

Technical White Paper



Requirements of a Reporting Application Platform

The economic climate is applying enormous pressure upon businesses to increase the utility of the information collected from the transactional systems that automate their operations. While the most obvious operational inefficiencies have likely been addressed, these organizations must use information in new ways to monitor, measure and evaluate their performance in order to uncover the next levels of improvement. Business are always striving to increase revenue making opportunities and reduce the expenses associated with achieving them. Meanwhile, companies must demonstrate their adherence to corporate governance standards as defined by financial, healthcare and other regulatory authorities or pay millions in fines. They need to recognize and react quickly to changing economic conditions, essentially improve their business agility or become the economy's next victim. They must do this while leveraging their existing investments in the systems—customer portals, ERP and CRM applications, data warehouses, heterogeneous hardware environments and software—that are already in place, running their businesses. This leverage can be obtained by deploying Reporting Applications that empower 100% of the enterprise by giving each different information consumer the critical information they need, in the formats they expect, as it's needed, allowing them to put into motion the most appropriate action and then evaluate its success.

This document outlines the key requirements of a Reporting Application platform in the areas of content creation, content manipulation and transformation, content management and security, application integration and information delivery through a scalable and manageable architecture.

Notice

The information in this white paper is proprietary to Actuate Corporation ("Actuate") and may not be used in any form without the prior consent of Actuate.

© 2006 by Actuate Corporation. All rights reserved.

Actuate Corporation trademarks and registered trademarks:

Actuate, e.Analysis, e.Report, e.Reporting, e.Spreadsheets, e.Spreadsheets Engine, Internet Spreadsheet, Live Report Extension, ReportCast, Report Encyclopedia, SmartReports, Spreadsheets Everywhere, SmartSheets, Tidestone, and XML reports.

All other trademarks are property of their respective owners.

Version 4 – February 1, 2006 © Actuate Corporation

Actuate Corporation

701 Gateway Boulevard
South San Francisco, CA 94080
Tel: (888) 422-8828
<http://www.actuate.com>

Table of Contents

Introduction.....	6
Summary of Requirements.....	8
User Segmentation and Deployment Requirements	9
100% adoption of reporting across the enterprise	9
Wide variety of reporting applications	9
Zero training query capabilities	9
Pure browser viewing of DHTML content, no plug-ins or applets	9
Flexible output formats	9
Purpose-built design environments	9
Metadata and Data Integration Requirements.....	10
Multi-source metadata layer	10
In-transit information integration and transformation	10
Caching data within metadata layer	10
Data-centric design environment.....	10
Common data transformation & integration language	10
Hierarchical metadata objects	10
Logical metadata representation	10
Impact analysis of data changes.....	10
Drag and drop metadata design environment.....	11
Extensible data connector framework	11
Dynamic-parameter selection via metadata query.....	11
Reporting Application Design Requirements	12
Visual development environment	12
Supports programming language to extend designs	12
Extensible Component-based paradigm	12
Content layout controls, builders and editors	12
Post-query data processing and calculations.....	12
Create content with any look-and-feel.....	12
Leverage familiar design paradigms	12
Access any data source, multiple data sources.....	12
Direct access to SAP Business Warehouse	13
Direct access to Java and .NET objects	13
Specify XML tags for information	13
Support for actionable content via embedded code.....	13
Parameterized control over data access, calculations and formatting.....	13
Automatically generated user interface.....	13
Reporting Application Deployment Requirements	14
Choice of content caching options	14
Automatic content versioning	14
Storage of cached content in flexible hierarchical system	14
Scheduled generation of cached content.....	14
Flexible security infrastructure.....	14
Leverage existing security services without replication	14
Secure pages of content without creating multiple documents.....	14
Full support for printing.....	14
Direct PDF generation	15
Content available as XML stream	15
Reporting Applications Presentation Requirements.....	16
Portal integration	16
Modular, seamless web site integration	16
Automatic demand paging.....	16

Free-form hyperlinking into and out of content.....	16
Powerful content search facilities.....	16
Content navigation facilities.....	16
Progressive page viewing.....	16
Localized content.....	16
Deliver and manage reports in multiple languages using Unicode....	17
Layout optimized for browser based viewing.....	17
Frameless delivery of embedded content.....	17
Seamless web-based workflow.....	17
Comprehensive Spreadsheet Management Requirements	18
Support for fully functional Excel content.....	18
Query-driven live spreadsheets.....	18
Spreadsheets built by design.....	18
Spreadsheet design environment.....	18
Use existing spreadsheets as templates for new content design.....	18
Pivot table definition and generation.....	18
Dynamic matrix definition and generation.....	18
Worksheet bursting within workbook.....	18
Pre-defined data filters.....	18
Server managed spreadsheets.....	19
Visual basic macro support within spreadsheet.....	19
Multi-data sources and queries in worksheets and workbooks.....	19
Smart worksheet assembly from centralized cache.....	19
Ad-Hoc Query and Ad-Hoc Reporting Requirements.....	20
Save, share and schedule query results.....	20
Insulate end users from data sources.....	20
Simple, end-user defined queries and ad-hoc reports.....	20
Integrated security for ad-hoc reporting.....	20
Ad-hoc query access to metadata layer.....	20
Flexible output choices for ad-hoc query results.....	20
Web and fat client report template development.....	20
Common design format.....	20
Extensible design format with Java.....	21
Progressive, WYSIWYG web construction.....	21
Web-based report viewing.....	21
Reusable template-driven construction.....	21
Integrated security for web reporting.....	21
Robust formatting for web-based report construction.....	21
OLAP Analysis Requirements.....	22
Multi-dimensional OLAP Analytics.....	22
Brand-able OLAP viewing experience.....	22
View dimensions and measures.....	22
Drag and drop manipulation of dimensions and measures.....	22
Multiple analytic viewing options.....	22
Save analytic views as reports.....	22
Link to reports or cubes.....	22
Calculated columns and conditional formatting.....	22
Hierarchy and metric data filtering.....	22
Flexible personalized cubes.....	22
Support large numbers of users.....	23
Cube definition environment.....	23
Customizable viewing environment based on consumer skill level ...	23
Ragged hierarchy support.....	23
Extensible cube design.....	23
Content presentation API.....	23

Platform Services and Architectural Requirements	24
Comprehensive platform support	24
Complete installation	24
Platform independent APIs.....	24
URL-based APIs.....	24
Seamless integration to .NET and J2EE portal environments	24
Clustered architecture	24
Processing scalability	24
Storage scalability	24
Highly available architecture	25
Seamless integration of structured and unstructured data	25
Ability to generate, distribute and store all types of content.....	25
Automatic content archiving	25
Push notification to alert users of new content.....	25
Remote administration for system-wide control	25
Email notification and report delivery	25
Server-side processing.....	25
System monitoring capabilities.....	25
Log consolidation and centralized reporting.....	26
Support for SNMP monitoring tools.....	26
Online system backup	26
Migration, upgrade and coexistence	26

Introduction

The economic climate is applying enormous pressure upon businesses to increase the utility of the information collected from the transactional systems that automate their operations. While the most obvious operational inefficiencies have likely been addressed, these organizations must use information in new ways to monitor, measure and evaluate their performance in order to uncover the next levels of improvement. Businesses are always striving to increase revenue making opportunities and reduce the expenses associated with achieving them. Meanwhile, companies must demonstrate their adherence to corporate governance standards as defined by financial, healthcare and other regulatory authorities, or pay millions in fines. They need to recognize and react quickly to changing economic conditions, essentially improve their business agility, or become the economy's next victim. They must do this while leveraging their existing investments in the systems—customer portals, ERP and CRM applications, data warehouses, heterogeneous hardware environments and software—that are already in place, running their businesses. This leverage can be obtained by deploying Reporting Applications that empower 100% of the enterprise by giving each different information consumer the critical information they need, in the formats they expect, as it's needed, allowing them to put into motion the most appropriate action and then evaluate its success.

Nearly every individual involved with an organization, whether an employee, a customer or partner, a business analyst or an executive, needs information contained within that organization's transactional applications. These applications, such as SAP, PeopleSoft and Siebel, are used to automate the business processes within the company. Unfortunately, these applications do not typically include capabilities to allow these different user constituencies to reflect upon the data stored within and take action based upon that reflection. These users need their own Reporting Applications that safely obtain and present the transactional data in ways that empower these information consumers.

Reporting Applications can transform transactional data into forms and formats that are easily consumed by all types of users. Application types include performance scorecards and visual, interactive dashboards for executives; secure, personalized customer and partner information portals; personally tailored spreadsheets for customer-facing business users; and interactive, analytic applications for power users.

Reporting Applications extend the reach of enterprise data beyond the handful of report developers and power users served by traditional business intelligence tools. Reporting Applications recognize that there are both different types of information consumer and different types of reports that they consume. Analysts estimate that this population represents 95% of the audience for enterprise data, and that they expect to receive content that is:

- **High Fidelity and Quality**—Content of any complexity needs to be formatted in a professional, usable and attractive manner. This can manifest itself in any number of shapes and sizes. Briefing books, balance sheets and brochure-quality web pages are all examples of the requirements for both high fidelity and high quality information.
 - **Personalized**—Content must be tuned to the precise needs of each individual, their role, context and their preferred presentation and analytic format. This now includes not only web reporting, but also spreadsheet, ad-hoc and analytic report consumption.
 - **Localized**—Content must be presented in the consumer's native language and cultural context without sacrificing or compromising any of the other requirements.
 - **Interactive**—Information consumers should be able to navigate and create customized content easily with no training.
 - **Ready to Review**—Content should fulfill end user needs to obtain and manipulate the information using skills commonly found in the enterprise, such as web browsers and Microsoft Office.
- Privately Branded**—The content should seamlessly blend with web applications, sites, portals and printed materials.

- **Self Service**—Content, including new reports, should be available on demand as required by the user.
- **Purpose-Built**—The content should be purpose built using design and development tools appropriate for the output. web pages and scorecards are not the same type of content, nor are spreadsheets, cubes or metadata definitions. Each of these should be designed using appropriate tools for the job, not a universal hammer. The productivity gains in using this purpose built approach should far outweigh any concern with learning multiple design paradigms.

In addition to meeting these Information Consumer needs, Reporting Applications must meet scalability, manageability and integration requirements that arise from addressing these large, diverse user populations. To build Reporting Applications, organizations need a platform to provide:

- **Easy Integration**—Integrate easily with existing systems and application via support for standards and extensive APIs.
- **Open Architecture**—Be based on an extensible architecture from a vendor committed to open standards and practices such as open source software development.
- **Information Integration**—Offer, but don't require, the ability to separate data definition, transformation and integration from content preparation. This will allow the organization to isolate and optimize IT skills.
- **Extensible Content Tools**—Offer “no-walls” content creation solutions that can create any application, with no exceptions.
- **Browser-based Delivery**—Deliver web-based content that requires no special software or downloads.
- **High Performance**—Use high performance servers that are always available to handle user information requests and are available 24/7.
- **Manageability**—Include secure web-based administration capabilities for project and user management, short and long term storage, and overall environment operation.
- **Predictability**—Provide a cost conscious method for expansion to serve large consumer communities.

These requirements map to several key features of a Reporting Application platform in the areas of content creation, content manipulation and transformation, content management and security, application integration and information delivery through a scaleable and manageable architecture. The requirements listed in this document should be considered essential in the evaluation and purchase of any Reporting Application platform and any participating vendor should be measured against this yardstick.

Summary of Requirements

A Reporting Application platform must include the following high-level capabilities:

- **User Segmentation and Deployment** capabilities include the ability for a single reporting application platform to cater to the differing needs of the entire enterprise.
- **Metadata and Data Integration** capabilities include the ability to define how data is accessed, integrated and presented to the reporting platform.
- **Reporting Application Design** capabilities describe the requirements for building web-based reporting applications that are adopted by the entire user audience.
- **Reporting Application Deployment** capabilities are the requirements that are necessary to deliver reporting applications to large user communities.
- **Reporting Application Presentation** capabilities describe how the end user is to experience reporting applications.
- **Comprehensive Spreadsheet Management** describes the necessary capabilities in generating server-managed Excel files.
- **Ad-hoc Query and Reporting** capabilities describe the end-user self-service features necessary for adoption.
- **OLAP Analysis** capabilities include the ability to deliver personalized cubes to all analytic consumers.
- **System Management** capabilities include the administrative features required by system administrators to drive the Reporting Application platform software.

User Segmentation and Deployment Requirements

100% adoption of reporting across the enterprise

The platform must be able to cater to the reporting needs of every user. Key to this is recognizing that a “one size fits all” strategy will fail. The platform, therefore, must be applicable across a wide variety of deployments. These should include the ability to embed content generation technology within other applications; support small, departmental or workgroup deployments; support department and division-wide deployments as well as reaching beyond the firewall.

Wide variety of reporting applications

Users needs will vary within these types of deployments. A single platform must be able to support all consumption formats including that for: scorecards (top level managers & subordinates); dashboard portals for all users; printed, static and dynamic internal web pages for all users; high-fidelity, branded customer and partner external portals; server-generated fully functioning spreadsheets for business users; ad hoc query and reporting for business users and individualized OLAP analytics for power users.

Zero training query capabilities

End users should be presented with content in their preferred, readily accessible format; these formats should cater and be specific to the type of user receiving the information, their role within the organization and the applications the recipient regularly uses. End users should not be required to undergo any training to learn how to use a separate tool. For example, operational sales data should be presented as Excel spreadsheets. Content should also be provided over the web and users should be able to interact with it as they would any web page. In the case of web sites where the end-user community is large and dynamic, training is simply not practical.

Pure browser viewing of DHTML content, no plug-ins or applets

The content should be viewable in a pure web browser environment. The use of plug-ins poses deployment problems for web sites with large user bases. Java applets may be time consuming to download and may not function properly through firewalls. Ideally, the content should be delivered as DHTML that can be rendered by pure browsers but also offers content in a high-resolution, attractive format. The DHTML rendering of the content should be consistent across popular browsers. Likewise, the application should avoid requiring plug-ins to client applications such as Excel as this too, introduces unnecessary compatibility headaches.

Flexible output formats

There should be a broad range of output formats supported by the platform. These formats should include DHTML, Excel, RTF, HTML, XML stream, CSV, and PDF for maximum end-user flexibility.

Purpose-built design environments

The platform should include purpose-built construction or design tools for each of the abovementioned consumption formats and the metadata definition that provides data from multiple sources: High fidelity web and printed report designs, spreadsheet designs; ad-hoc report designs; cube designs; metadata designs.

Metadata and Data Integration Requirements

Multi-source metadata layer

The platform must include a multi-layered or hierarchical metadata layer. This layer, in turn must be able to connect or map to a number of disparate data sources. Disparate data sources can be relational databases, live data feeds, XML, multidimensional databases or other applications. Beware of metadata layers that are limited to a single source, or are not able to build transformation hierarchies as they usually will not provide enough flexibility in supporting the needs of the entire enterprise.

In-transit information integration and transformation

The platform must include the ability to retrieve, transform and integrate data from multiple data sources in real-time. This distributed query processing is necessary to provide complete operational perspective that includes information from the data warehouse in addition to operational information from transactional systems.

Caching data within metadata layer

The platform should provide a means of caching data as defined by the metadata to improve overall performance and utility of the reporting environment. This will allow organizations to serve reporting needs without adversely affecting the performance of the system that provides the data. The platform should be able to seamlessly access cached objects as easily as it accesses data via direct queries.

Data-centric design environment

The platform must include a data-centric design environment in which to define the mappings of data sources and to define the metadata views of data drawn from mapped systems.

Common data transformation & integration language

The platform must include a common query language through which transformations across disparate data sources can be performed. This is in addition to its understood ability to perform transformations of data within a single data source, across tables, perhaps.

Hierarchical metadata objects

The metadata definitions must be able to act as data sources themselves, hence enabling the hierarchical layering mentioned above.

Logical metadata representation

The metadata definition should be a virtual (logical) representation of the data and not a physical representation of the information. This virtual definition should be available, and readable in a standard format, such as XML.

Impact analysis of data changes

The metadata definition environment should include the ability to evaluate the impact of change upon the system such as providing a map of components that are descendants and antecedents of a particular column. This will allow the data developer to identify the impact of changing that column.

Drag and drop metadata design environment

The metadata definition environment should include a drag-and drop interface for defining data objects easily and quickly while also providing flexibility and power by including function building tools, SQL copy and paste, and query debugging and preview capabilities.

Extensible data connector framework

The platform should include the ability to extend support for additional or custom data connectors through a series of programming interfaces. These interfaces should also be the way that the vendor implements commercial connectors within their system.

Dynamic-parameter selection via metadata query

The platform should include the ability to define dynamic parameters that populate selection menus in requestor pages based on the results of a query against the metadata.

Reporting Application Design Requirements

Visual development environment

The content development environment should be visual, enabling developers to quickly create content without writing code. For example, the solution should support features like visual query specification, drag-and-drop WYSIWYG layout of pages and visual layout of graphs and charts.

Supports programming language to extend designs

The content development environment should offer a scripting or programming language to extend the visual capabilities of the development environment. The presence of a flexible language ensures that any content can be created, without limitations. An integrated debugging facility should be available to enhance productivity. A choice of programming languages should be offered including Visual Basic, Java and XSLT.

Extensible Component-based paradigm

The content development environment should be component-based to provide productivity benefits when maintaining existing content and creating new content. For example, by creating a library of components used to design multiple sets of pages, developers can maintain multiple designs efficiently by making changes in a single, central location. A component-based approach also aids in developing new content rapidly as new designs can easily be created by using existing components.

Content layout controls, builders and editors

The content development environment should include tools or editors for creating and managing the layout of report content. These editors should include cross-table construction, charts with hyperlinking, expression and date builders and tools to build input pages that include graphical controls such as radio buttons and dynamic drop-down lists.

Post-query data processing and calculations

The content development environment should support the ability to perform any calculations on retrieved data, thus eliminating the need to create temporary objects in either the file system or database.

Create content with any look-and-feel

The development environment should pose no constraints to the format of the content that can be created. This is vital for supporting the complex layout requirements for online statements and summaries. For example, it should be possible to present the data in non-tabular format. It should be possible to lay out objects on a page with fine precision, thus enabling the creation of presentation-quality output.

Leverage familiar design paradigms

The platform should offer multiple content creation tools that cater to the specific skills of the information producer. For example, business users who know Microsoft Office products should be able to apply these skills to create reports using a spreadsheet paradigm and/or MS Access-like interface.

Access any data source, multiple data sources

The content development environment should be able to access any data source including relational databases, XML data, flat files, mainframe systems, Enterprise JavaBeans, COM

objects, .Net applications, etc. Additionally, the environment should support the merging of data from multiple sources before it is presented on a page.

Direct access to SAP Business Warehouse

The content development environment should have specific hooks to SAP's Business Warehouse as a data source, to extend the reach of information managed by SAP. The environment should support access to all BW components including multi-dimensional InfoCubes, the Operational Data Store, and Master Data tables. It also should include an interface to construct MDX queries and support lookups against SAP's role-based security and operate directly against R/3 when necessary.

Direct access to Java and .NET objects

The content development environment should support the ability to directly access Java and .NET objects without writing wrapper code. This direct access to Java and Microsoft .NET objects is necessary to leverage any existing business logic implemented through Java and Microsoft technologies.

Specify XML tags for information

The content development environment should have direct support for creating XML streams that can conform to any Document Type Definition. Developers should be able to assign XML tags to individual objects when designing the content.

Support for actionable content via embedded code

The content development environment should also support the embedding of HTML or JavaScript code within the content. With this capability, the content can be extended to include interactive features such as searching, applets or any arbitrary Web content.

Parameterized control over data access, calculations and formatting

The content creation environment should be flexible enough to enable user-specified parameters to control a) the data that is retrieved and displayed, b) any calculations performed on the data, and c) how the information is formatted. By enabling the parameterization of these aspects, the number of distinct content designs that need to be created is reduced.

Automatically generated user interface

The content development environment should automatically create a parameterized web-based user interface through which users can request new content. This automatically provided user interface should be flexible enough to be offered as stand-alone web pages, or embedded within the content itself.

Reporting Application Deployment Requirements

Choice of content caching options

The platform should provide a choice of options for caching policies and these policies should be granular enough to apply to individual pieces of content. For example, it should be possible to cache content so that the database need not be queried every time a user needs to view the content. This is to ensure that requests for content don't necessarily produce a proliferation of documents. By providing a choice, the system allows for the selection of caching that is optimal for the application requirement.

Automatic content versioning

For explicitly cached content, the system should support automatic version content so that older versions are still available if users need to see information from certain points in time. For example, sales data can be cached on a quarterly basis.

Storage of cached content in flexible hierarchical system

The platform should provide facilities for organizing cached content in a hierarchical manner. The facilities should be flexible; placing no limits on the depth of the hierarchy or on the location of distinct versions of the same cached content within the hierarchy.

Scheduled generation of cached content

The platform should support the creation of cached content on a scheduled basis so that updated information is automatically available on a periodic basis.

Flexible security infrastructure

The platform should provide a security infrastructure that allows for securing both the ability to generate content on-demand and the content itself. The security mechanism should be role-based for easier management. The platform should be flexible and granular. For example, it should be possible to grant different privileges to multiple named users and roles for different versions of the same content.

Leverage existing security services without replication

The platform should support the use of any external user administration systems in order to maintain user security attributes including authentication, authorization, object access and general user attributes. This should be done without replicating the user information across systems. In addition to enabling single sign-on, the use of external security information can also reduce the administrative burden associated with user management.

Secure pages of content without creating multiple documents

The platform should support a manageable and efficient way to secure access to information within the content for individual users. The solution should not result in the creation of multiple documents, each intended for a different user, as this approach creates an added administrative burden.

Full support for printing

The content should be printable and the configuration of the printing should be flexible, supporting printing both on the server-side as well as from the browser with a minimum of clicks from the aforementioned web-based environment. The environment should supply meaningful feedback and let the user interrupt jobs in process.

Direct PDF generation

The platform should be able to directly generate printable PDF documents as an option without generating intermediate formats.

Content available as XML stream

The content should be readily available in XML format to facilitate automatic communication between applications. In addition the system should be able to return an XML stream when it is requested by another application.

Reporting Applications Presentation Requirements

Portal integration

The platform should include its own extensible portal environment for customer convenience. It should also support content presentation to industry standard portal and application server environments. Content delivered to third-party portals should be compliant with industry standards.

Modular, seamless web site integration

The content should lend itself to being seamlessly integrated with the look-and-feel of the web site. This means that the user interface for navigating through the content should be completely customizable in its location, appearance and even existence. Ultimately, the end users should feel that they are viewing just another page on the web site.

Automatic demand paging

For large volumes of information, the system should automatically deliver the content of a page at a time in order to conserve network bandwidth and improve usability. Without demand paging, all the content would be sent over the network in one large block, resulting in excessive network traffic and a slow load time on the browser. The demand paging should occur automatically for all content, without requiring any special configuration.

Free-form hyperlinking into and out of content

End users should be able to freely hyperlink into and out of the content as if it were a standard web page. The content should support dynamic hyperlinking where end users will be taken to different locations based on the actual value of the data in the content.

Powerful content search facilities

End users should have access to features that make it easy for them to navigate through content to find the precise information they are seeking. These features should require no extra implementation effort on the part of content developers. An intelligent searching mechanism that allows them to search for specific information within the content they are viewing is essential. This search mechanism should support multiple search criteria, be context-sensitive in its search, and should also search across multiple related pages, not just the one the user is viewing.

Content navigation facilities

For all content, end users should automatically be presented with navigation facilities that present them with an overall view of the structure of the content. This view should be automatically generated but its content and appearance should be customizable if needed.

Progressive page viewing

End users should experience little to no wait to view multi-page reports. System logic should include the ability to send the first page, or a single page to the requesting user as soon as it is generated, preventing the user from having to wait for the entire report to be generated to begin viewing the results.

Localized content

Users should expect to receive all reporting content localized for their particular context, language and culture. For example, reporting formats should reflect local time zones, date formats, currencies and character set.

Deliver and manage reports in multiple languages using Unicode

The platform should include Unicode support to provide advanced internationalization capabilities to accommodate different languages, cultural contexts, data, and time zones. A single system installation should be able to deliver, manage and create reports in multiple languages. In addition, content created once should be able to be automatically configured for any locale, eliminating the need to redesign and maintain separate reports for each locale.

Layout optimized for browser based viewing

Content should be optimized for browser-based viewing, interaction and navigation. In particular web pages should be created for browser-based, scrollable viewing and not optimized solely for printing conventions. In addition a web page should automatically adjust its height according to the volume of data that needs to be displayed.

Frameless delivery of embedded content

Analytical content should be able to blend seamlessly into any web page or portal without the use of frames. Content developers should be able to extract only the content they need from a larger report set and place it exactly where they want it within an existing set of web pages. End users should be able to consume reporting content as an embedded and seamless part of their unstructured web pages.

Seamless web-based workflow

The browser-based report should also serve as the medium through which users can update data, perform workflow operations and specify preferences for requesting new information, job completion notices and publishing parameters.

Comprehensive Spreadsheet Management Requirements

Support for fully functional Excel content

Excel output should be fully functional and include live graphs, formulas and formatting. With fully functional Excel output, users never have to reformat reports or depend on others for the manual generation of spreadsheet reports.

Query-driven live spreadsheets

Excel output should be assembled from the results of one or many SQL queries or queries against the platform metadata or queries against SAP. The resulting spreadsheet should be assembled with pre-sorted columns, aggregated groupings, live formulas, live charts, conditional formatting or outlining as specified by the creator of the spreadsheet design.

Spreadsheets built by design

Excel output should be driven by a blueprint or design abstraction that helps eliminate formula and formatting errors. This abstraction should be used to generate dynamic spreadsheets that can vary in data set size, complexity and formula definition.

Spreadsheet design environment

The platform should include a spreadsheet design development tool to produce these spreadsheet blueprints.

Use existing spreadsheets as templates for new content design

Existing content should be able to be leveraged as prototypes or templates for the design of new content. For example, report developers should be able to utilize existing Excel reports as the basis for new spreadsheet report creation.

Pivot table definition and generation

Excel output and the associated design, should support the predefinition and generation of Excel Pivot Tables. Ideally, the paradigm used in creating the design abstraction of the pivot table should be similar to creating an actual pivot table in Excel, such that the skill of the user is portable from Excel to the design abstraction.

Dynamic matrix definition and generation

Excel output and the associated design should support the predefinition and generation of dynamic tables and workbooks. A dynamic table means that tables can be created of any size, where the summarization of data in the table is automatically calculated within rows and columns of intersecting dimensions in the table. Imagine a worksheet that displays product sales by quarter and year. The hierarchy could be summarized (added up) at each product category and at each month, quarter and year.

Worksheet bursting within workbook

Excel output and the associated design should support the automatic generation of new worksheets based on predefined rules such as the abovementioned hierarchies or based on rollover rules defined to overcome Excel's 64K row per worksheet limits.

Pre-defined data filters

Excel output and the associated design should support the automatic generation of data filters in worksheets based on the data retrieved from the design's query(ies).

Server managed spreadsheets

Excel output and the associated design should be managed (stored, published, generated and versioned) on a server.

Visual basic macro support within spreadsheet

Excel output and the associated design should include the ability to add pre-recorded Visual Basic macros created by the individual designing the spreadsheet. These macros should be included in the final Excel output delivered to users.

Multi-data sources and queries in worksheets and workbooks

The Excel blueprint design should be able to access and handle data from multiple data sources and place the results of multiple queries in multiple sections within a worksheet and workbook. It should also be able to access the platform's included metadata layer, if included.

Smart worksheet assembly from centralized cache

The Excel blueprint and the resulting output should be able to be cached and parsed at request time in order to minimize the number of spreadsheet designs that must be managed. This kind of caching prior to request should be able to dynamically define new worksheets and workbooks based on the request and permissions of the user.

Ad-Hoc Query and Ad-Hoc Reporting Requirements

Save, share and schedule query results

The platform should offer an ad-hoc query capability for end users to obtain information to which the platform is connected. This should be web-based and driven through a short series of wizard-like pages. Each user should be able to select, scope, filter, and select their preferred results output format. The output of this information should be flexible and support formats such as HTML, Excel, Word (RTF), PDF and any other proprietary formats.

End users should be able to save and publish their queries to the server. Further they should be able to share their queries with other users and schedule the frequency at which the query should run.

Insulate end users from data sources

The platform should insulate the end user who drives the abovementioned query capability from knowledge of the database schema by flattening the table structure and providing business-friendly names for each data field. Further it should allow for application of authentication, data access security, result transformation and rules to scope the depth of queries to eliminate runaway queries, or inappropriate access to the data.

Simple, end-user defined queries and ad-hoc reports

The platform must include the ability for users to define queries and simple reports from a web-based user interface in an ad-hoc manner.

Integrated security for ad-hoc reporting

The platform must support an integrated security model that makes this web-based ad-hoc report development both safe and possible.

Ad-hoc query access to metadata layer

The ad-hoc query definition should be available as an extension of data defined in the abovementioned metadata layer. Ad-hoc reports should inherit this capability (define or refine queries from existing objects of information) and should allow the user to create sub-queries, filter their results and present that information in simple, custom reports.

Flexible output choices for ad-hoc query results

The results of an ad-hoc query should be made available to the user in any format that they prefer, including as flat text, a formatted HTML page or an OLAP Cube. These queries should be re-usable or re-executable by this and other users (if the author permits).

Web and fat client report template development

The platform must provide both web-based and fat-client report development environments. These development paradigms vary because report producer needs vary. Fulfilling the request of a business user seeking to know last quarter's top ten products can be delivered through a web interface. A brochure-quality account portfolio statement typically cannot be developed in a web application due to the iterative nature and complexity of constructing the report.

Common design format

The web-based and fat-client development tools should share a common design format. Ideally, the format should be portable across both development tools, allowing for change from each tool to be read and understood by each tool. Design portability such as this would

allow an IT developer to create master templates, perhaps, for use as a starting point for the web-based tool.

Extensible design format with Java

The common design format should support the use and execution of Java as a means to provide pre-and post data processing within the report.

Progressive, WYSIWYG web construction

The web-based report construction tool should provide a WYSIWYG experience where data, format and progress toward completion are apparent as the user builds the report. This is key because the product should cater to the least sophisticated users in the organization. This philosophy, simplicity over complexity, should be pervasively apparent within the web-based tool.

Web-based report viewing

The platform should include not only the ability to build reports from a web-based interface, but also execute these reports and review their content within a web-based environment. The user should be able to modestly manipulate the report, such as re-sorting columns, without needing the development tool.

Reusable template-driven construction

The web-based construction tool should support reusable libraries of report designs (report templates) and data objects of information, allowing the novice report producer to pick their data object and then apply and modify a design template over that data object.

Integrated security for web reporting

The web-based construction tool should invisibly apply the security policies associated with the report producer.

Robust formatting for web-based report construction

The web-based construction tool should support placement of tabular, listing and charted information in a layout. It should support data filtering, column sorting, grouping and computation of calculated columns. It should automatically paginate, and print in HTML and other formats such as PDF or Word.

OLAP Analysis Requirements

Multi-dimensional OLAP Analytics

The platform should provide for web-based multi-dimensional analytics, referred to as the analytic environment. This must be flexible to cater to the varying analytic needs of different users. The platform must not assume that every user owns a particular set of skills in order to use this analytic environment.

Brand-able OLAP viewing experience

The analytic viewing environment should support the ability to expose and restrict functionality and support definition of specific formats and colors in order to adhere to corporate branding and other aesthetic standards.

View dimensions and measures

The analytic environment must provide the ability to view hierarchical business categories and display a variety of numeric metrics associated with those categories. The environment should provide the ability for the user to set which categories to analyze and at what level of the business category (this hierarchy is often described as a dimension of an OLAP cube).

Drag and drop manipulation of dimensions and measures

The analytic environment should allow the user to easily change categories, measures and levels through a drag-and-drop interface.

Multiple analytic viewing options

The analytic environment should support multiple visualization capabilities for the data that is being analyzed. Common presentation capabilities should include a tabular view with or without summary information, bar charts, pie charts and line graphs.

Save analytic views as reports

The analytic environment should allow the user to save, print and share their present view of a cube such that that view can be akin to a report that can be re-executed as the data is refreshed.

Link to reports or cubes

The analytic environment should allow for customization including the ability to link to other information, such as detail reports or other cubes.

Calculated columns and conditional formatting

The analytic environment should allow the user to add calculated columns as necessary and add conditional formatting to cell values in a table.

Hierarchy and metric data filtering

The analytic environment should support the ability to filter (include and exclude) values for measures and dimensions in order to allow the analyst to zero in on exactly the information needed.

Flexible personalized cubes

The analytic environment should scale to support the consolidation of very large data sets, such as that of one million records or more. It should not overwhelm users with too much

data, however, and avoid being restricted to a few skilled power users. The environment should support personalized cubes that cater to the analytic desires of each individual.

Support large numbers of users

The analytic environment should scale to support large user communities and allow for the users to perform most analysis locally, leveraging local computing resources. This is designed to minimize latency and the requirement to be connected to the home server in order to perform analysis.

Cube definition environment

The analytic environment should include a definition tool that describes the structure of each cube including the query it executes along with the parameters passed, the definition of the OLAP cube's dimensions and measures and the definition of any custom calculations, navigation and transformation.

Customizable viewing environment based on consumer skill level

The analytic environments should allow the individual who defines the cube to set the functionality level of the analysis environment with which consumers will interact with the cube.

Ragged hierarchy support

The analytic cube definition tool should allow for placement of the same data dimension in multiple hierarchies. It should also support "ragged" hierarchies where different levels in a hierarchy can be displayed as peers to other members of that hierarchy and not isolated to members of the same dimension level.

Extensible cube design

The analytic cube definition should be extensible by an IT person or programmer in order to add flexibility in how the cube is defined, perhaps by adding hierarchy definition calculations, or dynamic parameter definition as the query is being passed to the data source.

Content presentation API

The analytic environment should include a content programming interface such that the display of cube information can be presented in portals or other applications.

Platform Services and Architectural Requirements

Comprehensive platform support

The platform should be available on a wide variety of platforms including Windows 2000 Server, HP-UX, IBM AIX, Solaris and Red Hat and SuSE Linux.

Complete installation

The products must install easily and completely. All products, including those that run on servers or local fat-clients should be self-contained and include all required components and runtime libraries in the installation. The only exception to this rule should be the inclusion of each specific database or ODBC driver to which the reporting system connects, as these are often product, version and operating system specific. These drivers should be available from the manufacturer of the data source container.

Platform independent APIs

The platform should provide a rich set of platform independent APIs, through which external applications can directly access functionality related to the following areas: user management via LDAP and Active Directory, content creation (scheduled and on-demand), content viewing and content storage. This platform independent API should be implemented as an XML-based web services API with WSDL descriptions so that the system can be accessed from any platform including J2EE and .NET as well as other programming languages.

URL-based APIs

The content delivery solution should be "URL-driven" to further improve integration with the web. Any content generated by the system should be accessible by issuing the appropriate URL. The platform should also be capable of generating new content on-demand via a URL.

Seamless integration to .NET and J2EE portal environments

The platform should integrate seamlessly to portal environments using native technologies for the portal. This facilitates easy integration, branding and personalization of report content within the custom environment. Specifically the platform should support seamless integration to both Microsoft .NET and those built with the Java 2 Enterprise Edition (J2EE) frameworks.

Clustered architecture

The platform should provide a robust clustering architecture for efficient workload distribution and a highly available system. Applications should be capable of being distributed across multiple machines possibly of varying power and operating systems, enabling maximum leverage of hardware.

Processing scalability

The platform should be scalable, capable of supporting up to a million hits per day per CPU. In addition, it should be architected so that it makes efficient use of additional CPU's with near linear scalability.

Storage scalability

The platform should support the incremental addition of content storage capacity, so that larger volumes of content can be stored without taking the system offline.

Highly available architecture

The platform should be architected so that it has no single point of failure, but use hardware efficiently in doing so. The platform should support active-active fail over as well as active-passive configurations (hot and cold standby servers). The system administrator should have the option to configure the system to respond to failure with either a machine specifically designated for system failure or an active machine that can be additionally utilized to deal with the system failure.

Seamless integration of structured and unstructured data

The platform should provide an interface through which individual elements of content (e.g., a graph or table) from a large document can be embedded in other web pages. This enables the reuse of existing content in order to provide a view of content that is integrated with structured or unstructured web content from other systems.

Ability to generate, distribute and store all types of content

The platform should be open and capable of managing the creation, distribution and storage of content designed with practically any tools.

Automatic content archiving

The platform should support a flexible, policy-based automatic archiving system whereby administrators can specify archiving policies for cached content created by specific users, of a specified format or stored in a given location in the hierarchy.

Push notification to alert users of new content

The platform should support “push” capabilities to alert users when new content is available. Multiple, configurable channels to which users could subscribe provide an effective way to distribute information in a targeted fashion to large numbers of users. The platform should support emailing users to alert them of newly available content.

Remote administration for system-wide control

The platform should provide secure, system-wide control via a web-based management console. This enables a Systems Administrator to manage the platform’s services from any web-enabled desktop or device, making remote administration through firewalls possible. The web-based administration console should provide as much control as a server-based management console.

Email notification and report delivery

The content delivery platform should include the ability to deliver notifications and published report content via email. The SMTP messages should be flexible and include customizable message headers (To: From: Reply-To: Date: Subject:) and include extensible MIME message body parts including file attachments and formatted HTML.

Server-side processing

End-user requested operations like searching should occur on the server-side to maximize performance. It should be possible to index content that is likely to be searched in order to improve search performance.

System monitoring capabilities

The platform should provide monitoring capabilities that enable a system administrator to perform capacity planning, eliminate obsolete reports, check for unauthorized access

attempts, and generally monitor system health. Systems monitoring logs should be able to be customized to meet system administrators' needs.

Log consolidation and centralized reporting

For clustered systems, the platform should provide the ability to consolidate logs of all activities into a central repository from which system operations reports can be run to determine popularity and frequency of report requests, adoption rates and chargeback services.

Support for SNMP monitoring tools

The platform should provide an open SNMP API so that the platform can be monitored using industry-standard tools.

Online system backup

The platform should be capable of being backed up without having to be taken offline.

Migration, upgrade and coexistence

The platform should support migration and upgrade paths from previous versions for non-custom code and reports and allow for current and previous versions of the product to coexist on the host machine.