one or more identical rows of a table-like structure
- TREE – similar to TABLE but with a collapsible, hierarchical tree structure
- STACK – arrange child widgets in a style that is native to the user interface
- GROUP – place a border around several child contents and optionally give a title
- FOLDERS + PAGES – arrange child containers into multiple pages where only one page's content is visible, and the others appear as tabs or collapsible accordions.
- VBOX – stack child containers vertically
- HBOX – arrange child containers horizontally

Webcomponents – enables use of existing 3rd party HTML technologies (HTML/JS/CSS) and creation of custom UI components. Example uses include: embedding videos, HTML Documents, maps, charts, rich text editing, calendar, signature capture, rich tables, image maps / POS touch screens, SSO authentication. Pre-defined Webcomponents provide rich text editing, multiple image selection, and SVG rendering.

Action defaults that centralize the definition of text, comments, images, accelerators, and other properties of actions

Top menus (Menu bar) – a structured arrangement of menu options at the top of a window

Toolbars – an arrangement of buttons along the top, bottom, left or right of the form

Parallel dialogs – enables a split screen with each pane executing independent code

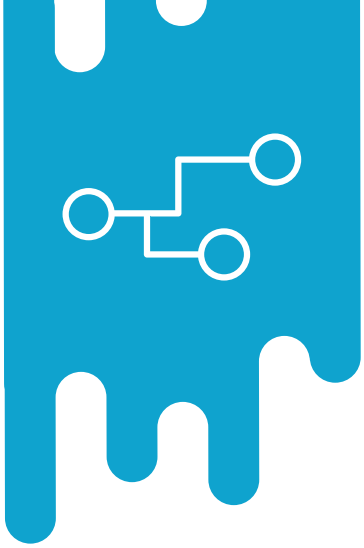
Multiple dialogs – merge multiple user-interaction statements (MENU, INPUT, INPUT ARRAY, DISPLAY ARRAY, CONSTRUCT) into a single active dialog facilitating user navigation

Dynamic dialogs – a user-interaction statement (INPUT, CONSTRUCT, DISPLAY ARRAY, INPUT ARRAY, DIALOG,…) that is defined at run-time, rather than at compile-time. This allows the use of generic code with a variable data structure. Best used in conjunction with creating forms dynamically at run-time, and using dynamic database cursors from the base.Sqlhandle class.

Multiple selects – the ability to select multiple rows within an array, for the purposes of performing an action such as deleting or printing

Focus on field – the ability to select a cell from a DISPLAY ARRAY, not just a row

Drag and drop – enables the dragging and dropping of certain objects on top of each other to perform a given action, within or between applications



INTEROPERABILITY

Genero has evolved over the last 25 years to include the latest APIs and data structures for interaction with other systems. This includes the most popular data structures: XML and JSON along with the most popular methods: Web Services, both SOAP and RESTful.

Data Structures

XML

Genero provides a comprehensive set of classes and methods for working with XML documents. This includes support of both the DOM (Document Object Modelling), and StAX (Streaming API for XML) models as well as serialization and XSLT transformations.

JSON

Genero provides classes to work with JSON (JavaScript Object Notation) – a well-known lightweight data interchange format.

Methods

Web Services – Genero provides libraries to create Web Services, both as the server or provider of the Web service, and as the client of the Web service.

SOAP – a key part of this is the WSDL file which provides the interface details. Take an existing 4GL function with defined input and output parameters, and create or publish a Web Service operation based on that function.

RESTful – more lightweight than SOAP. Pass messages (JSON, XML typically) in a format of your choosing.

File – use of these classes all but eliminates the need for 'C' libraries.

Base.Channel – prior to Web Services, you might have interfaced to other systems by reading/writing to files. Since 4GL's system call API was so poor, you had to write libraries written in C and/or shell scripts. The base.Channel class is a set of methods used for reading and writing to files and can be used to replace C and shell scripts.

OS – this class enables listing files in directories and testing for file existence and permissions. This removes OS dependencies and makes code portable.

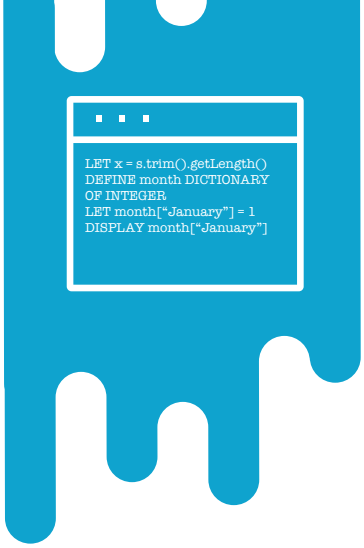
Sockets – methods were added to the base.Channel class to facilitate socket communication, both as a client or a server to/from a given server and port.

Pipes – the openPipe method was added to the base.Channel class to facilitate communication with a sub-process via a pipe channel.

SECURITY

Security Class – the security package provides classes and methods to perform basic cryptography. These include PBKDF2 and BCrypt classes for the storing and checking of encrypted passwords. Also provided are classes to implement digest algorithms such as SHA512, MD5 (etc.) along with classes for HexBinary and Base64 algorithms.

Secure Communications – if an application wants to exchange messages with a financial, business, or personal application on the web, it must be able to authenticate the origin of the message, ensure that no malicious application has altered the original message, and ensure that no third party application can intercept the message. This involves certificates, certificate authorities, private keys etc. all provided for by Genero.



LANGUAGE IMPROVEMENTS

The 4GL language syntax has been greatly enhanced to remove its limitations and conform to modern-day coding practises. Here is a non-exhaustive list of improvements we have made:

Syntax

Keywords – STRING, CONSTANT, BOOLEAN, TYPE, BIGINT, TINYINT, PUBLIC, PRIVATE, NVL, IFF, SFMT, DICTIONARY, DYNAMIC ARRAY, TRY/CATCH

ON ACTION – the ON ACTION block executes code when a user event occurs:

```
ON ACTION zoom
```

rather than:

```
ON KEY(F5)
```

The 'zoom' action is specified by a .per form or action default file, and executed irrespective of how it was triggered.

Passing function parameters by reference – this feature is useful when dealing with complex structures, records, and arrays. Complex structures are passed to the function without the need to deal with the returning result. Only one instance of it resides in memory, avoiding two copies (one in the calling function and one in the called function), so consuming less memory.

```
DEFINE arr DYNAMIC ARRAY OF complexDataType

# arr contains untransformed data
CALL transform_array(arr)
# arr contains transformed data
```

Literal initialization – literals can be initialized simply:

```
TYPE colorType RECORD
   red, green, blue INTEGER
END RECORD

DEFINE purple colorType = (255,0,255)
```

Case sensitivity (optional) - if selected, this becomes invalid syntax:

```
DEFINE foo, Foo, FOO STRING
```

and:

```
DEFINE accountCode STRING
DEFINE accountCodeLength INTEGER
        LET accountCodeLength = accountCode.getlength()
```

Named parameters – Named parameters remove the need to specify every argument and get their order right when calling functions. Unspecified parameters are set to default values. Improves the code readability.

```
CALL fgl_report_configurePDFDevice(fromPage = 1, toPage = 1)
```

Other features – these are examples of syntax seen in other languages, but not seen in 4GL. This facilitates migration for developers unaware of 4GL's limitations.

Type declarations are allowed in calling parameters and return types of functions – a shorthand way to improve type safety:

```
FUNCTION add(x INTEGER, y INTEGER) RETURNS INTEGER
```

Shorthand object method chaining:

```
LET x = s.trim().getLength()
```

A new DICTIONARY data type (aka. hash table) uses a string as a key for indexing tables. It can store all Genero data types:

```
DEFINE month DICTIONARY OF INTEGER
    LET month["January"] = 1
    DISPLAY month["January"]
```

ANYRECORD – *generic record type*
At compile time, only a function's type (parameters, return values) is known when using function references. The ANYRECORD generic record type is used when a function takes as input a record whose structure is unknown.

IMPORT – if there is an external library you would like to use within your code, the IMPORT statement can be used as follows:

- `IMPORT FGL` - to import a 4GL library
- `IMPORT` - to import a 'C' library
- `IMPORT JAVA` - to import a Java library

Our developer community uses for example the IMPORT JAVA statements for the following purposes:
- Creating MS® Word and Excel files using the Apache POI library
- Uploading videos to You-Tube®

Internationalization – Genero applications are designed to work the world over via support for UTF-8 and character-set independent code.

A *Localized String* is application text that is defined in string catalogs outside the standard source code. This facilitates the maintenance of translated text, and the use of a particular translation at run-time without the need to recompile.

An *Application Locale* determines which regional catalog is in use for a given user client. Thus the same application instance can support multiple languages simultaneously. Right to left tracking is also supported for Arabic regions and both BYTE and CHARACTER length semantics are supported for Asia.

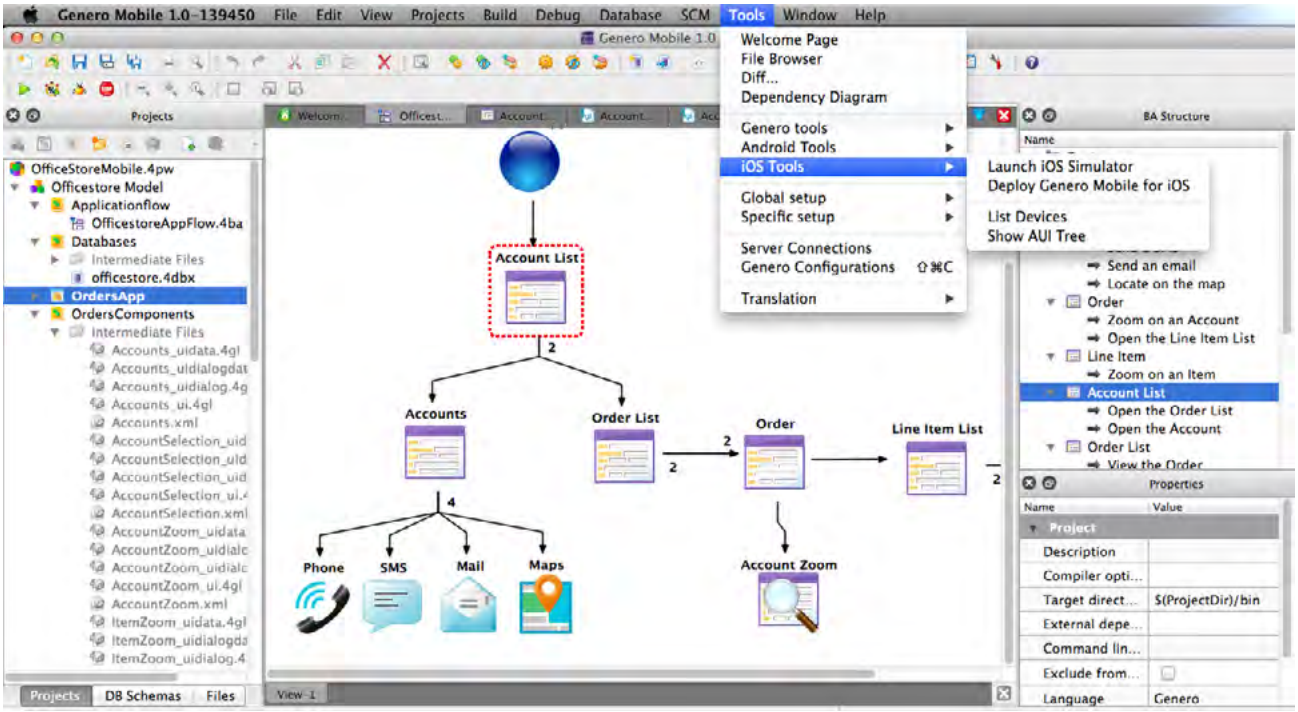
AN INTEGRATED DEVELOPMENT ENVIRONMENT

Genero Studio is an Integrated Development Environment (IDE) designed to improve developer productivity. It enables the simultaneous debugging of applications developed for the desktop, browsers, and mobile devices.

It includes a language-sensitive code editor that supports code completion, real-time syntax checking, keyword highlighting of 4GL and other file formats. It also includes a graphical debugger, a graphical forms designer, a graphical report writer, a project manager for 'making' complex applications, a database schema builder, a meta-schema, a code generator, a visual modelling tool, profiling and analysis tools.

A LOW CODE DEVELOPMENT PLATFORM

Genero Studio's Business Application Modeller (BAM) is a low-code development platform (LCDP) destined for developers looking for an incremental, continuous, cross-platform development life-cycle using diagrams rather than hand-coding. It is ideal for the creation of brand new applications.



BAM - integrating native smartphone apps into a Genero Mobile module

Code for applications and Web services is defined in templates and modelled visually, which guarantees consistency and structured code organized in layers:

- Data storage
- Data services
- Web services publication (RESTful, SOAP)
- Forms and reports

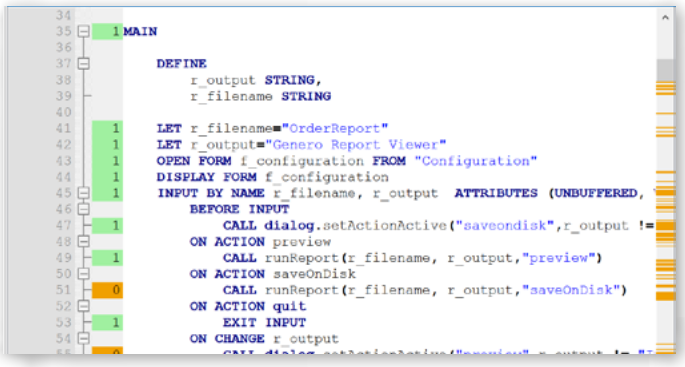
Most business specific coding patterns are covered in the default templates as 'CRUD' forms, lists, details, filtering, navigation, concurrency management, and reports. Native mobile apps such as contacts, calendar, and phone can be tightly integrated via the description into the model. This allows large swathes of the application to be described as diagrams with no coding whatsoever. Diagrams are modified over time by the developer to improve or alter the behaviour of the application. Bespoke business rules can be coded in events.

POWERFUL DEVELOPMENT AIDS

These tools can be activated at compile or run-time from within Genero Studio or from the command line.

Profiler – profiler identifies program bottlenecks. It highlights the percentage of time spent in functions, and how many times functions are called from other functions.

Code coverage – the source code coverage tool analyses whether a test process is testing every line of code in an application. Run a program many times with Code Coverage enabled and a counter will be kept for each individual line of code as it is executed.



```
34
35  1 MAIN
36
37        DEFINE
38            r_output STRING,
39            r_filename STRING
40
41  1     LET r_filename="OrderReport"
42  1     LET r_output="Genero Report Viewer"
43  1     OPEN FORM f_configuration FROM "Configuration"
44  1     DISPLAY FORM f_configuration
45  1     INPUT BY NAME r_filename, r_output  ATTRIBUTES (UNBUFFERED,
46        BEFORE INPUT
47  1        CALL dialog.setActionActive("saveondisk",r_output !=
48        ON ACTION preview
49  1        CALL runReport(r_filename, r_output,"preview")
50        ON ACTION saveOnDisk
51  0        CALL runReport(r_filename, r_output,"saveOnDisk")
52        ON ACTION quit
53  1        EXIT INPUT
54        ON CHANGE r_output
```

Trace – Trace is a useful debugging feature that creates a full or filtered log of function calls that were made by a program, what the arguments passed were, and what was returned

Shared libraries – the deployment architecture is based on shared libraries, which removes the need for building 'runners'

Tools for indentation and refactoring

AUTOMATED TESTING (GENERO GHOST CLIENT)

The Genero Ghost Client provides the following capabilities:

- Performs unit tests using scenarios that validate business logic
- Performs load tests to determine how many users the infrastructure can support
- Benchmarks response times at high loads

A REPORT GENERATOR (GENERO REPORT WRITER)

The REPORT statement in 4GL is used to produce management reports, which by today's standards look very primitive.

Genero Report Writer provides a sophisticated publishing system for the production of executive level business reports, invoices and shipping labels to name but a few uses. Its particularity lies in its ability to stream high-volume documents to printers, displays or indeed to many different file formats including: pdf, svg, xlsx, docx, html, ps, jpg, and png. Streaming enables immediate first page printing when there are hundreds or even thousands of pages to print. It saves valuable system resources for voluminous documents by eliminating the need to create large temporary files.

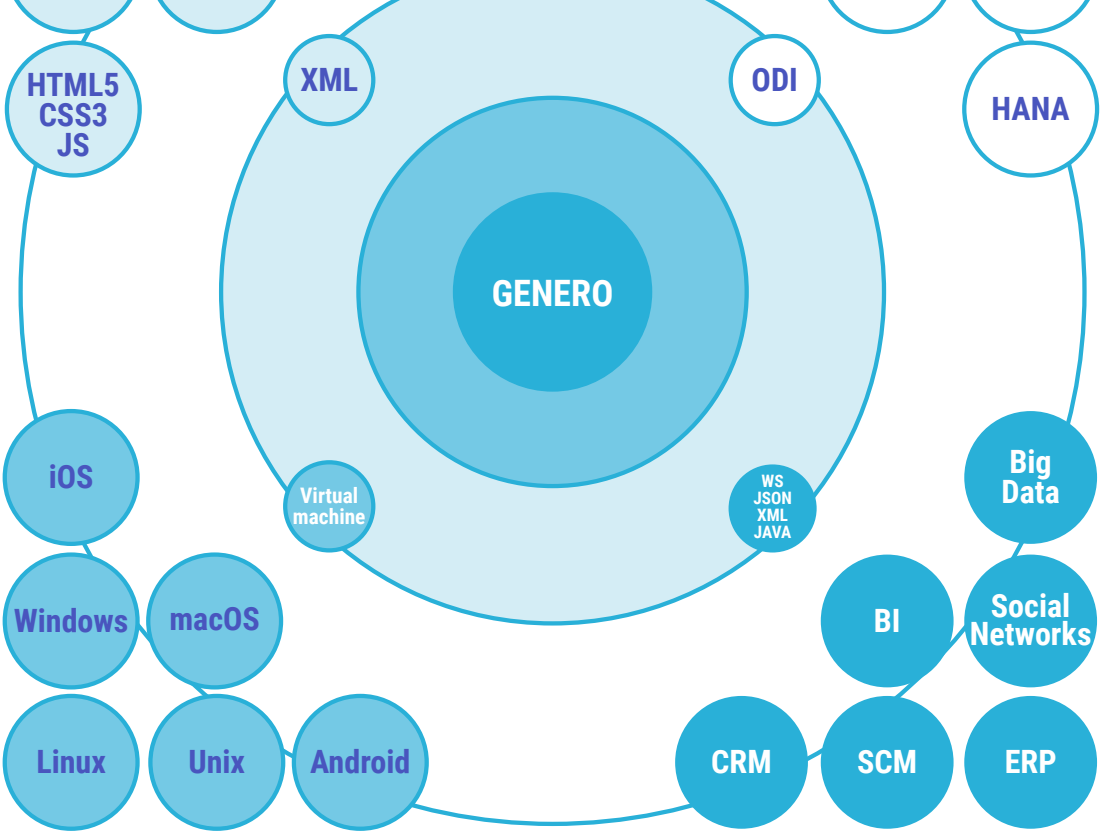
Genero Report Writer supports the most popular graphical objects such as: images, tables, barcodes, QR codes, and charts, as well as manages different typefaces, point sizes and colours. Layouts are dynamic to facilitate the transformation of documents from one form factor to another, without the need to rework the design.

Reports are created with Graphical Report Designer – a near WYSIWYG designer that enables page formats to be modified simply without the need to revisit the report layout.

Genero Report Writer can be used with languages other than Genero such as Java, 'C' and PHP.

Compatibility mode – 'Compatibility mode' allows existing 4GL Reports to be output using any of the methods supported by Genero Report Writer.

ARCHITECTURE



Secure, portable, and scalable to thousands of users without the need for expensive middleware.

WORLDWIDE OFFICES

USA & CANADA
Four Js Development Tools Inc.

251 O'Connor Ridge Blvd. Suite 125
Irving, Texas, 75038
United States

Toll free phone: 866 314 7300 (Free)
Phone: (972) 893-7300
Fax: (972) 893-7304

LATIN AMERICA
Four Js Development Tools Latin America SACV.

Insurgentes Sur 1602
Col. Credito Constructor
Mexico City, DF 03940
Mexico

Phone: +52 (55) 1000 9160

EUROPE
Four Js Development Tools Europe Ltd.

Suite 5, Rineanna House
Shannon Free Zone
Shannon V14 CA36
Co Clare, Ireland

Phone: +353 61 708 314
Fax: +353 61 708 315

ASIA
Four J's Development Tools Asia

Suite 210 29 Kiora Rd,
Miranda NSW, 2228 Sydney
Australia

Phone: +612-8004-5890

FRANCE
Four Js Development Tools SARL.

28 Quai Gallieni
92 150 Suresnes
France

Phone: +33 1 41 38 86 30
Fax: +33 1 41 38 84 46

UNITED KINGDOM
Four Js Development Tools UK LTD.

Regus House, Victory Way Admirals Park
Dartford, DA2 6QD
United Kingdom

Phone: +44 (0) 370 111 5140
Fax: +44 (0) 370 111 5154

CENTRAL EUROPE
Four Js Development Tools Software Vertriebs GmbH

Leonhardsweg 2,
82008 Unterhaching
Germany

Phone: +49 89 60815400
Fax: +49 89 60815402

IBERICA
Four Js Development Tools Iberica LTDA.

Martinez Villergas 49, 1a Planta,
28027 Madrid
Spain

Phone: +34 91 047 76 61
Fax: +34 91 047 76 57

ITALY
Four Js Development Tools Italia

Via Cipriani, 2
42124 Reggio Emilia
Italy

Phone: +39 (0)522 420786
Fax: +39 (0)522 420768

www.4js.com

FOUR Js
The Power of Simplicity