

Vogo For Visto – Upgrade to 3.3 Issues

This document discusses the major issues regarding an upgrade of the **Vogo For Visto** product to include the features, capabilities, and database schema of the Vogo 3.3 product.

The following four major areas are covered.

- User Configuration (Web)
- User Signup (Web)
- Voice UI
- Migration of Existing User Base

User Configuration (Web)

The ability to suppress the display and editing of designated database fields is already present in the Vogo 3.3 product. This ability would be used (via configuration of the Visto provider) to prevent the user from changing the POP3 server. Other fields that are inappropriate for the user to modify can be similarly configured.

Currently, there is no functionality in place for suppressing the display and editing of an entire user configuration section. Because Vogo / Visto users make use of the Visto address book, this would need to be added to the 3.3 product in order to prevent the user from displaying / editing the Vogo contact list.

There would be no need to add another field to the database schema to accomplish this. The field *ProviderEditMask* is a bit mapped integer (described in *\$/FreeAgent/Documentation/Feature Descriptions: Provider Information-Custom Fields.doc*) that has 20 unused bits. Its sphere of influence would be augmented to include an entire screen (for a particularly defined bit) in addition to the individual fields that its bits now represent.

However, this introduces another consideration that must be addressed. If an entire screen (such as Vogo contacts) has been disabled for a particular provider, how should the web application react? There are four choices, listed in increased order of complexity.

- **Menu / Generic** – Leave the menu alone. If the user clicks on the disabled selection, show a generic message. No database schema changes required.
- **Menu / Specific** – Leave the menu alone. If the user clicks on the disabled selection, show a provider specific message. This would involve only slightly more work to establish the provider specific message in the string database. The functionality to prioritize a provider specific string over a provider neutral string is already present.
- **No Menu** – Don't show the disabled choice on the menu. This is the probably the cleanest solution (from a user experience perspective) but it involves the modification of common menu code that is used in other areas

- **Redo Menu** – Remove the current menu design (vertical choices on the left of the screen) and replace with the menu design currently in use on the corporate web pages (horizontal choices on the top of the screen). Include conditional menu presentation functionality and multiple language support. The result would be a unified corporate site / product site look and feel. It would also involve a large amount of changes both to corporate menu functionality and throughout the user signup and configuration pages.

User Signup (Web)

The **Vogo For Visto** product uses provider cooperative URL passing to achieve concurrent Vogo and Visto signup. The basic steps are:

1. Create a Vogo account
2. Perform credit card operations if needed
3. Create Visto Account

Step 1 uses standard database population techniques. Step 3 uses HTTP to pass information to an external server and to receive return confirmation messages. **Step 2 in not currently implemented in the Vogo 3.3 product.**

The major problem with this approach is that is not transactional in nature. A failure on step 3 leaves an incomplete record in the Vogo database. This can lead to user confusion when users retry the signup process. In addition, personal phone numbers may be allocated to an unused account.

A more integrated approach would be to engineer the external communication component (**VSHAG**) so that it could be used as a SQL 7.0 **external stored procedure**. This would allow the SQL signup routine to encapsulate all the necessary steps and perform a transactional rollback if needed.

One of the largest challenges is to establish a configuration arrangement that would allow a provider to declare the necessary external URL and associated parameters that would be needed as well as all pertinent result messages.

In addition, the 3.3 signup form would need to be expanded to include information that it currently does not require. The following sample URL demonstrates this need:

| | URL Portion | Description |
|---|---------------------------|---|
| | http://www.roamtest.com | External host |
| 1 | ?service=registrationvogo | Static parameter |
| 2 | &method=register | Static parameter |
| 3 | &firstname=Joe | Obtained from user input |
| 4 | &lastname=Smith | Obtained from user input |
| 5 | &gender=M | Obtained from user input – not currently on 3.3 input form |
| 6 | &b_year=69 | User birth year – not currently on 3.3 input form |
| 7 | &login=joesmith999 | Requested external ID – not currently on 3.3 input form |

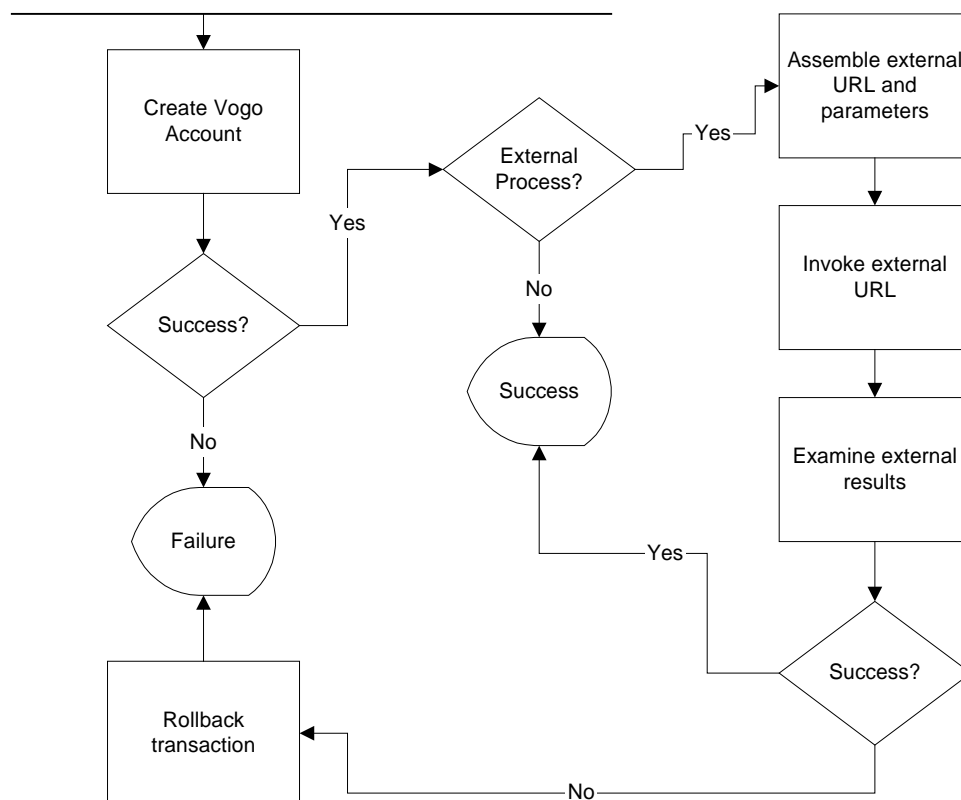
| | URL Portion | Description |
|----|---|--|
| 8 | &password=123456 | External password – not currently on 3.3 input form |
| 9 | &pop3server=mail.visto.com | Static parameter |
| 10 | &pop3login=joesmith999 | Same as #7 |
| 11 | &pop3password=123456 | Same as #8 |
| 12 | &profileid=1001699 | Obtained from Vogo database |
| 13 | &secret=Allen | External password hint. Not currently on 3.3 input form |
| 14 | &timezone=Pacific+Time+%28US+%26+Canada%3B+Tijuana%29 | Obtained from user input* |

- although obtained from the user, the value passed to the account creation routine is an index into the time zone table, not the actual name of the time zone. For this to work properly, the time zone names configured within the Vogo database and the time zone names configured within the external provider's database would need to be the same

The information required for a successful Visto signup could be conditionally presented to the user via the *ProviderEditMask* functionality as described earlier. Adding the five fields shown above and the Vogo contact page as discussed earlier would leave 14 unused bit positions in the mask.

As mentioned above, the larger task involves establishing the external URL, its parameters, and the applicable result messages. To that end, the 3.3 database schema would need to be modified to accommodate this information and to implement the logic as shown in the following (simplified) flow chart.

SQL Stored Procedure - Create Account



In the step **Assemble external URL and parameters**, information would need to be extracted from the database that specified URL parameters that were both static and replaceable. Example:

| Parameter | Value | Type |
|-----------|------------------|-------------|
| service | registrationvogo | Static |
| FirstName | %FNAME% | Replaceable |

For replaceable parameters, a convention would need to be defined that would both establish the parameter as one requiring replacement and indicating the actual value to be used. In the example above, the % signs are used to indicate that a replaceable (not static) value is to be used, and the token **FNAME** is used to indicate that the current value of the user first name is to provide the replacement value.

The simplest way to implement this functionality would be to assign a provider specific string value that held the entire URL (including parameters) that was necessary to achieve the desired result. For example:

http://www.someplace.com?service=registrationvogo&firstName=%FNAME%

is parsed at run time and becomes:

http://www.someplace.com?service=registrationvogo&firstName=Joe

In SQL 7.0, a string may hold up to 8000 characters. This would be more than adequate to configure the external URL.

Another important consideration involves the interpretation of the external results. To accomplish this, it would be necessary to create a table that mapped external results to an internal action code. Example:

| ProviderID | Result Name | Result Value | Action ID |
|------------|---------------------------|----------------|-----------|
| 4000 | x-rp-registration-status | Success | 0 |
| 4000 | x-rpservice-response | AlternateUsers | 1 |
| 4000 | x-rp-registration-status | Failure | 16 |
| 4000 | x-rp-alternate-username | * | -1 |
| 4000 | x-rp-registration-message | * | -1 |

The Action ID column would map to an action table that defines the action.

| Action ID | Class | Description |
|-----------|-----------|---|
| -1 | Info Only | Return result value to caller |
| 0 | Err Code | Successful – continue |
| 1 | Err Code | Failure (external ID clash) – rollback and return |
| 16 | Err Code | Failure (general) – rollback and return |

Actions are of two classes (a) *Error Code*, and (b) *Informational*. Only action codes of class *Error Code* are eligible to be included as a return vale to the caller.

The external operation is assumed to have failed if:

- The external results do not include a name/value pair that contains an **ActionID** of zero
- The external results do not include a result name/pair that is mapped to a non-informational action code
- The external results are contradictory

Having determined the status of the external operation, the SQL server may then continue or rollback as necessary. In addition, the routine would return to the caller a record set containing the overall operational result and the individual result strings that are mapped to an informational action code. The caller may then choose the appropriate response.

Example returned record set (successful operation)

| Overall Result | Result Name | Result Value |
|----------------|-------------|--------------|
| 0 | Null | Null |

Example returned record set (name clash failure)

| Overall Result | Result Name | Result Value |
|----------------|---------------------------|----------------------------|
| 2 | x-rp-alternate-usernames | Joe1 joe2 joe3 |
| 2 | x-rp-registration-message | Houston, we have a problem |

Example returned record set (general failure)

| Overall Result | Result Name | Result Value |
|----------------|---------------------------|----------------------------------|
| 16 | Null | Null |
| 16 | x-rp-registration-message | Houston, we still have a problem |

Credit Card - In addition to the signup considerations already discussed, the issue of credit card verification would need to be addressed. In the Vogo 3.3 product, no credit card validation functionality is present. Since Vogo For Visto 3.1 users are permitted to utilize a credit card to upgrade their account, this functionality would need to be included in the Vogo 3.3 product. As with the **VSHAG** component discussed above, in order to achieve the maximum level of transactional integrity, the **VCASH** component would need to be implemented as a SQL 7.0 **external stored procedure**.

Voice UI

In Vogo 3.3, the facility for having provider specific voice interface pages has already been established. Via configuration of the base profiles that belong to a particular provider, the administrator can utilize separate directories to hold the provider specific home pages.

Currently, the Vogo 3.1 voice UI pages are English only. In order to internationalize these pages, some Visto specific strings would need to be entered into the string database, and the

pages would need to be rewritten to use the calls into the language translation modules rather than referring to static constants as they do now.

Migration of Existing User Base

The existing **Vogo For Visto** user base will require migration from the 3.1 schema to the 3.3 schema. This may be accomplished by setting up a batch routine to handle the processing. This routine would of necessity be fairly complex in order to take into account the various differences between the two database models.

Once this routine has been written and tested, a coordinated effort would be required to make the transition.

- 1) Replace 3.1 user configuration and signup pages with ones that announce the system's temporary unavailability and/or refer users to the new user configuration entry point
- 2) Change the Visto 3.1 database to DBO use only, preventing users from accessing it.
- 3) Perform the migration routine
- 4) Replace the corporate web pages that contain links to 3.1 Visto signup / user configuration with ones that have the new links.

Profile ID considerations

When a user is migrated to the 3.3 database, they would normally obtain a new profile ID. The new profile ID will no longer map into the Visto system or may perhaps (though unlikely) map to an incorrect Visto user.

One solution would be to explicitly transfer the old profile ID into the new profile ID. This would require that the 3.3 database be freshly created or not contain any profile ID conflicts. Since the Visto 3.1 database begins its Profile ID numbering at 1000000, and the 3.3 database begins its Profile ID numbering at 100, it is unlikely that there would be any conflict.

A side effect of this solution is that future Profile ID numbers would be assigned starting above the largest current 3.1 Profile ID. This diminishes the capacity of the system by approximately one million records.

Another solution would involve placing the old Profile ID into the External Subscriber ID field. This would allow access to the old ID while still retaining the 3.3 profile ID allocation method. However, this would mean that modules requiring the old ID value would need to be modified to look into the External Subscriber ID. In addition, this raises issues regarding interpretation of the External Subscriber ID; it is conceivable that a single database may hold records that use this field for two different purposes.

User ID Considerations

A more serious problem occurs with the User ID.

Note: **User ID** refers to the number that you punch into the telephone key pad when accessing the voice UI and the number that you enter to gain access to the web user configuration pages. Although this does not agree with the actual field names used in the 3.1 and the 3.3 product, for purposes of this discussion, the common terminology **User ID** will be used.

In the 3.1 database, users are able to choose their own User ID. In 3.3, the system assigns all User ID values. The conflict occurs when a self selected User ID is the same as a system selected one. Examining the current Visto data, it can be determined that – without countermeasures - the first post migration signup process will fail.

Basically, the problem occurs because the 3.3 system – in assigning User ID values – is expecting to be the sole agent in charge. Its allocation method does not take data placed by other agents into account. The User ID is determined with:

```
SELECT @GuestID= ISNULL(MAX(GuestID1)+1,100001)  
FROM voicenet.user_profiles WITH ( UPDLOCK )  
WHERE GuestID1 LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]'
```

Or: Find the largest 6 digit User ID in the database and add one to it. If there aren't any, choose the value 100001

Note: 6 digit values for User ID were chosen in order to present a balance between usability and capacity.

The largest 6 digit User ID currently in the Visto data is: 999999. Since there is also a User ID of 1000000, conflict occurs at once. Even if the User ID of 1000000 were not present, the selection method shown above fails after a single signup.

Note: There are also existing User ID values of 999980, 999987, 999988, 999989, 999991, and 999994 which would cause problems after 5-19 signups. These however are base profiles and may safely be altered during the batch migration routine.

Assuming the account containing the 999999 User ID could be removed or altered, there is still a User ID in the current Visto data that would trigger a conflict after 877 signups.

If the above selection method is modified to be based upon 7 digit User ID values, the problem still exists because of the existence of self selected User ID 9999999. After one signup, the process will fail.

The same is true of 4, 5, 8 and 9 digit values. The User ID values 9999, 99999, 99999999, and 999999999 have all been self selected.

Possible Solutions for User ID Considerations

Assign new User ID (non transparently) to all migrated users

From a system standpoint, this would be the best solution. From a user standpoint, it would likely be the worst solution. Assuming even that users would be accept the idea of obtaining a new User ID, the matter of communicating this information to them and dealing with the resultant misunderstandings would very likely be an large administrative load.

Assign new User ID (transparently) to all migrated users

In this solution, new User ID values would be assigned to all migrated users but they would not be aware of it. By creating a new table to map the old User ID values to the new User ID values, users would still enter their regular User ID to gain access to the system.

The drawbacks of this solution are increased complexity of the user validation process (must check User ID in two places and perform translation if necessary) and the continued possibility of clashes between newly assigned User ID values (obtained during post migration signup) and old User ID values. For instance, assuming only one user requires migration, the following situation could likely arise:

| Profile Table | |
|-------------------|---------|
| User Name | User ID |
| Joe Migrated User | 100001 |
| Joe New User 1 | 100002 |
| ... | |
| Joe New User 290 | 100291 |

| Mapping Table | |
|---------------|-------------|
| Old User ID | New User ID |
| 100291 | 100001 |
| | |
| | |
| | |

Additional mechanisms would be required to either resolve or prevent the conflict

Keep original User ID values removing sources of conflict

If it could be accurately determined that certain, specific profiles could be removed or altered, it would be possible to free up enough User ID values to guarantee successful post migration signup. The number of successful post migration signups that would be possible depends upon the number of (specific) profiles that could be removed or altered.

| Using 6 digit User ID Values | |
|--------------------------------|--|
| # of Existing Profiles Removed | # of Successful Post Migration Signups |
| 1 | 5 |
| 6 | 19 |
| 8 | 3238 |
| 12 | 8833 |
| 16 | 10101 |
| 19 | 18020 |
| 22 | 26517 |
| 25 | 35427 |
| 31 | 49281 |

| Using 7 digit User ID Values | |
|--------------------------------|--|
| # of Existing Profiles Removed | # of Successful Post Migration Signups |
| 1 | 7670 |
| 3 | 14998 |
| 4 | 34822 |
| 6 | 53702 |
| 8 | 62379 |
| 9 | 76412 |
| 10 | 87611 |
| 12 | 114752 |
| 16 | 141617 |

To achieve these results, it is necessary to remove or alter *specific* user profiles. Also, these values are based upon data analysis as of January 31, 2001 and may change as new users enter the system and choose their User ID values.

Modify The User ID Allocation Process

To retain all existing User ID values, it would be necessary to modify the User ID allocation process that is currently in use. Normally (using 6 digit User ID values), there are 899, 999 possible User ID values that may be assigned. As mentioned above, 6 digit values for User ID were chosen in order to present a balance between usability and capacity.

Currently, the Vogo 3.1 database contains 872 six digit User ID values. To accommodate these existing values, the User ID allocation process could still deal in 6 digits values but would need to do so non sequentially. This would involve extra processing and complexity during the signup process but would result in the system's ability to allocate nearly all the User ID values it was originally designed to handle.

Summary

The **Vogo For Visto** upgrade to 3.3 involves several challenges.

Depending upon certain design decisions, the changes to the user configuration web pages *may* not involve new techniques, but rather a modification and extension of existing ones.

The changes to the signup routines are considerable. This involves developing new external stored procedure functionality, making database schema changes, developing credit card validation functionality, seriously modifying the signup routines and modifying some of the administrative modules to handle new provider configuration issues.

The voice user interface could be used with minimal modification. However, to incorporate internationalization functionality (cornerstone of Vogo 3.3) would in essence involve **rewriting the home pages in their entirety.**

User migration presents data transformation and conflict issues that must be resolved. These issues have signup and (possibly) user configuration ramifications as well.