# Platform Objectives

**SVG** (Scalable Vector Graphic) scales perfectly compared to **Raster** (Gif JPG Png) suffering pixilation when resolution increases on the device displays
Our **objective** is to provide a convenient way to **publish**, **contain** and **broadcast** extensible **SVG** content with graphic navigational ability on devices accessing the web. Its **effective solution** combining technology and methodology **remedies** plenty inadequacies of the current **conventional means** by offering an alternative that also overcomes a mobile devices limitations of local storage by making the **SVG** content externally extensible and interactive

**Problem**

Conventional use of **SVG** in applications and devices have challenges concerning requirement of publishing, containing and broadcasting its **graphic** content. Its greatest limitation is the constraint of having to define static content in **files** preventing dynamic manipulation and needing numerous applications and technical expertise to maintain them. So an integrated **platform** to resolve those issues is required

**Solution**

Our **platform** solves these problems with an effective combination of technology and methodology allowing all users to concentrate effort on delivering ideal **graphics** illustrating concepts in the domain unencumbered by technicalities of employing many processes and utilities to achieve the **objective**. Usage of our well integrated **utility suites** with advised **work flows** maximize productivity and efficiency

The **platform** consists of components which collectively collaborate according to specialized roles to offer **services** of the overall solution.
3 primary concerns of **publishing**, **containing** and **broadcasting** of **SVG** content in **repositories** are accommodated by components
To interact securely all have **secure keys** acting as identifiers on the **platform** which must be confirmed before permitting further interactions.
All aspects of the **repository** get administrated from a central **web portal** facilitating a tool suite provisioning further specialized functionality

**Publish**

Facilitated by utilizing the linguistic model designer by the domain experts and graphic model designer by the graphic designers
**Linguist Model Designer** enables the domain experts without graphics expertise to publish a quick preliminary graphics specification
**Graphic Model Designer** enables the designers with graphics expertise to publish graphics that comply with the specifications stipulated

**Contain**

Facilitated by the **services** of hosting graphics repositories and freely available symbol catalogs constantly having symbols added to
**Graphic Repository** stores : shapes, symbols and text composed and linked in the applicable designers within in the studio app
**Symbol Catalog** stores : symbols in categories ie nature, tech etc ... importable into repositories for inclusion into compositions

**Broadcast**

Facilitated by use of channels which are access points for a specific repository to stream **SVG** content to be consumed by devices
**Web Channel** : outputs **SVG** in a W3C compliant format that is standard and supported by web browsers like Edge, Chrome, Firefox
**Mobile Channel** : outputs **SVG** in a tailored W3C compliant format consumed by the android apps providing touch navigation


Components | Platform Benefits | Solution Benefits | Prove Claim | SVG Utils 1 | SVG Utils 2 | Repository | Methodology
Linguistic Model | Graphic Model | Pub Mechanisms | Roles | Work Attributes | Work Structure | Work Flow | Resources

# Platform Components

Index

| | |
|---|---|
| **Studio Designer** | **SVG** editor managing all the aspects for the publishing of all the **SVG** content in multiple secured repositories |
| **Cataloger** | **SVG** editor which parses **SVG** documents or also traces bitmap images for importing the symbols into catalogs |
| **Symbol Catalog** | Production ready commercial quality **SVG** symbols importable through the web portal or studio app into repositories |
| **Repository** | Secure location storing **SVG** content which is maintained by applicable components collaborating within the platform |
| **Web Channel** | **SVG** content in a repository is trafficked through this secure central point accessible through a registered web site |
| **Mobile Channel** | **SVG** content in a repository is trafficked through this secure central point accessible by the registered mobile app |
| **Android API** | The application programmable interface for communication between the mobile applications and the mobile channels |
| **Starter Kit** | Android code template downloaded from our Git Hub repository freely available for customization to specific preference |
| **Android App** | Mobile application using the template configured with a channel URL and app key for interaction with a mobile channel |
| **Secure Key** | Issued to all components needing secure interactions on the platform which does reject privileged access if it is incorrect |
| **Studio Router** | An authentication, routing and session mechanism required by the studio designer to maintain content in repositories |
| **Trusted Station** | Security measure whereby a work stations identity is registered to the user account for secure usage of a repository |
| **Audit Trail** | All activity on this platform is recorded and accessible by the search facility in the Admin Web Portal and Studio |
| **Admin Console** | A web admin console managing all of the repositories and other relevant services and resources on the platform |
| **Tasking** | Maintains and monitors the projects, in-trays, tasks and notes for users working on the repositories contents |

# Platform Benefits

Index

For an **overview** of the benefits of using this platform instead of numerous systems and adhoc manual processes we mention all significant advantages :

1. **Optimal Integration** - central access to the web portal (admin) and studio app (svg designer) facilitate all the following without requiring any access to physical files, additional utilities and numerous adhoc processes to publish, manage and monitor graphics content and so record change events
   - **Permissioned Access** : role based functionality through the secure admin portal and studio login which determine the utility availability
   - **Automatic Auditing** : records all the system activity in the admin portal and studio app storing pertinent details of all the change events
   - **Tasking Facility** : assigning work loads in projects to in-trays with allocated tasks and notes mapped to the graphic specifications
   - **Change History** : conveniently navigate through session activity with ability to go directly to changed resources in its editor for review
   - **Requirement Profiling** : graphic specifications drafted through text and a labeled sketch which features as a graphic design strategy
   - **Interactive Help** : in the studio app explains tools functionality and highlights relevant components on focus of instructions for orientation
   - **Intuitive Work Flow** : convenient integration of an in-tray, interactive help, design strategy view, designers and a browser in the user-interface
   - **Collaboration** : provides secure collaborative efforts controlled by the restrictive roles and influence of individuals in multiple repositories
2. **Time Efficiency** - configuration, registration and activation of the repository, users, broadcasting channels and interfacing mobile application require under a day to be fully ready for content publishing by an SVG graphic editor, broadcasting by the channels and reception by consuming mobile devices
   - **Repository** : when built and registered all the secure key entries are entered into the platforms registry for providing secure transactions
   - **Users** : the repository owner is by default the repository manager responsible for the administration of users and content in a repository
   - **Workstation** : a workstation profile is registered to every single user account ensuring the most secure usage of the repository user account
   - **Studio** : an SVG graphics editor compiled for a user has its unique app key registered to the user account for secure access to publishing
   - **Channels** : automatic and instantaneous broadcasting on the web channel for websites and mobile channel for the android apps
   - **Android App** : customizable template requiring an app key and channel URL configuration for restricted access to a mobile channel
3. **Graphic Resources** - pool google fonts pack and our symbol catalogs immediately available to be included into any graphics compositions requiring no manual adhoc processes to take the descriptors from different sources in various separate graphic utility apps (regrettably common practice)
   - **Google Fonts** : 2000+ fonts are automatically available through the fonts manager facilitating effortless importing of the manipulable fonts
   - **Symbol Catalogs** : 50 000+ symbols (constantly increasing) can be sent from the central web portal to selected libraries in the repository
4. **Best Practice** - best practices have been implemented by default in the system design along with the facilitation of utilities and templates along with strongly advised work flows and methodology to ensure that best practices are always being adhered to in order to reap from the significant benefits
   - **Complexity Avoidance** : integrated utilities feature with automatic processes simplifying work flows that optimizes productivity and efficiency
   - **Duplication Avoidance** : database design ensures maximum re-use of all the graphics entities and resources to the fullest extent that is possible
   - **Redundancy Avoidance** : design strategy and templates ensure specifications are met accurately and only the correct entries are then submitted
   - **Bloat Avoidance** : excluding irrelevant meta-data and descriptors in the database and content payload broadcasted to all the consuming devices
5. **Review Process** - for quality assurance content is to be saved to design templates during sessions and pass a review cycle before being saved to a repository once they have been approved ensuring clean submissions without constant re-drafting which litters the repository with redundant entries
   - **Symbol Template** : shaped symbols must be saved to templates awaiting review and imported symbols await approval before being committed
   - **Symbology Template** : symbology (graphic composition) must be saved to templates awaiting review before being committed to the repository
   - **Review** : process whereby the design template is viewed in the studio app and the relevant task item and note is marked as attended to

# Solution Benefits

Index

For an **overview** of our platforms solution using virtual **SVG** documents instead of the conventional usage of files we compare both approaches :

1. **Convenient Access** - automatic secure access and change control of the SVG is greatly simplified for optimum convenience of a non technical end user :
   - **Database** : secure access achieved by means of the web portal **(Admin)** login and also the scalable vector graphics editor **(Studio)** login
   - **SVG file** : direct access to the hosting file system on the network or through an ftp client to access it is required when needing to make changes
2. **Structural Integrity** - a non technical user does not need to be aware of or follow any file management convention because they have become obsolete :
   - **Database** : integrity and ordered composition is automatically enforced through a conventional database architecture and design strategy (once)
   - **SVG file** : requires a consistent and deliberate file system management convention for adherence to in order to manage SVG documents (daily)
3. **Lower Maintenance** - optimizes the maximum reuse of all graphic compositions in different contexts eliminating duplication of any resources or effort :
   - **Database** : SQL database principles like normalization whereby data is assigned a key and referenced by it when ever a relationship is required
   - **SVG file** : considerable monitoring and manual labor is required to maintain unenforced internal file path references between all the documents
4. **Economized Storage** - a relational database design for composition is tremendously superior to flat files not accommodating this specific optimization :
   - **Database** : graphic paths are stored as strictly text excluding all meta-data and descriptors therefore resulting in the substantial decrease in size
   - **SVG file** : graphic visual elements like color, gradients and shape are encapsulated in meta-data and descriptors in the files thus increasing size
5. **Economized Payload** - raw data without any irrelevant descriptors or meta-data is stored and only minimally encapsulated when it is to be broadcasted :
   - **Database** : to prevent network congestion and use data-bundles cost effectively a reduced payload of content to consuming devices is essential
   - **SVG file** : a greater payload since indiscriminately containing irrelevant references, meta-data, descriptors and unsupported elements regardles
6. **Search Enhanced** - advantage of relational database intrinsic search capabilities as opposed to the file text scanning on that specific hosting file system :
   - **Database** : internal architecture of primary, foreign keys and indexing is specially designed for optimized referencing or searches through data
   - **SVG file** : exclusively for storing meta-data and descriptors of graphic content and is not optimized for referencing and searching through data
7. **Flexible Portability** - since the non formatted raw path data can be effortlessly transformed into multiple formats with a negligible processing overhead :
   - **Database** : raw data not residing in a file is free from the entanglements of descriptors and meta data thus needs no transformational extraction
   - **SVG file** : data residing in flat files and entanglement with meta-data and descriptors requires more resource intensive processing for extraction
8. **Process Enabling** - conditional and combinational logic is possible in a SQL database and not possible in meta-data and descriptors of SVG documents :
   - **Database** : SQL in a database is a programming language therefore it can directly manipulate content opening up vaster processing possibilities
   - **SVG file** : XML tag descriptor based technology and is not a programming language therefore manipulation is only possible on a manual basis
9. **Integrated Resources** - a vast collection of Fonts and Symbols integrated into the tool suite and database for convenient inclusion in any compositions :
   - **Database** : symbols available in the libraries are compatible and immediately available for re-use and flexible manipulation in any composition
   - **SVG file** : symbols must be procured then reformatted and manually inserted into SVG documents with additional utilities and adhoc processes
10. **Virtual Definition** - composition are virtual in both declaration and instance enhanced with the numerous advantages of relational database capabilities :
    - **Database** : Symbology records are virtual SVG docs and Visual Element records are virtual SVG elements which are manipulable in designers
    - **SVG file** : is defined in a static file and its graphic elements feature as descriptors (xml tags) requiring parsing before manipulation in designers
11. **Configurable Linking** - enabled by computational conditional inclusion or exclusion through links on specific visual elements per context driven basis :
    - **Database** : is possible through the ability of dynamic and highly configurable data selection and join statements provisioned by default in SQL
    - **SVG file** : not possible as the files content is statically hard coded into the file preventing possible optimization of re-usability in other contexts
12. **Dynamic Hierarchy** - a virtual document navigational structure is created and persisted with an infinite depth which is only limited by storage capacity :
    - **Database** : this database schema design allows intermediate virtual linking ( contexts) of the virtual **SVG** documents ( Symbology)
    - **SVG file** : static files need manual editing to wrap a visual element with only a single link directing to a single document thus preventing re-use

To substantiate claims that our **platform** generates minimal **SVG** tags compared to the established competitors generating heavily bloated **xml** files, we created the **SVG** examples in both platforms and compared them directly with the comparisons viewable here. Click and view the source in a browser and notice a very significant difference in byte size. The **bloated** output impacts **data bundle** depletion and **network traffic** congestion. To remedy the negative result is removing unnecessary tags from the **SVG** which is a pure waste of time if it can be avoided

| Source | Logo | File | VS | KByte | Image | Process |
|---|---|---|---|---|---|---|
| Symbology | | Brevity-Symbology.svg | VS | 8 KB | Brevity<br>Say as much as possible with as little as possible | **Path Import** : directly from a catalog in the repository<br>**Font Import** : directly from google fonts pack automatically converting path |
| Ink Scape |  | Brevity-InkScape.svg | VS | 14 KB | Brevity<br>Say as much as possible with as little as possible | **Path Import** : directly from an **SVG** file<br>**Font Import** : install font in OS, refer to it and convert the text into a path |

To visually demonstrate how our **platform** generates virtual **SVG** documents with dynamically re-configurable **linking** determined at **run time**. We provide the below illustration conveying the concept of **contexts** facilitating the dynamic behavior not achievable by static **xml** in **SVG** files The long **alpha numeric** strings of characters are **unique keys** used to identify contexts as they wrap specified visual elements with hyper links to more contexts facilitating a loose linking mechanism to navigate domain imagery. So graphic compositions are re-usable from different perspectives

```
<svg width="240.0" height="155.0" xmlns="http://www.w3.org/2000/svg" >
  <g id="root" transform="scale(1.0)">
    Reduced SVG
    <a href="/rest/contextSVG/b6dec1ad6a314ddfbbb5a0236a5ff7f2/context.svg"/>
      <path d="M . . . . . .Z" id="Ellipse-3bf4376f"/>
    </a>
    <a href="/rest/contextSVG/9e7af801cc324acd864e2225129c6432/context.svg"/>
      <path d="M . . . . . .Z" id="Symbol-89c16e0d"/>
    </a>
    <a href="/rest/contextSVG/f8020d91b72f4d799d3000f868ca5a1f/context.svg"/>
      <path d="M . . . . . .Z" id="Halant-Bold.ttf-eafe88a2"/>
    </a>
    <a href="/rest/contextSVG/31ab0943022640c180d5a4df3440477e/1.0/context.svg"
      <text font-family="Serif" font-size="18" id="9d5e205f">Role</text>
    </a>
  </g>
</svg>
```

At runtime a context wraps configured visual elements with clickable hyper links forwarding to other contexts

Shape

Symbol

Path Text

Plain Text

The **cost effectiveness** of your web hosting data plan, is a real world problem that can be remedied by a stream lined payload of data to ensure advantages of :

- **Speed** : a bloated payload of excessive content will quickly cause network traffic congestion and then slow down the pace of serving the requests
- **Volume** : a bloated payload of excessive content will quickly cause hosting data bundle depletion constantly increasing costs of your hosting plan

For an **overview** of broadcasting the **SVG** content to an exclusive audience of intended recipients we must mention the mechanisms termed channels securely exposing repositories graphic content by dynamically generating **SVG** to be consumed by devices and the compatible android apps namely :

1. **Web Channel** - generates the **SVG** in a W3C format and temporarily embeds links around elements linking to other **virtual** **SVG** documents :
   - **SVG** : the W3C spec format (with exception of filters and transformations partially supported) has sufficient support in all the major browsers
   - **Browser** : consumes **SVG** output in the standard W3C spec format to be interacted with by the mouse in the browsers typical functionality
   - **Web Address** : must be registered with the channel before the **SVG** content is served through specific web sites to a web browser
2. **Mobile Channel** - generates the **SVG** in a W3C format and temporarily embeds links around elements navigating to virtual **SVG** documents :
   - **SVG** : the W3C spec format has good support in the standard android architecture through utilizing the Google Chromium Browser Engine
   - **Mobile App** : consumes the tailored **SVG** output to be interacted with through the standard android component called the web view
   - **App Key** : must be registered with the channel and feature in the request header before any **SVG** content is served to the android app

For an **overview** of an android devices requirement to interface with a mobile channel which broadcasts **SVG** content of a graphics repository the essential software instillation and configuration is indicated by the following list of requirements :

- **Starter Kit** : must use the application template which is freely available to be downloaded from the **Git Hub** repository that can be customized
- **Channel URL** : the template is to be configured with the parameter called **"channel-url"** which specifies the channels central web site address
- **App Key** : the template is to be configured with the parameter called **"app-key"** containing a secure key that is registered to a mobile channel
- **Web View** : the template uses a standard android component called web view which uses the default Google Chromium Browser Engine

For an **overview** of the benefits of the **Studio App** it must be known that its not intended to compete with the more extensive features of the other **SVG** editors but avoids the **bloated content** resulting from inflexible and excessive **XML** they produce along with providing a mechanism for linking them :

1. **Resource Pooling** : integrated with the **Google** fonts collection (2000 +) to facilitate the creation and manipulation of any **font** decorated text
2. **Resource Pooling** : integrated with an extensive **symbol catalogs** (constantly growing) which can easily be imported into a **repository** library
3. **Integrated Tool Suite** : requires no technical skill set and additional utilities to assemble the design materials that will consolidate your composition
4. **Integrated Tool Suite** : **Bitmap** images to **SVG** conversion is integrated merely requiring the industry standard **Po-Trace** exceutable file
5. **Format Flexibility** : the raw non-encapsulated path data in the repositories are transformed by means of the channels within this platform
6. **Storage Economy** : raw path data not encapsulated by further descriptors or meta-data is but a fraction the size of the standard **SVG** documents
7. **Payload Economy** : eliminates the irrelevant and unsupported descriptors making for an optimally streamlined content payload that is not bloated
8. **Payload Economy** : avoids generating excessive meta-data made by other **SVG** editors as configuration aids requiring further manual extraction

# SVG Utilities

Index

In addition to the afore mentioned advantages of the **Studio App** submitting exclusively raw path data for storage and broadcasting to and fro repositories the following utilities for viewing and manipulating **SVG** content in the user interface of the application must also be mentioned briefly :

**Informative Views** that are exposed in the user interface are as follows :

**Role Profile** : Permissions enforce restrictive access to interfaces relevant to the users role so ensuring that they can only execute functionality in the system that aligns with the responsibilities of their mandate. A users influence over the content in a repository can also be restricted to :

- **Isolated** (user can only change content they created)
- **Global** (users can also change content others created)

**Audit Profile** : searchable change history for design sessions can locate specific activity and open the graphic entity in the applicable designer

**Repository Profile** : provide an overview of the repository containing the primary symbology and its secondary visual elements namely that of the :

- **Symbology** : capacity and used storage space
- **Symbols** : capacity and used storage space
- **Text** : capacity and used storage space

**Catalog Manager** : shows the overview of the symbol catalogs and links to the web portal to either transfer or request creation of symbols

**Font Manager** : manages the importing of fonts from the downloadable Google Font Pack which can supply fonts in the symbology designer

**Designer Views** that are exposed in the user interface are as follows :

**Symbol Designer** : facilitates maintenance of symbols for use in compositions (symbology) maintained from within the symbology designer

- **Parsing** : paths **parsed** from **SVG** documents and imported into staging ready for approval for porting into production
- **Tracing** : paths **traced** from silhouette bitmap images and imported into staging for approval for porting into production
- **Shaping** : paths **formed** in a process of shaping (principal form influenced by other forms) immediately available in production
- **Import** : paths **sent** from the symbol catalogs in the web portal directly to specified symbol libraries awaiting approval
- **Approve** : paths **approved** after being selected out of staging (with limited editing) before release into production for usage

**Symbology Designer** : facilitates maintenance of **SVG** visual compositions (cluster of visual elements) which is compromised of the following :

- **Shapes** : created and maintained through a comprehensive SVG designer that can apply a wide range of manipulation and decoration
- **Symbols** : removed or added (shaped or imported) and manipulated along with decoration through a comprehensive suite of designers
- **Texts** : created and maintained through a comprehensive SVG designer that can apply a wide range of manipulation and decoration

**Context Designer** : facilitates the temporary linking of compositions (symbology) through specified visual element to other composition

**Article Designer** : facilitates the temporary linking of articles (substantial body of text) to a composition then style and position them within

**Context Browser** : facilitates browsing of compositions by navigating its hierarchical tree structure menu of expandable parents having child nodes

# Repository

Index

To ensure usage of the **publishing mechanisms** correctly follows the intended **work flows**, we devised their accompanying **methodology** ordering the distinct processes into a logical progression as each depends on the formers results with very clearly defined objectives achieved on their conclusion in the order : **Preliminize** (gathering of the domain material), **Rationalize** (understanding of the domain material) and **Visualize** (presenting all the domain material)

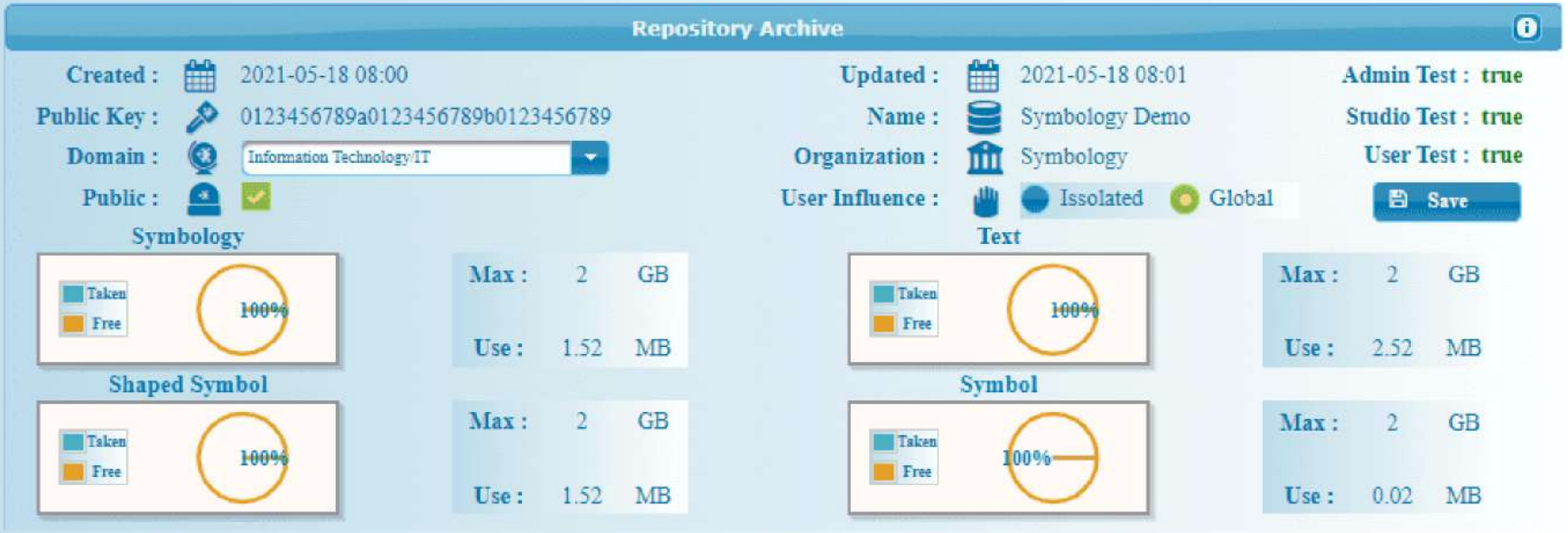
**Repository** : Is essentially a database which has a limited storage capacity therefore serious consideration must always be given to that capacity. Having this fact in mind we advocate adherence to these processes and their underlying work flows for benefit of the repositories storage capacity. Disciplined adherence to these processes has further relevance as the content size grows thus ever more requiring effective management strategies

The repository has internal catalogs which store the **SVG** content in, of which the main graphic entities taking up the most storage space are :

| | |
|---|---|
| **Symbolism** | The graphics specifications with an estimated limit of 100 000 entries at ± 20 KB per detailed sketch to max of 2 GB |
| **Symbology** | The primary graphic entity with an estimated limit of 100 000 entries at ± 20 KB per a thumb tag to the max of 2 GB |
| **Text** | A secondary graphic entity with an estimated limit of 200 000 entries at ± 10 KB per path pattern to the max of 2 GB |
| **Shaped Symbol** | A secondary graphic entity with an estimated limit of 200 000 entries at ± 10 KB per path pattern to the max of 2 GB |
| **Imported Symbol** | A secondary graphic entity with an estimated limit of 200 000 entries at ± 10 KB per path pattern to the max of 2 GB |

To effectively conserve the storage capacity in the repository it is critical that best practices be followed consistently in a users work flow According to the best practice followed within the workflow in the platform, the common pitfalls that you are advised to avoid is the following :

| | |
|---|---|
| **Redundancy** | avoid littering its storage with redundant entries due to redrafts, by defining design strategies and implementing them |
| **Duplication** | reuse all of the graphic entities to their maximum extent through the usage of clones, symbols and contexts |



**Preliminize** : to gather, order and analyze material relevant to subject matter of concepts and constructs in a domain. The individual with ideal skill sets to fulfill all of the obligations of these requirements are the domain expert (having in-depth knowledge regarding the domain) They are responsible for sourcing and maintenance of subject matter material to be in a readily accessible location and format for drafting specs. As a means by which the subject matter is collected, ordered and analyzed is outside the scope of this solution general steps are defined here :

1. **Gather** : identifying all data sources to collect material for subject matter that is relevant to that specific domain
2. **Research** : categorizing and then reorganizing material for subject matter that is relevant to that specific domain
3. **Analyze** : systematically dismantle and dissect material for subject matter that is relevant to that specific domain
4. **Article** : compile dissertations comprehending material for subject matter that is relevant to that specific domain

**Rationalize** : process of drafting specifications for illustrating subject matter of concepts and constructs in a domain. The individuals with ideal skill sets to fulfill all of the obligations of these requirements are the domain expert (having in-depth knowledge regarding the domain) They are responsible for drafting the preliminary graphics specifications providing the design strategy for illustrations of concepts in the domain. This is achieved by means of the tool suite in the web portal where upon accessing any repository the narrative designer is then accessible. On this platform the construct which contains this definition is termed as the linguistic model (provisioning definition through language)

**Linguistic Model** : in this platform further consists of the multitude of narratives which in turn each are comprised of the further multitude of symbolism which it uses to communicate specifications of the imagery that illustrates concepts and constructs within the domain by means of the processes to the right which draft the graphics specifications here which is known as the design strategy on this platform and should be strictly adhered to when composing the final graphic model inside of studio

**Specifying Graphics Specs**

1. create symbolism in narrative
2. note definition of behavior
3. note definition of the imagery
4. do a rough sketch of imagery
5. peg labels of visual elements
6. all results in a design strategy

**Visualize** : process according to a spec graphically illustrates subject matter of the concepts and constructs in the domain. The individuals with the ideal skill sets to fulfill the obligations of these requirements is the graphic designer (having knowledge of the elements of designing) They are responsible for implementing the design strategies of the graphics specifications to visualize concepts and constructs within the domain. This is achieved by a tool suite in a studio app where upon accessing a repository various graphics designers and a browser are accessible. On this platform the construct which contains this definition is termed as the graphic model (provisioning definition through visualization)

**Graphic Model** : in this platform is further comprised of a multitude of contexts which in turn each on their own also envelope their individual symbology which it utilizes to communicate vivid imagery that illustrate concepts and constructs in the domain by means of the processes to the right that implement the graphics specifications so that they conform to the exact requirement by referencing it in the design of their composition and loosely link them for navigation during a design session in the studio application

**Implement Graphics Specs**

1. reference the design strategy
2. create, reuse or get symbols
3. create or reuse a symbology
4. contextualize the symbology
5. link contexts in the designer
6. attach articles in the designer

# Linguistic Model

Index

To deal with the abstract aspects within the platform solution, for convenience of a common language dealing with the system we use specific terminology to define the constructs (definitions persisted) within the framework of those concepts. To compartmentalize responsibilities to fall within the specific role and segregate concerns addressed by specific functionality, the publishing mechanism supported by this platform consists of the tool suite such as the linguistic and graphic designers provided specifically to manipulate the constructs of the linguistic model as well as the graphic model persisted in repositories

**Linguistic Model** *by Domain Expert*

The **Linguistic Model** contains graphic specifications which are to be used as design strategies for creating compositions in the **Graphic Model**
Defines the domain concepts in terms of wording, which in this platform are rationalized through the usage of both the narrative and symbolism
It is **maintained** through the **Linguistic Model designer** in the web **Portal Publisher**.

**Domain** A specific area of interest defined by specialized concepts and constructs that are represented in the imagery contained in a graphic repository

- **Vocabulary** Technical terminology of the domain language (jargon) is to be put into the plain language of laymen terms for easier understanding
- **Semantics** Phrasing the description of intended technical imagery of the domain into more simple wording of precise and concise semantics
- **Article** Substantial body of text that assists as a treatise elaborating on the intended meaning of technical illustrations in the domain imagery

**Narrative** Is a hierarchical containment mechanism that scopes and groups symbolism (specs) under a specific topic which is explored through imagery

**Scope** **Group** **Identify** **Brevity**

- **Subject** Indicates the focal point of the subject matter the symbolism concentrates on in both writing (articles) and illustration (symbology).
- **Reference** Structured Reference translates a symbolisms position in a **hierarchy** into a **directory path notation** for easy referencing

**Symbolism** Is a preliminary graphics specification of the intended imagery, that is to be followed as a definitive design strategy for a graphic composition

- **Link** Visual element appearing in a graphic composition that is temporarily configured in a context to link to another context on clicking
- **Imagery** A description of intended imagery with elaborating descriptions for annotated visual elements or a single structured reference
- **Behavior** Interactive animated behavior can be assigned to specific visual elements featuring in the graphic composition put into a context
- **Sketch** A preliminary sketch of the intended imagery's visual elements and layout usable as the visual guide for creating the composition
- **Element** Visual elements occurring in 3 specific categories namely : shapes, symbol and text is specified within the composition
- **Tagging** Tag sketch with visual elements (name, type & position) that can be annotated with elaborated descriptions in the imagery

**Graphical Model** *by Graphic Designer*

The **Graphic Model** contains all the graphics compositions which implements the design strategies specified in the **Linguistic Model**.
Defines domain concepts in terms of imagery, which is illustrated through the usage of symbology and context which is graphically navigable
It is **maintained** through the **Graphic Model designer** in the **Studio Publisher**.

| | | |
|---|---|---|
| **Context** | | A mechanism facilitating navigation through imagery of the domain by temporarily interlinking compositions through link elements |
| | **Link** | Visual element appearing in a graphic composition that is temporarily configured in a context to link to another context per click |
| | **Background** | Multiple symbology (compositions) can be imported, scaled and positioned as backgrounds for symbology (composition) to reuse |
| | **Article** | Multiple bodies of text can be attached to a symbology (composition) which can be stylized to complement the imagery esthetic |
| | **Behavior** | Visual elements can be decorated with combined filter effects along with assigning event driven animated behavior for dynamics |
| **Symbology** | | Virtual graphic document composed of numerous visual elements clustered in independent re-usable units that can be put in different contexts |
| | **Element** | Visual elements falling within 3 specific categories shape symbol text that is rendered in the canvas of compositions |
| | **Shape** | Archetypal, compound or path shape are used to create forms by means of shaping in the symbol and symbology designer |
| | **Symbol** | shaped or imported symbols conveniently drawn from a freely available resource of symbol catalogs in the platform |
| | **Text** | Native undecorated plain text, font decorated text or shaped text (follows form perimeters) can be put in graphic compositions |

# Organizing the Graphic Model

Index

Both the collections of the linguistic model with its hierarchy of narratives containing symbolism and graphic model with its library of contexts are contained in those collections to logically group them. As the graphic model with its subject matter is the concrete part of the domain represented in a repository its library structure should be logically ordered into a structure that makes sense for grouping of its graphic content. But due to the intention that the contexts should mirror that of the symbolism it does not mean that the organized collection they are grouped in should also mirror one another. Instead the narratives and their symbolism are merely temporary perspectives of the subject matters graphics which it temporarily re-arranges to suit a narrative, so should not dictate the structuring thereof. For the sake of an **intuitive design** it is imperative that the library structure of the contexts **correlates** with that of the symbology libraries. In this manner all the contexts can be found in a predictable location circumventing the necessity for inconvenient searching whilst first trying to find a context for possible re-use to avoid unwanted duplication. As the graphic content in the repository grows this practice becomes more vital for convenient maintenance. It must also be pointed out that when the user thinks of visuals they first see the symbology and then think of a context it can be put into

| Id | Context Library |
|---|---|
| 1 | context-abstract |
| 7 | context-graphic |
| 2 | context-methodology |
| 3 | context-platform |
| 4 | context-service |
| 5 | context-stakeholder |
| 6 | context-tasking |

The **context** libraries should correlate with the **symbology** libraries so that they are found in a predictable location when a symbology is considered

| Id | Symbology Library |
|---|---|
| 1 | symbology-abstract |
| 7 | symbology-graphic |
| 2 | symbology-methodology |
| 3 | symbology-platform |
| 4 | symbology-service |
| 5 | symbology-stakeholder |
| 6 | symbology-tasking |

We observed that there are typical requirements for defining and presenting concepts and constructs within the domain using SVG which are also common to them regardless of their respective industries nature, as there is a typical process it follows for domain experts to describe and designers to visualize.

So we proceeded work on a solution with the realization that when conversing to relate domain specific concepts and constructs, that wording (definition through rationalization) convey explanation generally but illustrating mental imagery (definition through visualization) provides accurate comprehension

The solution thus effectively needs to accommodate a methodology by which to precisely define graphic content specifications, and also that by which to fulfill that graphic content specification which further reveal logical and creative aspects individually dealing with a distinct construct of their own namely that of :

1. **Rationalizing** that deals with the *construct* of a linguistic model which is maintained by the **domain expert**
2. **Visualizing** that deals with the *construct* of a graphic model which is maintained by the **graphic designer**

Once a domain is comprehended through a Linguistic Model, its transposed into a tangible medium of vivid imagery depicted in a Graphic Model.

These constructs essentially pre-determine components of the publishing mechanism, as it needs to structure and navigate explanation and also structure and navigate illustration of that explanation. This requires special utilities that can provide the required functionality on the platform namely that of the :

The **Portal Publisher** which has all the designers and browser to facilitate maintenance and browsing of all content in the linguistic model
The **Linguistic Model** defines concepts and constructs in the domain in terms of words **rationalized** in the platform by the constructs of :

1. **Narrative** (scoping and structuring of the subject matter) requiring a hierarchical mechanism that relates, directs and navigates explanation
2. **Symbolism** (describing and illustrating subject matter) requiring a diagrammatical mechanism which visualizes illustrations of explanation

Thus the **Linguistic Model** is maintained through the :

1. **Narrative Designer** : facilitating the narration through a hierarchical tree menu representing grouping and directional flow of the explanation
2. **Symbolism Designer** : facilitating the symbolism through a diagrammatic sketch labeled by visual elements showing the required illustration

The **Studio Publisher** which has all the designers and a browser to facilitate maintenance and browsing of all content in the graphic model
As the **Graphic Model** depicts concepts and constructs in the domain in terms of imagery **illustrated** in the platform by the constructs of :

1. **Symbology** (reuse-able visual of subject matter) requiring graphic mechanisms that can render a composition with specific visual elements
2. **Context** (navigation of subject matter) requiring a linking mechanism which can temporarily couple compositions through visual elements

Thus the **Graphic Model** is maintained through the :

1. **Symbology Designer** : facilitates composition of graphics through a canvas and menu of visual elements with effects that can be painted on it
2. **Context Designer** : facilitates inter linking by a search and listing that can drag compositions onto a canvas and link elements to compositions

# Roles

Index

To control the relative capacity of a users influence on their repositories. It must be aligned with their station in the scope of the platform.
To affect only what they are responsible for, roles were defined to facilitate privileged access to utilities that functionally support their mandate.
Through this means permissions that allow execution of specific functionality relevant to them, can be allocated to all platform users on a role basis.
This way it controls what they can view and execute in the web portal or studio facilitating all functionality of the publishing mechanisms.

| | |
|---|---|
| **Admin** | Responsible for the administration of the services and builders but can only add users to and view content in a repository |
| **Manager** | Responsible for managing all of the aspects of a repository including the administration of users and access to all the tool suites |
| **Expert** | Responsible for defining and also maintaining the linguistic model of the domain which is contained within a repository |
| **Designer** | Responsible for defining and also maintaining the graphical model of the domain which is contained within a repository |
| **Cataloger** | Responsible for populating and maintaining all the symbols in the symbol catalogs readily available for use in the platform |
| **Developer** | Responsible for developing and maintaining the android application consuming graphics from a repository on the platform |
| **Guest** | Can browse and view the content in the linguistic and graphic models of the repository through the web portal and studio designer |
| **Public** | Access the web channels through registered web sites or mobile channels through registered mobile apps on the platform |

# Work Load Attributes

Index

**Integrated Tasking** manages the **work load** of all users. Is comprised of projects containing tasks that are assigned to the users **in-trays**
The repository managers define and assign tasks to the repository users **in-trays** through the repository console within the web portal
The users can manage their **in-trays** (for multiple repositories) and tasks through the web portal and also in the studio application

**Project** : is the root containment mechanism of the **work load** created first when starting and new increment of development when going through the 3 fundamental processes of the aforementioned **methodology** to deliver their specific goals. Its further comprised of tasks which define the actual work to be done. These then in turn are also further refined in notes which serve as commentary or single item units of work to be delivered

A **Project** is defined by the following properties :

1. **Created** : initially set on the creation of the Project
2. **Updated** : refreshed every time a property is updated
3. **Expected** : target date by which it is to be delivered
4. **Priority** : its overall level of importance for delivery
5. **Status** : to be updated during each stage of progress
6. **Name** : an abbreviation by which it can be referred to
7. **Description** : a general description (refined in tasks)
8. **Tasks** : unit of work (follow the methodical process)

**Priority :**
1. High
2. Moderate
3. Low

**Status :**
1. Planning
2. Info Needed
3. Referring
4. In Progress
5. Postpone
6. Canceled
7. In Review
8. Completed

**Task** : contains the general instructions of the **work load** to be attended by the user and is further refined into more specific instructions of itemized deliverables that are to be attended to

A **Task** is defined by the following properties :

1. **Created** : initially set on creation of the Task
2. **Updated** : refreshed every time it is updated
3. **Expected** : the target date of expected delivered
4. **Priority** : its level of importance for delivery
5. **Status** : to be updated during phases of progress
6. **Name** : a general name it can be referred to as
7. **Description** : general information (refined in notes)
8. **Notes** : a list of task items for attendance

**Notes** : contains the specific instructions of the **work load** that is to be attended to by a **user** and contain either commentary or instruction with references to a model for what is to be delivered

A **Note** is defined by the following properties :

1. **Created** : set on creation
2. **Note** : specific information
3. **Attended** : status indicator

# Work Load Structure

Index

Its considered **best practice** to direct the initial **tasks** of the development process at addressing requirements for the three processes of the **methodology** :
(Refer to the **demo project** example for a correct work load structure and be aware that the next **work flow** chapter refers to this work load structure.

**Prelim-Process Work** : concerned with delivering the **Subject Matter** (too domain centric to define specifics here) Its only predictable definition is the general logistics of the preliminary materials, thus specified by the manager to the expert sufficiently indicated in a **task** and its **notes** :

1. Create a **notes** : Deposit the material in this location and adhere to the directory structure
2. Create a **notes** : The documentation must be in this format ... before it can be transposed into Articles
3. Create a **notes** : Provide images used as preliminaries for the symbols (To : shape or parse or trace)
4. Create a **notes** : Provide images used as preliminaries for the symbology
5. Create a **notes** : Ensure correct access permissions to all the resources are granted to the user

**Rational-Process Work** : concerned with providing a **Linguistic Model** (too domain centric to define specifics here) Its only predictable definition is the narrative and symbolism relationships, thus specified by the manager to the expert sufficiently indicated in a **task** and its **notes** :

1. Create a **task** for the **narratives** (contain notes for every narrative it contains)
2. Create a **notes** for every **narrative** that is to explain a concept in the domain (for attendance)

Keep in mind the following :

1. Ensure the sketches are clear and the labels are positioned.
2. Ensure the imagery contains the correct structural reference and annotated visual elements.
3. Use the symbolism browser to make sure the linking is correct

**Visual-Process Work** : concerned with delivering the **Graphic Model** (established by these proprietary constructs). Its only predictable definition is relationships defined by design strategies, thus specified by an expert to a designer sufficiently indicated in multiple **tasks** and their **notes** :

1. Create a **task** for every **narrative** (contain notes for every symbolism it contains)
2. Create a **notes** for every **symbolism** and request a correlating context (symbologies are implied) (for attendance)

Keep in mind the following :

1. Utilize pre-existing symbology whenever possible to avoid duplication
2. Utilize pre-existing symbols in the public catalog before shaping, parsing or tracing any symbols
3. Utilize symbol and symbology templates during the design sessions and engage the review process before doing a repo save
4. The hierarchy of link elements in the symbolism ultimately translated into the mirroring graphic context
5. Compare the graphic model (browse contexts in studio) with the linguistic model (browse narratives in web portal)

As seamless integration optimizes convenience and thus efficiency we made sure that the tasking, audit and design tools are made readily accessible where relevant. To make optimum use of all tools available to you and benefit from them improving productivity we advise you to use these specific work flows regarded as best practice on the platform. Initially it seems cumbersome but could become second nature as you attend allocated tasks. The benefits of a structured work flow will prove to be worth the effort as the SVG content in the repository grows.

The auditing mechanism is triggered automatically on the platform to record all privileged activity of interest along with the time and user account Thus its a productivity monitoring tool and if wrong doing on the platform occurs it can assist in providing evidence for accountability measures if required As repository managers typically needs to supervise all of the contributors (some temporary) specifically that of the domain expert and the designers, they specify and assign tasks. For the very likely scenario where the repository manager is a domain expert they have the capacity on the platform

The 3 phases of the **methodology** dictates the proposed **life cycle** of the entire collective work flow :

1. **Preliminize** : specified by a manager and attended by an expert - focus on gathering, ordering and analyzing preliminary material for subject matter
2. **Rationalize** : specified by a manager and attended by an expert - focus on explaining and defining the specification (formulating design strategy)
3. **Visualize** : specified by an expert and attended by a designer - focus on graphic compositions for the illustration (implementing design strategy)

The user must access their **in-tray** to monitor and attend their **work load** during their work flow :

1. **Task** : consult the general instruction in the description and change the status accordingly as altered
2. **Notes** : consult the specific instructions in the description and mark them attended when completed

**Phase Preliminize :** we recommend the repository manager follow these steps as their **work flow** :

They should access the in-trays admin in the admin portal and define logistics for the domain material.
They should attend to that task and notes which is stipulated in the **prelim process work**

1. **Gather** material for the subject matter of the domain
2. **Research** material for the subject matter of the domain
3. **Analyze** material for the subject matter of the domain

They should formulate the **rational process work** based on the procured material and determined subject matter

**Phase Rationalize :** we recommend the *domain expert* follow these steps as their **work flow** :

They should access their in-tray in the admin portal for work requesting a linguistic model.
They should attend to the tasks and notes as specified in the **rational process work**

1. The **Linguistic Model** is the very first model to be defined and thus should be the starting process of the work.

    I. In the **Narrative Designer** for each narrative

        1. **Access** the task **note** for definition of the **requirement**
        2. Name and describe the **Narrative**
        3. Define the clustered hierarchy of **Symbolism** and for each :
            a. Describe **symbolism**
            b. Describe **behavior**
            c. Repeat the below for every **symbolism** as per **II**
        4. **Mark** the task **note** as **Attended**

    II. In the **Symbolism Designer** for each symbolism in a specific narrative :

        1. In the ***Context Design Strategy*** position contexts in the relationship diagram
        2. In the ***Symbology Design Strategy*** for each symbolism in the moveable pane
            a. Select link element of parent (notice link element in context diagram change)
            b. Draw the preliminary sketch
            c. Compile list of visual elements type and name
            d. Label the sketch with the visual elements
            e. Describe the annotated visual elements or a single structured reference
            f. The end result is the design strategy

2. The **Linguistic Model** is now ready to be **referenced** in the task and notes to start work on the **Graphic Model**.

They should formulate the **visual process work** based on the resulting narratives and symbolism

**Phase Visualize :** we recommend the ***Graphic Designer*** follow these steps as their **work flow** :

They should access their in-tray in studio for work requesting a graphic model.
They should attend to the tasks and notes as specified in the **visual process work**

1. The **Graphic Model** is the second model to be defined and thus should be the end process of the work.

    - In the **Symbol Designer** refer to the design strategy accessible from the help button

        1. **Access** the **task** for definition of the **requirement**
        2. **Refer** to the mentioned zipped archive of images to be used as preliminary imagery for creation of the symbols
        3. **Create** a template for the **symbol** (only if another cannot be re-used)
        4. **Revise** it through the review process and on approval commit to the repository

    - In the **Symbology Designer** refer to the design strategy accessible from the help button

        1. **Access** the **task** for definition of the **requirement**
        2. **Refer** to the mentioned zipped archive of images to be used as preliminary imagery for creation of the symbology
        3. **Create** a template for the **symbology** (only if another cannot be re-used)
        4. **Add** the shapes, symbols and text
        5. **Revise** it through the review process and on approval commit to the repository

    - In the **Context Designer** refer to the design strategy accessible from the help button

        1. **Access** the **note** for definition of the **requirement**
        2. **Contextualize** each **symbology** if there is no suitable pre-existing one
        3. **Link** the contexts by dragging them from right onto the relevant visual element on the canvas
        4. **Mark** the **note** as **Attended**

    - In the **Article Designer** refer to the design strategy accessible from the help button

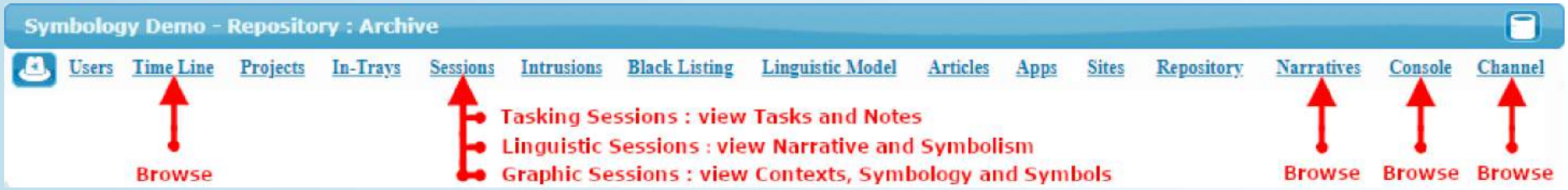
        1. **Access** the **note** for definition of the **requirement**
        2. **Link** the **Article** then position and style it.
        3. **Mark** the task **note** as **Attended**

A comprehensive **demo project** is available for referencing as an introductory guide to acquaint yourself with the tool suite and resources featuring in the platform. As demonstration by example is an **intuitive way to explain concepts**, its beneficial for you to move through its extensive set of examples that clearly demonstrate how the system functionality handles actual data and how the data should be structured according to best practice. The logical reasoning behind the abstractions by way of the **concepts** and underlying **constructs** handled by the solution **methodology** will be clarified as relevant examples demonstrate their practical **purpose**

The **pre-captured sample data** consists of **subject matter** describing the **platform** and its functionality with very basic symbols and text done according to the platforms graphic identity. For **preservation purposes** of its ideal state, the examples sample data **cant be edited** or deleted so they remain in the intended state for **future reference** by all users. The examples do not demonstrate the platforms entire ability for more elaborate graphics manipulation as they are intentionally basic.

For the sake of **simplicity** and **convenience** work session activity capturing all sample data for the **demo project** has been merged under the repository **system user account**. Users can thus search for the **demo project** activity under the repository **system user account** within the **session view** for the tasking, linguistic and graphic session history. For an uncluttered view of session activity (which would happen as users mark off their work in sequence as attended), to provide clarity we opted to set the task and notes status to complete and attended at the very end so that they are viewed neatly in the tasking session view. However **best practice** recommends that the tasks and notes should have their **status changed immediately** as the work is done so that they would accurately reflect the true work progress

There are three primary browsing mechanisms which are used in the web portal to preview three categories of data that is stored and maintained in every **repository** These mechanisms can be accessed from within the main **repository console** once a user has navigated to their organization and located a specific repository within The console navigation bar with links (enabled according to a users role) to all the administrative and linguistic model tools will feature as per the below illustration




The **demo project** examples residing in every **repository** accessed in the **web portal** will give a realistic idea of the repositories domain data presentation To familiarize them self with a superficial overview of what the **repository** domain data would resemble, the user must **browse** the following views namely :

The **project timeline** to preview the **project workload** content for an overview of what a projects tasks and notes data would look like
The **narrative browser** to preview the **linguistic model** content for an overview of what a projects graphic specifications data would look like
The **channel browser** to preview the **graphic model** content for an overview of what a projects graphics compositions data would look like

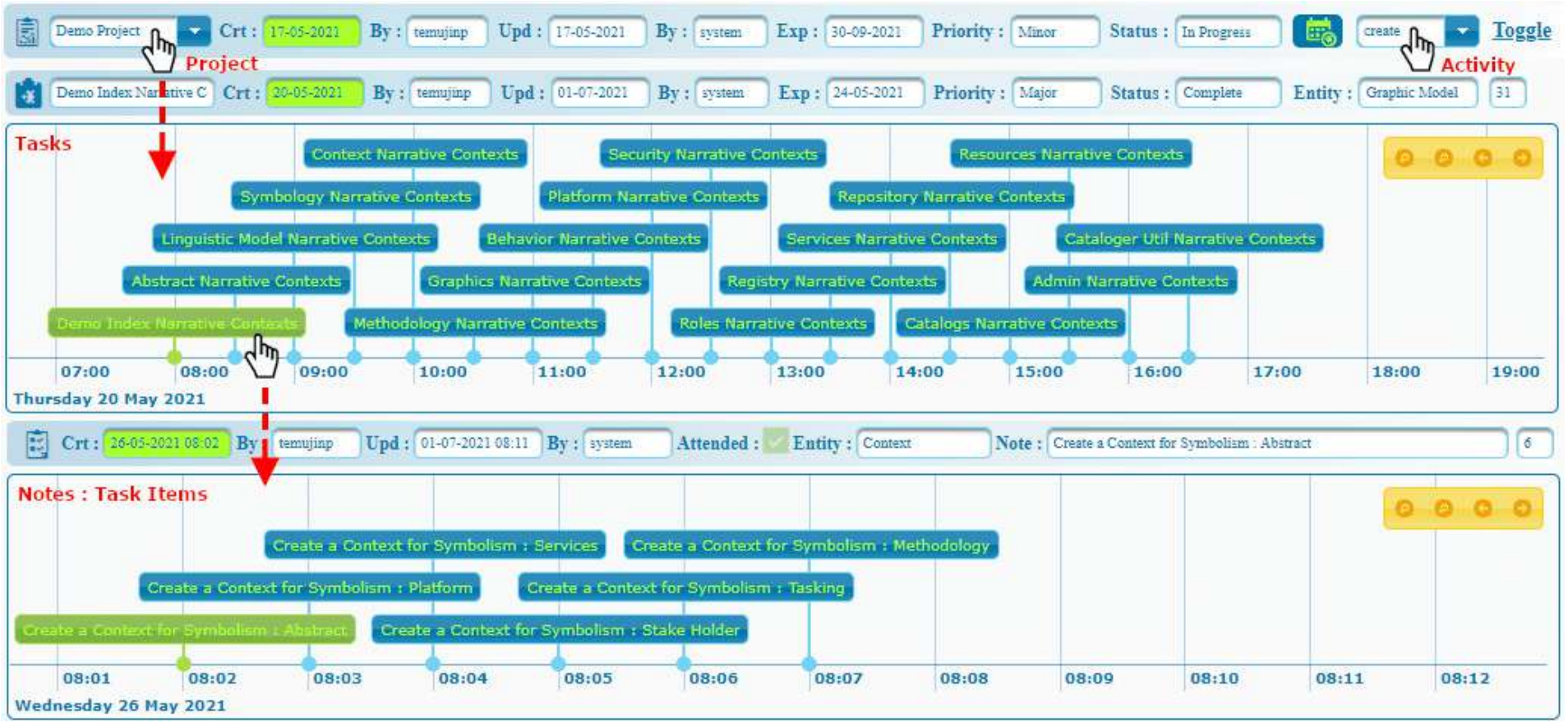
The **demo project** data was created between the date range of **2021-05-19** and **2021-07-02** which can be specified in the **session views** which lists all activity

The **tasking session view** locates the changes made in the **projects** and **in-trays** and provides a preview of the actual changes to the **task** and **notes**
The **linguistic session view** locates the changes made to the linguistic model and graphically preview the actual changes to the **narrative** and **symbolism**
The **graphic session view** locates the changes made to the graphic model and graphically preview the changes to all **context**, **symbology** and **symbols**

# Project Time Line

Index

The project time line facilitates a graphic view of a selected project along with a time line of its tasks and their respective notes. They can be listed according to their created, updated and target delivery dates. On mouse over of the project, task or notes icon their actual description will expand into view. To navigate the timeline to the users choosing, first the project must be selected at top left where upon the tasks time line will be populated. The timeline can then be scrolled to the time desired and it can be zoomed in or out to the users preference. Then secondly the user must click on a task bar in the task timeline to see its notes populated into the lower notes time line and then so also scroll and zoom in or out according to the users preference.




Projects group a collective work load of tasks assigned to user in trays into a manageable batch which can be conveniently maintained and monitored by the repository manager. It is considered best practice on the platform to divide each increment of work into projects which will have their own target date of delivery. The repository manager has access to the repository projects console where they can so define projects, tasks and notes as well as assign them to specific users in-trays. As each user attends to the tasks and their collective notes they must keep their statuses up to date to indicate the work progress as they attend to the tasks. They should be attended to as work is done in the manner indicated by a work flow process previously mentioned.

# Projects

Index

# Repository In-Trays

Index

All users have access to the repository in-trays view which exclusively views each in-tray tasks and their respective notes to monitor a users work attendance. In this manner a user with a smaller workload can easily see another users workload and be able to offer assistance to them if required.

# User In-Trays

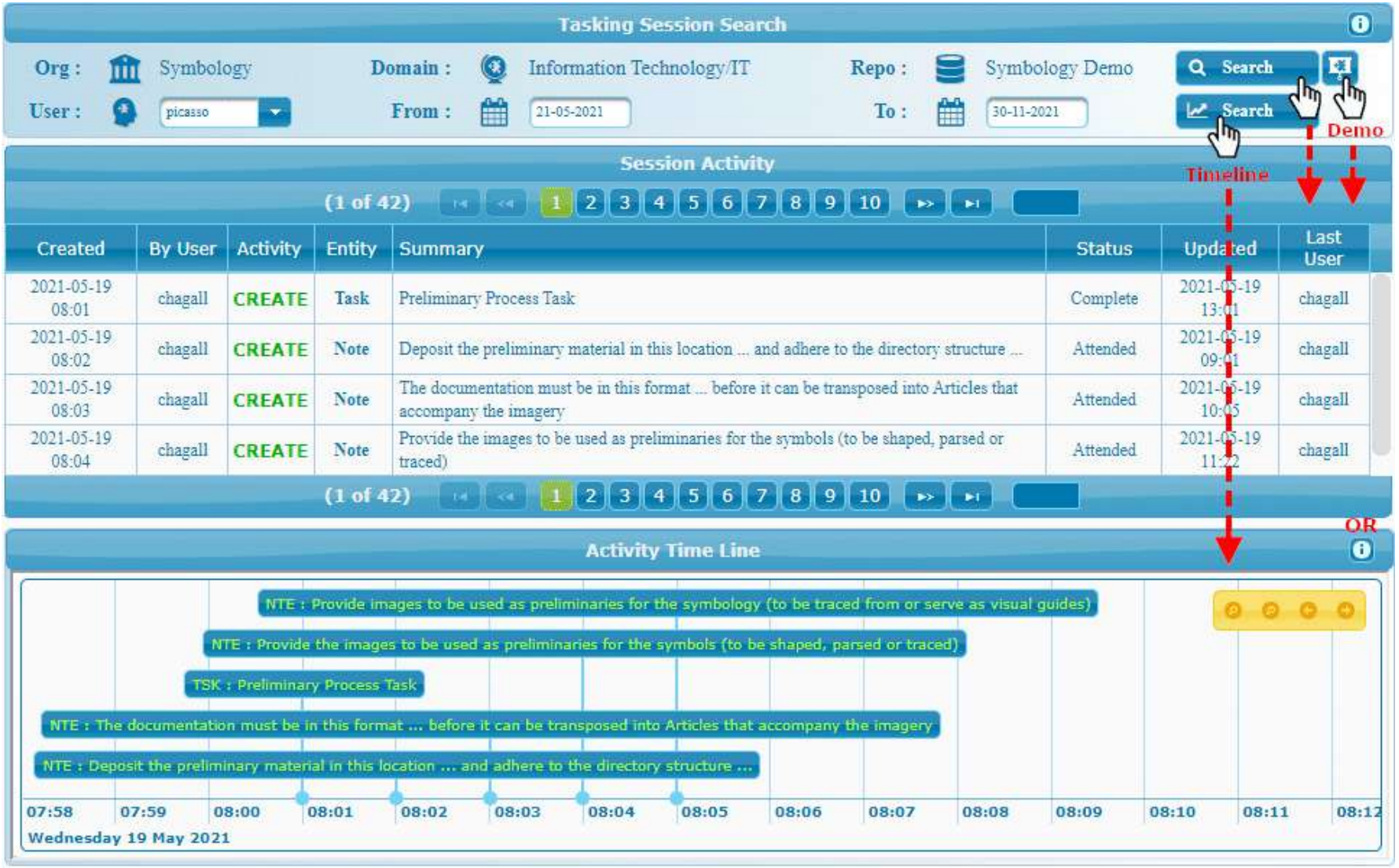
Index

A **user** can have access to multiple **repositories** in multiple capacities performing specific **roles**. They can access their allocated workload in their **in-trays** for the user accounts they have in those repositories. A **user** only has the provision to add more **notes** that provide or request explanations

# Tasking Session

Index

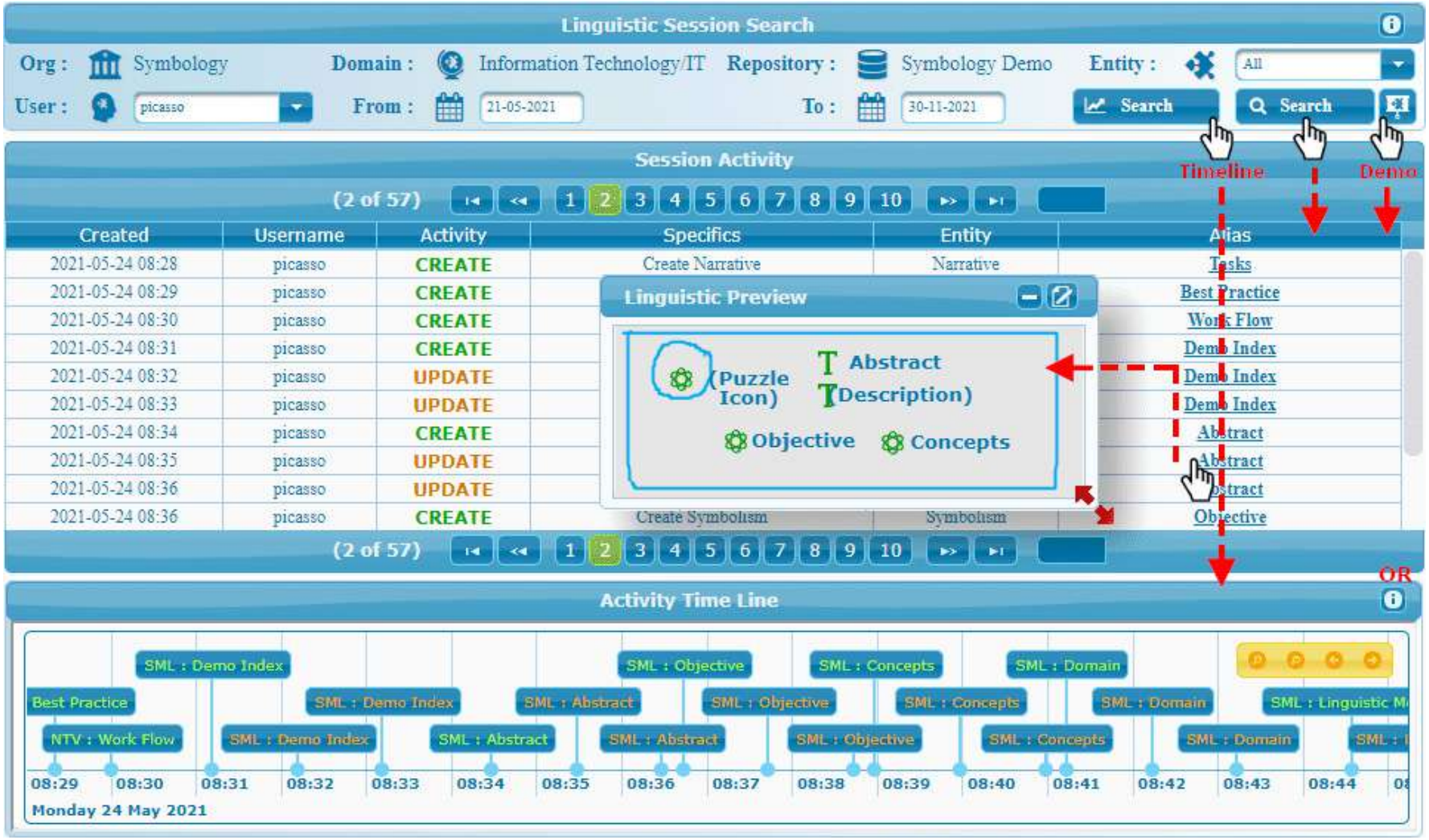
This is an invaluable utility that can assist the repository managers in ascertain according to a time line that is specified by the date range search parameters, what actual work has been done without the need of having to go into each users individual in-tray and determine the work done When either the listing view or the timeline view is selected the other will automatically be hidden from view in the portal web page to clear space.



# Linguistic Session

Index

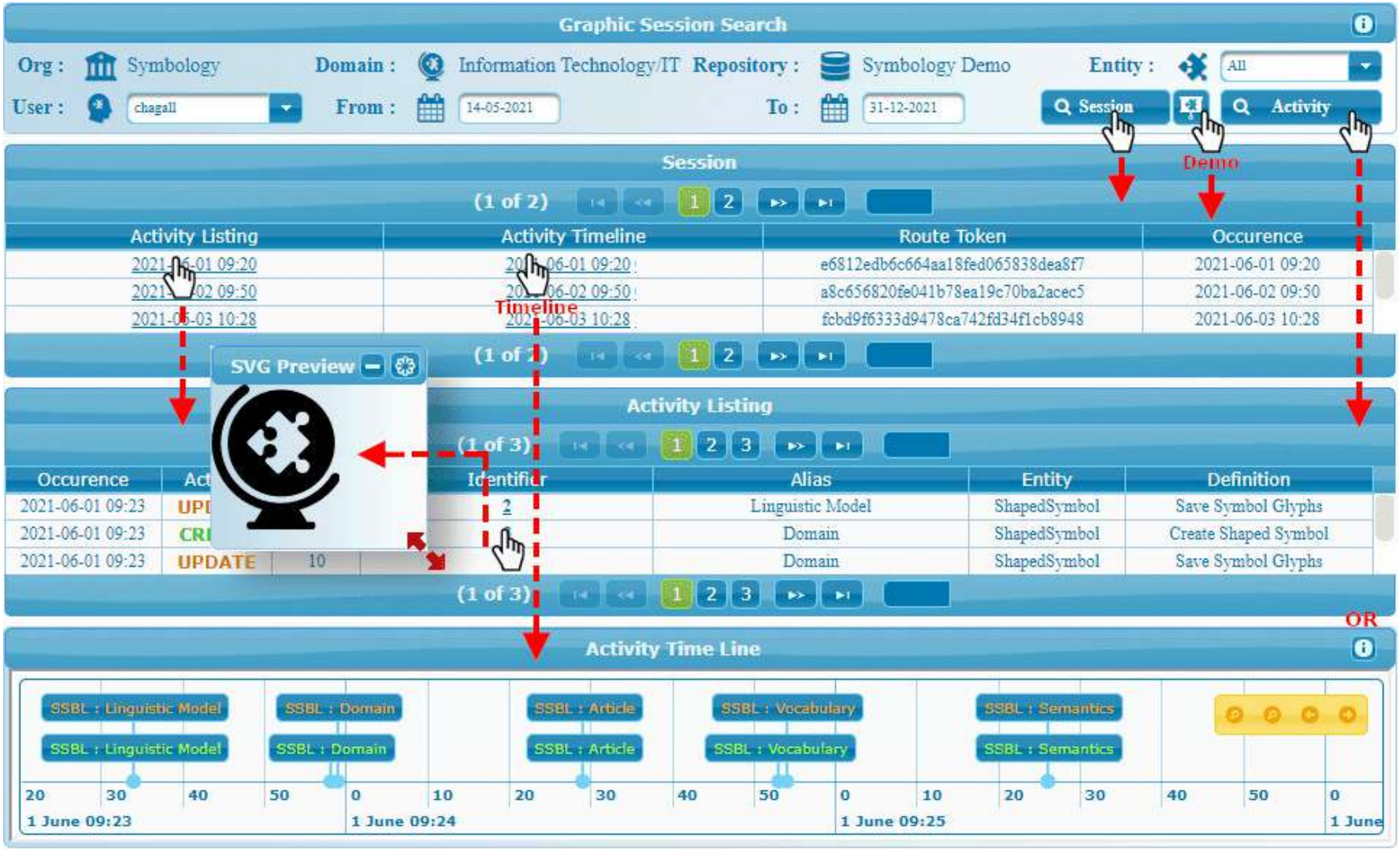
This is an invaluable utility that can assist the repository managers in the review process when they need to go through a users work and review their changes in the linguistic model. All the actual narratives and their symbolism can be navigated to directly according to date of creation or update and then be directly viewed for their evaluation. This view is convenient to only view changes made in the linguistic model. When clicking an "Alias" link it populates the "Linguistic Preview" window with either a **narrative** or **symbolism** diagram to be reviewed. When either the listing view or the timeline view is selected the other will automatically be hidden from view in the portal web page to save space.



# Graphics Session

Index

This utility assists repository managers in a review process when going through a users work to access changes in the graphic model. Clicking the "Identifier" link populates the "SVG Preview" window with **SVG** of the symbols, symbology and context ready for review. The contexts can also be navigated inside the **SVG** preview window to test that all the linking is correct and follows the design strategy. When either the listing view or the timeline view is selected the other will automatically be hidden from view in the portal web page to save space.



# Channel Access

Index

## Mobile Apps accessing Mobile Channels

All mobile apps need to be registered with a repository before they can access a mobile channel and serve as an entry point to the **SVG** of the domain content. In this way the platform can control a repositories data bundle only being depleted by an intended audience from an access point. In this manner access statistics from a particular channel can then also be recorded for future necessity when performing analysis on the data usage.

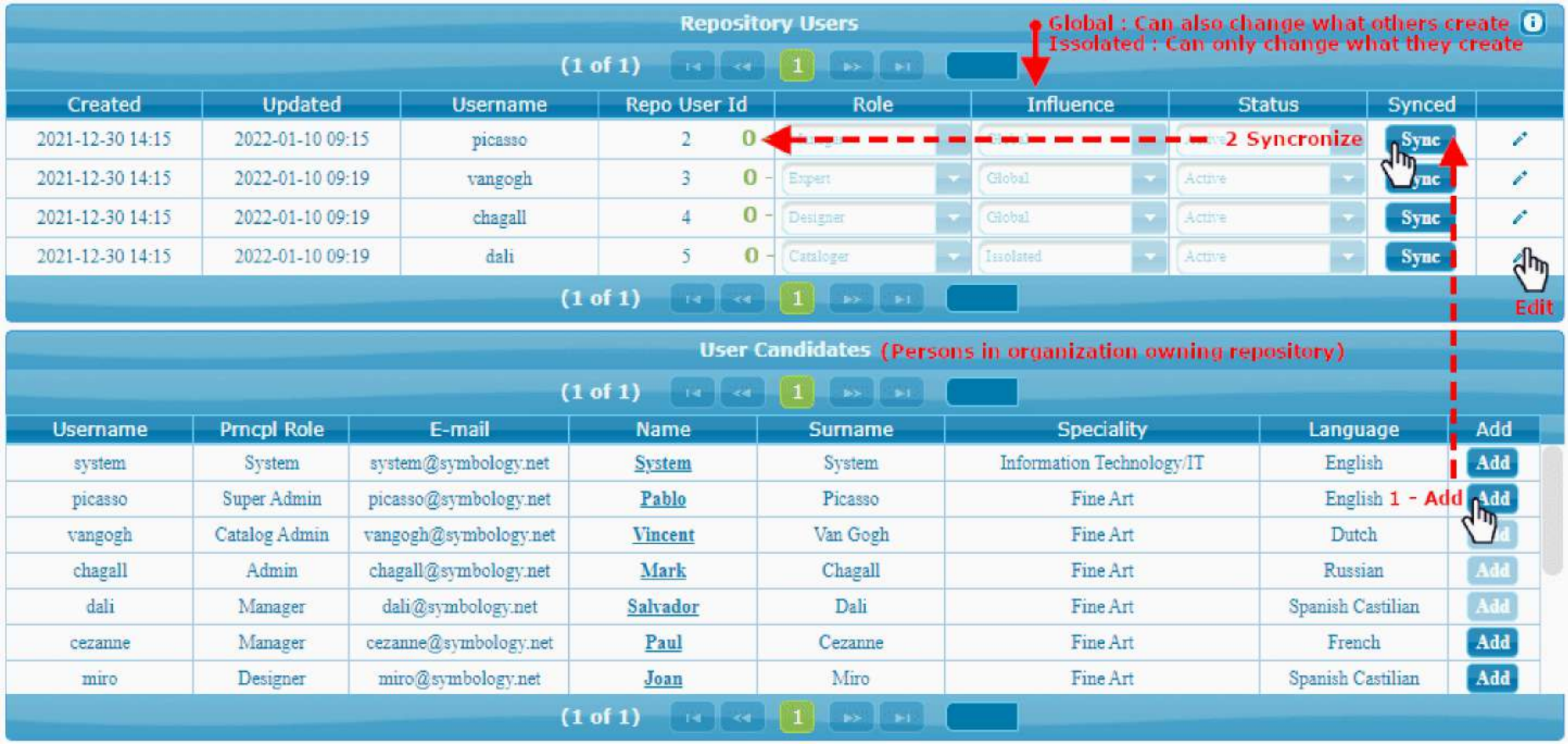

## Web Sites accessing Web Channels

All of the web sites need to be registered with a repository before they can access a web channel and serve as an entry point to the **SVG** of the domain content. In this way the platform can control a repositories data bundle only being depleted by an intended audience from an access point. In this manner access statistics from a particular channel can then also be recorded for future necessity when performing analysis on the data usage.

# Repository Users

Index

Multiple users can have access to a repository and a user can also have access to multiple repositories. Only persons that are members of the organization the repository is registered under are eligible for access to a repository. A user has got only one role in the scope of a repository. The seniority of the roles within a repository are descending as follows : manager, expert, designer, cataloger, developer and guest.



# Repository Intrusions

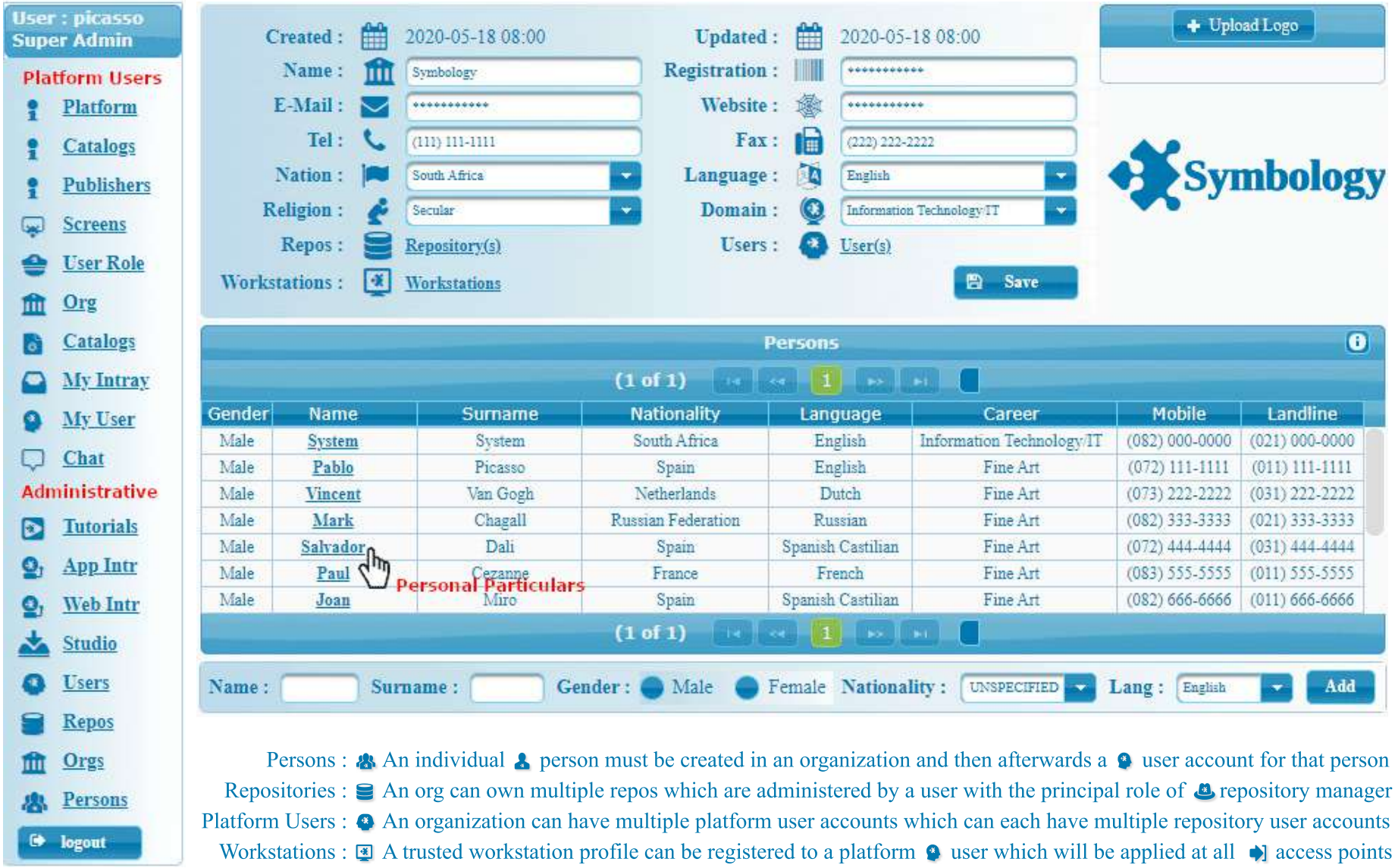
Index

In order to help with investigation of an attempt to compromise a user account by an imposter trying to gain illegal access, every login is recorded. A specific attempt at accessing a user account can be located by searching with the below depicted parameters to aim in on a potential instance. As each studio svg editor has a unique secure key that is register to a specific user account, its easy to locate failed login attempts for specific user accounts

# Organization

Index

An organization is the primary entity on the platform and its particulars are maintained from the org view in the web portal
When an individual logs in to the web portal they have a link to their left from where they can access the organization view



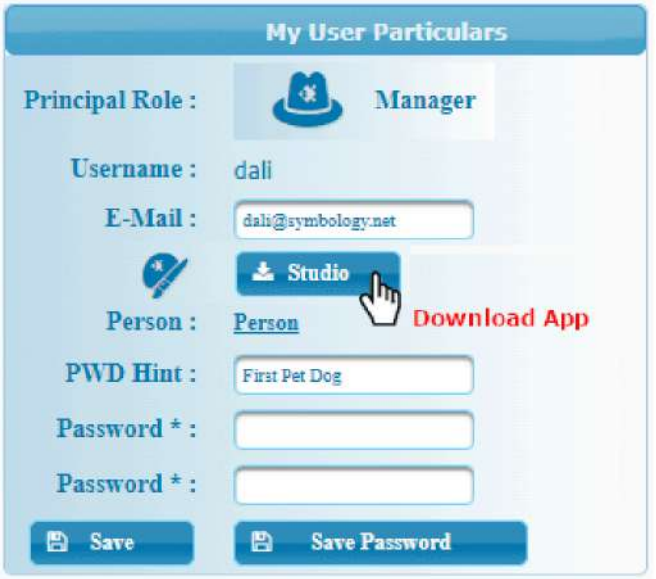

Persons : An individual person must be created in an organization and then afterwards a user account for that person
Repositories : An org can own multiple repos which are administered by a user with the principal role of repository manager
Platform Users : An organization can have multiple platform user accounts which can each have multiple repository user accounts
Workstations : A trusted workstation profile can be registered to a platform user which will be applied at all access points

# Person and User

Index

A personal profile must be created for a working individual to utilize services in the platform


A user account must be created for a person before they can work in repositories on the platform

# Repositories and Users

Index

Organizations can own multiple repositories administered by repo managers with its content maintained by repository users in specialized roles


A principal user account is created for every individual person in the organization which can then have multiple user accounts in repositories

# Trusted Workstation

Index

The security measure of a trusted workstation that must be registered to a user account provides an additional layer an intruder must contend with. The new user or a user with a suspected attempt at compromising an account should coordinate a new login with their repository manager. The repository manager must first set the bypass and request the individual to login, then search for the login and locate the desired workstation and then trust it.

# Resources

Index

3 types of resources are made readily available on the platform which are integrated into its tool suite for convenient usage
**Google Fonts Pack** : fonts collected off the internet and then compiled into a pack which they made available for free on **Github**
**Symbols Catalog** : extensive set of symbols procured from many sources and constantly being added to and is also available for free
**Symbol Templates** : shaped symbol templates re-used interchangeably in repositories and thus be shared through the community

The **Google Font Pack** of + 2000 TTF (True Type Fonts) which is an outline font standard developed by Apple in the late 1980s as their competitor to Adobe's Type 1 fonts used in PostScript. It is the most common format for fonts on Linux, Mac and MS Windows. All the fonts are under open license but is not of concern as you don't distribute them and use it in the studio app (graphic editor) to parse the fonts and generate the shapes from them. The fonts pack is downloadable from **Googles** - **Github** : **https://github.com/google/fonts**

The **Symbol Catalog** contains + 50 000 symbols of things covering a wide variety of subject matter like - technology, commerce and nature etc ... constantly being extended by our catalogers. All symbols are in a path format stored within the database and can be sent from a specific catalog in this web portal to any repository the user can access. The symbol catalogs can be accessed from the left catalogs link. There catalogs are browsed through at right and a specific symbol library in a repository is selectable from at left

**Symbols** can be created through a process of shaping them in a designer whereby an initial shape can be created and other shapes can assert an influence over it such as add, subtract, exclude or intersect their own area. All these relationships can be contained in a portable template persisted in an industry standard format called JSON (Java Script Object Notation) residing in text annotated by curl braces The studio app can export and import these into compositions. It is advised to save to one before finally committing after approval

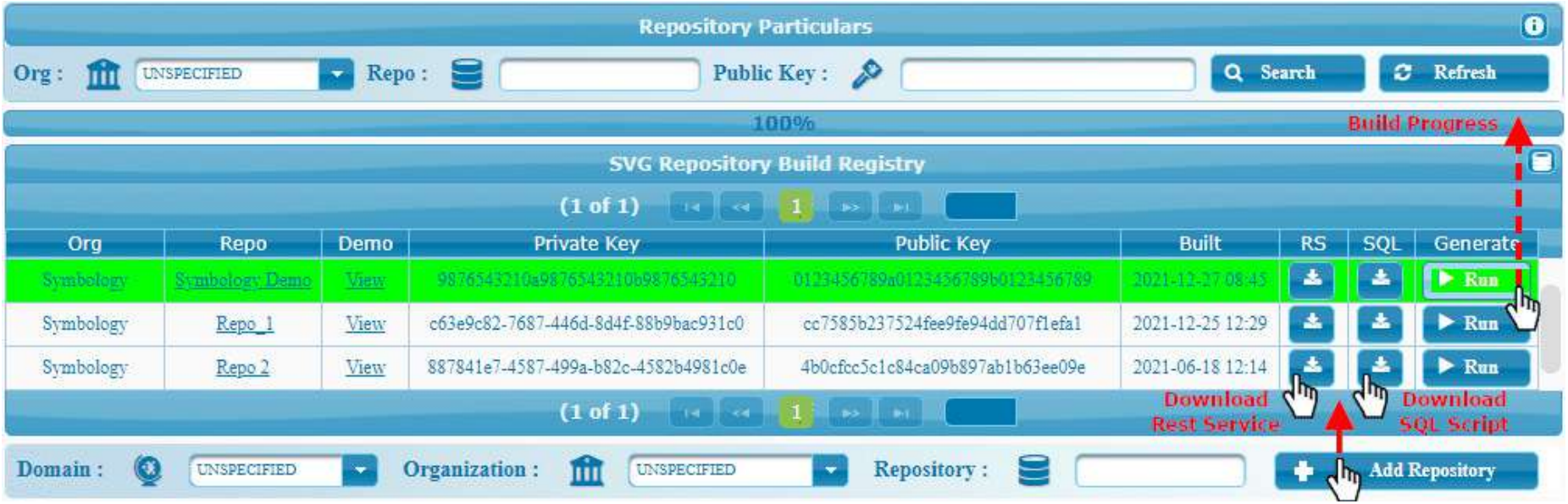
**Symbology** can be created through a process of adding and editing shapes, text and symbols on a canvas. However references to symbols are unique to each repository having their own libraries of symbols. All these relationships can be contained within a portable template persisted in an industry standard format called JSON (Java Script Object Notation) residing in text annotated by curl braces The studio app can export and import these into compositions. It is advised to save to one before finally committing after approval

**Animation** can be created through a process of adding and editing behavior like decorating with filters and event driven animations. However references to visual elements are unique to a composition so not re-usable. All the relationships are contained within a portable template persisted in an industry standard format called JSON (Java Script Object Notation) residing in text annotated by curl braces The studio app can export and import these into animations. It is advised to save to one before finally committing after approval

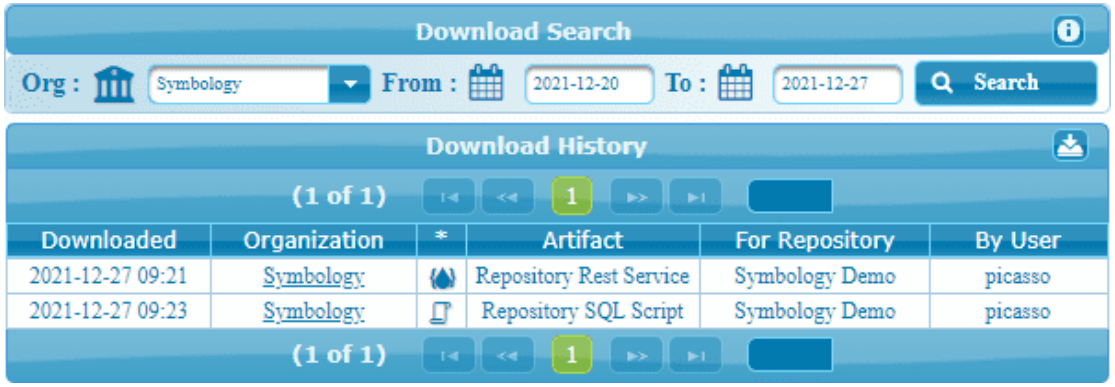
# Repository Registry

Index

The repository registry facilitates the generation of a repository consisting of a configured template which consists of a rest service that is bound to a database and configured with specific secure keys used to identify it on the platform and allow interaction with the web portal and studio apps For obvious security reasons strictly repository administrators and not repository managers can create repositories and run repository builds on the platform. This is only possible for an already preexisting repository profile on the platform already having an organization that it is registered to Once a rest service template has been configured through the build its corresponding database must also be created through running a template SQL script which creates a demo project containing practical examples demonstrating the data structure and platform functionality in the publication mechanisms




All the downloads of the repositories artifacts namely the Rest Services and SQL Scripts are logged for administrative and security reasons.



# Cataloger App and Studio App Registry

Index

The cataloger registry facilitates the generation of a cataloger app and registers the app with the user account for the individual it is generated for. An app key is embedded inside the application which is generated a new for the user account every time it is run and is required at login on the platform. If the app key is not registered with the user account a login is rejected and its failure type is registered in the intrusion logs ensuring that the app can only be used by the person that it is intended for and assists an investigation into a possible attempt at compromising the user account.


The studio registry facilitates the generation of a studio app and registers the app with the user account for the individual person it is generated for. An app key is embedded inside the application which is generated a new for the user account every time it is run and is required at login on the platform. If the app key is not registered with the user account a login is rejected and its failure type is registered in the intrusion logs ensuring that the app can only be used by the person that it is intended for and assists an investigation into a possible attempt at compromising the user account.

# Studio App Permissions

Index

The studio applications roles configuration utility provides a means by which to specify specialized role based permissions that facilitates refinement of privileged access to utilities that functionally support the responsibilities of their mandate. So doing it can dictate which views can be accessed along with which features are enabled in those exact views. After the user has done a login on the platform and they have selected the repository they wish to work on, these configuration settings will be enforced within the applications user interface and the underlying client to server communication