Junction Networks

Bulk Create

Technical Specifications v1.2

Martin Constantine
6/22/2010

Contents

Introduction

This document contains details on the implementation of Bulk Create.

Change Log

Date	Version	Change	Author
06/21/2010	0.1	Initial draft	Martin Constantine
06/29/2010	1.0	UI update and service call updates	Martin Constantine

8/16/2010	1.1	Added notes on Excel macro and installers	Martin Constantine
9/19/2010	1.2	Updated installer notes.	Martin Constantine

Functional Overview

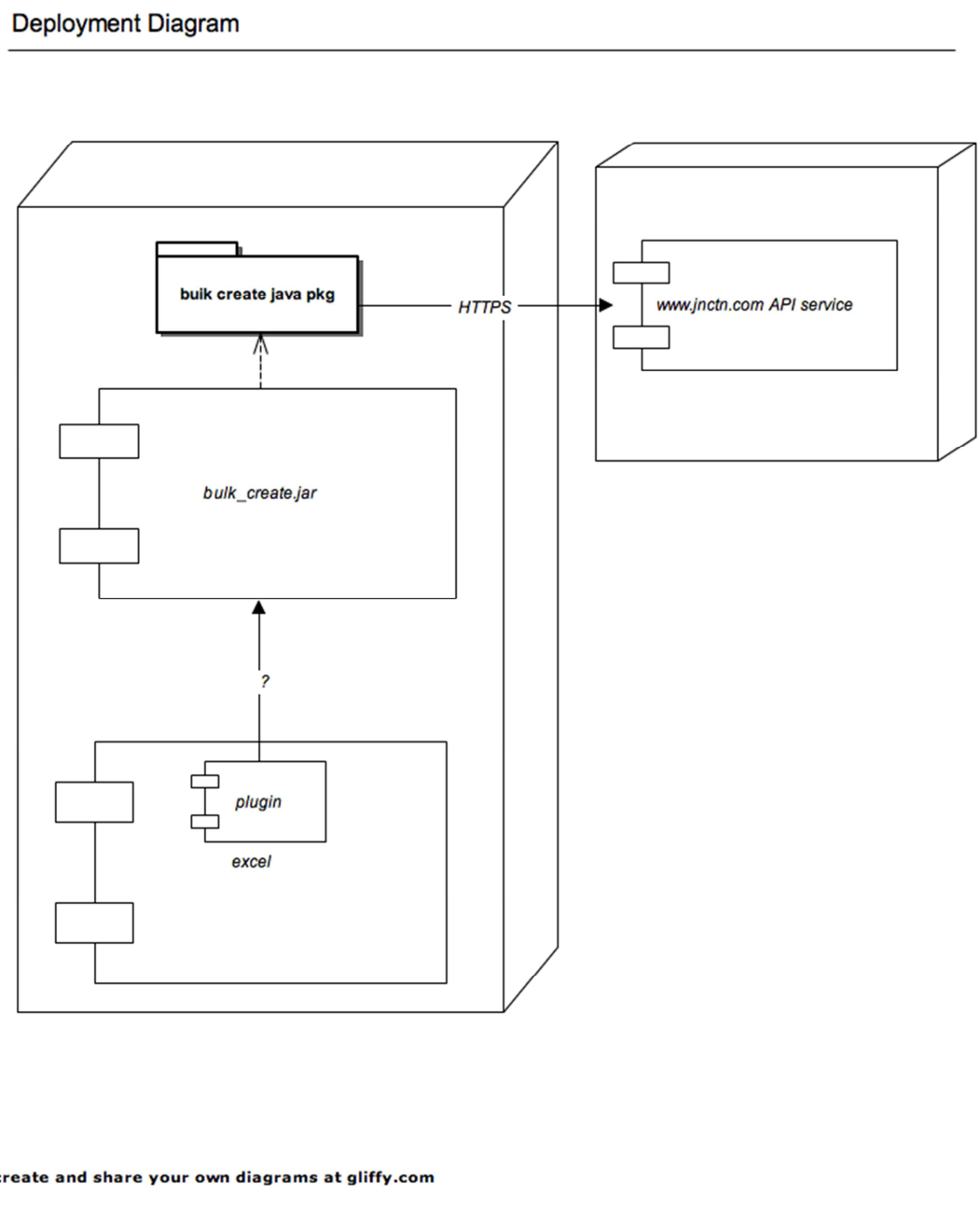
The goal of the project is to provide a streamlined approach to adding new users to the system. Currently, a web service exists that allows the addition of a single user. This project aims to leverage that service in batch mode. The project will provide a portable desktop tool that handles the core of the bulk creation as well as a means of launching that tool via Microsoft Excel.

Technologies and Versions

Application Layer	Technology	Implementation	Version
Front End	Programming Language	Java	1.6.x
Front End	Programming Language	VBA Script	Excel 2007/2010
Front End	UI	Java Swing	1.6.x
Front End	UI Launcher	Excel	2007/2010
Service	Webservice client	JSON Simple	1.1
All	Versioning	Git	

Architecture

The project architecture is best described by the following diagram (provided by Junction Networks):



Everything above the Excel module will be handled by a Java desktop GUI application. The application itself will be layered using an MVC paradigm. View and controller will be handled by the Swing library controls and event listeners. UI controllers will delegate to service classes which will provide access to the model (request and response objects for the webservice).

Data Model

The data model will consist of mostly Java POJOs representing web service request and response messages. These objects will be dynamically generated using a tool such as wsimport. There will be no

persistent data stored on the user's workstation, except for perhaps application preferences (not currently specified).

The other objects making up the data model, will be standard Swing components for the Java GUI.

User Object

New users will have certain fields auto-generated based on the input of the CSV row. Here's the algorithm for creating the auth_username and username fields respectively:

Auth_username:

1) Get the username portion of the organization's Onsip domain.
2) Get the username portion of the current user's email address.
3) Concatenate the two pieces together with an underscore
4) Check that the auth_username is less than 32 chars and matches this regex:
   /^[a-z]([a-z0-9_])*[a-z0-9]$/
5) Otherwise generate a random pronounceable username w/ the algorithm below.

Username

1) Get the username portion of the current user's email address
2) Try to match it to this regex by replacing non matching chars with underscores and stripping trailing/leading white space: **/^[a-z][_a-z0-9-]*(\.[_a-z0-9-]+)*$/**
3) If it doesn't match the regex or it's greater than 32 chars then just try 'user'

Here's some pseudocode:

————gen_authusername() domain_portion = (foo.onsip.com).split('.').first # 'foo' user_portion = (bob.smith@foo.com).split('@').first.replace(/[^a-z0-9_]/,'') # 'bob_smith' auth_username = domain_portion + "" + user_portion # 'foo_bob_smith'

if !(auth_username not match /^[a-z]([a-z0-9_])*[a-z0-9]$/ && authusername.size <= 32) { auth_username = random_pronounceable_username }

return auth_username

————- gen_username() username = (bob.smith@foo.com).split('@').first # 'bob_smith' username.strip(leading/trailing [white space or non alph chars]) username.toLowerCase

if username matches /^[a-z][_a-z0-9-]*(\.[_a-z0-9-]+)*$/ && username <= 32 chars return username

return ‘user’

————generate_users() foreach user { auth_username = gen_authusername() username = gen_username()

...

suffix = 1 while (!user) user = user_add(auth_username => foo_bob_smith, username => bob_smith, ...) if (!user && error is from auth_username or username) { ++suffix auth_username = auth_username + suffix username = username + suffix } }

————- this is in ruby:

def random_pronounceable_name c = [b c d f g h j k l m n p qu r s t v w ch cr fr nd ng nk nt ph pr rd sh sl sp st th tr] v = [a e i o u y] f, r = true, ’’ 8.times do r << (f ? c[rand * c.size] : v[rand * v.size]) f = !f end r end

Java Desktop Application

The core of the project will be a Java desktop application built with Swing. The application will be packaged with platform-specific installers.

GUI Form

The GUI will consist of a single view with four input components and two action buttons. The input components are username text field, password text field, domain text field, and file chooser. The buttons are upload and cancel. Event handlers will be added to each button for the appropriate task and these tasks will be abstracted to the service layer. The credentials supplied to username, password and domain fields are used for authenticating a user with the *Account Admin* privilege. The file chooser is used to provide the CSV input file.

The GUI will be built using the NetBeans IDE Matisse plugin, which allows rapid form design and layout functionality via an intuitive and powerful interface.

Input File Parsing

The input file is expected to be CSV format with the following columns:

First Name,Last Name,Extension,Email Address,Voicemail (Y/N)

This file will be parsed using core Java string handling utilities like StringTokenizer. Parsing errors will result in a popup to the user indicating such. A report of the total amount of items processed will be generated, along with any erroneous rows. Validation will be performed on each row as follows:

1) Extension must be a valid number.

2) Email address must be valid.

3) Voicemail must have 'Y' or 'N' (case insensitive).

Service Layer (Webservice Interaction)

A RESTful webservice client will be created for interacting with the Junction Networks services. This client will have to be hand-crafted as follows:

1. Create model objects from request and response XSDs.
2. Create service interface based on documented operations.
3. Create communication layer for sending requests and receiving responses.
4. Create marshalling/unmarshalling framework for JSON messages.

This will form the basis of the service layer. The interface created in item 2 above will be leveraged from the front-end. The Junction Networks services used by this application are:

1. SessionCreate—creates an authenticated session and returns a token for use in future calls. See http://www.jnctn.com/webservices/api/action.SessionCreate.html for more details.
2. OrganizationRead--queries the OrganizationId from the domain. See http://www.jnctn.com/webservices/api/action.OrganizationRead.html for more details.
3. UserAdd—adds a user to the system. See http://www.jnctn.com/webservices/api/action.UserAdd.html for more details.
4. UserAliasAdd—adds an extension for a user. See http://www.jnctn.com/webservices/api/action.UserAliasAdd.html for more details.
5. VoicemailboxAdd—used to add a voicemail box. See http://www.jnctn.com/webservices/api/action.VoicemailboxAdd.html for more details.
6. UserAddressBrowse--Used to retrieve a UserAddress object representing the address of a user (e.g. the one added via UserAdd). See http://www.jnctn.com/webservices/api/action.UserAddressBrowse.html
7. UserAddressEdit—edits a UserAddress object. See http://www.jnctn.com/webservices/api/action.UserAddressEdit.html for more details.

Data Processing Flow

This is the flow of data as it relates to a single user record. Decision points are based on the existence and/or value of certain fields in the user record. For example a user record with Voicemail set to N will cause the code to not call the VoicemailboxAdd method.

get security token--SessionCreate.

add user--UserAdd

add extension--UserAliasAdd

add vm box--VoicemailoboxAdd

lookup user address--UserAddressBrowse

link user to vm--UserAddressEdit.

Excel Plug-in

A plug-in will be developed for Microsoft Excel that will enable the user to launch the desktop GUI in order to provide input for the upload. The plug-in will be provided in the form of an Excel spreadsheet that contains a programmed VBA script. This script will be developed in a Windows environment and tested across the supported platforms and Excel versions. As of Office 2008, MAC support for VBA scripting in Excel documents was dropped. This presents a risk that the spreadsheet may not work on the MAC. Should this be the case, an alternative implementation strategy will have to apply for that platform.

Windows

A VBA macro was written to launch the Java program from within MS Excel 2007/2010. The entry point is a simple spreadsheet containing a sample row. To run the macro, follow the steps below:

1) Launch the spreadsheet by double-clicking it.
2) Excel will likely give a macro security prompt. Choose to enable content in this case.
3) Click on the **View** tab, then click the **Macros** button.
4) In the dialog that appears, make sure the macro **UploadUsers** is selected, then click the **Run** button.

The macro performs the following steps:

1) Attempts to save a CSV version of the **New Users** worksheet to the location **c:\temp\jctn_users.csv**.

   Hint: If the CSV file already exists, the user is prompted to overwrite. Any option chosen besides Yes will cause the macro to end in an error.

2) Attempts to launch the executable JAR that was installed. The JAR is launched with the full path to the CSV file as its only argument. This causes the Java desktop GUI application to run with the CSV file already selected.

Mac

Since support for VBA macros is disabled in Excel 2008, users will have to manually create the CSV and launch the Java program.

Supported Platforms

Initial platform support includes the following:

1. Windows XP 32-bit
2. Windows Vista 32-bit
3. Windows VIsta 64-bit
4. Windows 7 32-bit
5. Windows 7 64-bit
6. MAC OSX (Snow Leopard)

Supported excel versions include:

1) Windows Excel 2010 (version 14)
2) MAC Excel 2008 (versio 12) –VBA support lacking so manual launch of Java is required.

Installers

Installer packages were created for both Windows and MAC platforms. The installers are located in dist/<platform> under the root Git repository folder.

Windows

The Windows installer is available under REPOS_ROOT/dist/windows/BulkUserUploaderSetup.exe. It installs the executable JAR and the Excel spreadsheet in the default x86 program location. It also creates a program group icons launching the Excel macro as well as the Java GUI. A typical installation will create the following path: **C:\Program Files (x86)\Junction Networks**.

MAC

The MAC installer is available under REPOS_ROOT/dist/mac/BulkUserUploader.pkg. It installs a JAR application bundle in /Applications. A typical installation will create the following path: **/Applications/JCTN Bulk User Uploader.** To launch the Java GUI, just launch this application.

Installer Sources

Installer sources are located in the source tree REPOS_ROOT/src/main/installers. A folder exists for each of MAC and Windows platforms.

Versioning

Code will be versioned in a public GitHub repository. The repository URL is **git@github.com:saltysnow/jn_bulk_uploads.git**.