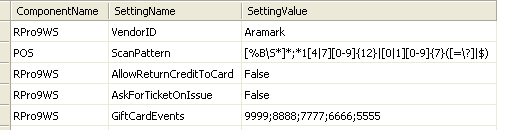
Retail Pro V9 STADIS Plugin

# Installation

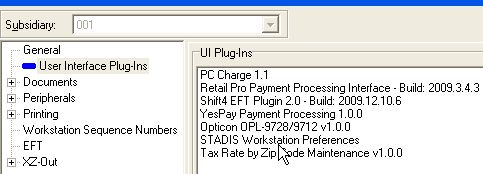
1. .Net V4 must be installed on the machine.
2. Make sure that “Item Note1” and “Item Note2” are added to the list view fields on the Invoice. Item Note1 will contain issue info. Item Note2 will contain the gift card name for receipt printing.
   1. Go to the Invoices screen.
   2. Switch to form view.
   3. Right-click on the column header in the item listing.
   4. Select Page manager.
   5. Find “Item Note1” and double-click it.
   6. Find “Item Note2” and double-click it.
3. Add workstation settings to the STADIS database InstallationSetting table.
   1. Get file PluginSettings.txt from P:\Projects\ StadisRProV9Plugin\Doc.
   2. The file contains SQL INSERT statements for the settings. Change the SettingValues to the desired settings.
   3. Execute the statements in SSMS to add the records.
4. Choose an ALU or ALUs to be used for gift cards.
   1. Add an inventory item for each ALU.
   2. Record the ALU(s) and the options for each type of gift card in the STADIS database GiftCardInfo table.
5. Install StadisRProV9Plugin.msi to a work directory.
6. Edit StadisRProV9Plugin.dll.config.
   1. Change “StadisWebServiceURL” to point to correct web location.
7. Copy files to the workstation Retail Pro plugin directory, usually C:\RetailPro9\Plugins.
8. Register the plugin using Regasm.exe:
   1. Make sure the version of Regasm you are using is correct for the .Net release and 32/64 bits.
   2. Start / Run… / cmd
   3. Enter: cd c:\RetailPro9\Plugins
   4. C:\RetailPro9\Plugins > regasm StadisRProV9Plugin.dll
9. In Retail Pro, add the buttons to the screens.

# Settings

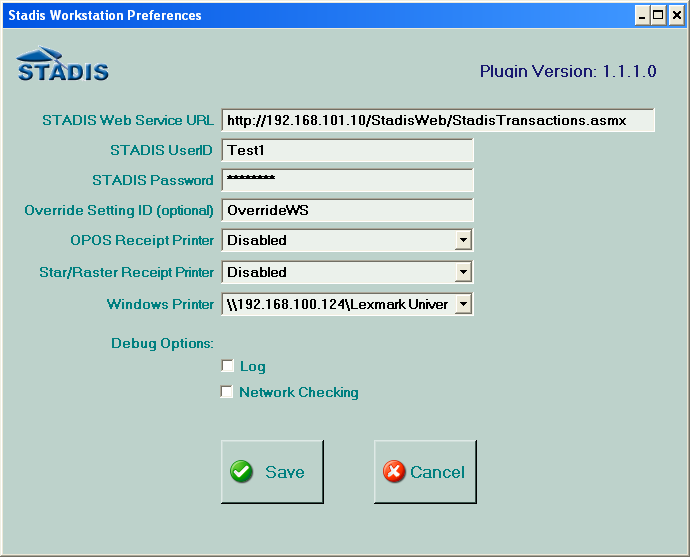
The V8 plugin stored all settings in an ini file. The V9 plugin stores settings that are the same for all of a vendor’s workstations, or for large groups within a vendor’s workstations, in the STADIS database InstallationSetting table. For instance, all of Aramark might have the same settings, or Retail might have different settings from Food & Beverage. The base, or most common set of settings, is in the table with the standard ComponentName of “RPro9WS”. These are loaded at all workstations and do not need to be specified. If any workstations depart from these standard settings, they may specify an Override Setting ID. In a second pass, these settings will override the standard settings.



Only the handful of settings that are specific to the workstation are stored locally, such as UserID, Password, the URL of the Web Service, and the ComponentName corresponding to any override settings that are to be loaded for this workstation. The V8 plugin used a standalone configuration program to change settings. The V9 plugin uses Retail Pro’s WS Preferences:



Double-clicking on “STADIS Workstation Preferences” brings up this screen:



The V8 plugin stored the settings in an ini file. The V9 plugin uses a .Net settings file. The first time the plugin executes, the settings file doesn’t exist. .Net takes the default values from StadisRProV9Plugin.dll.config and creates it. It is an XML file named user.config located in C:\Documents and Settings\<user name>\Local Settings\Application Data\Retail\_Pro\_International,\DefaultDomain\_Path\_<numbers>\<release>\user.config.

**After the first time the plugin is run, changing the values in StadisRProV9Plugin.dll.config has no effect on the settings.** StadisRProV9Plugin.dll.config is only a template. This means that new versions of the plugin can be copied in without overwriting the local settings. The settings have to be changed through WS Preferences or by editing user.config.

# Settings Explanations

Local Settings:

|  |  |  |
| --- | --- | --- |
| **Setting Name** | **Function** | **Example** |
| StadisWebServiceURL | URL of the web service being accessed by this installation. | http://192.168.101.10/StadisWeb/StadisTransactions.asmx |
| InstallationSettingComponent | Used to identify which group of settings should be loaded for this workstation. | AramarkWS |
| StadisUserID | Stadis UserID. Used in web service connection. May be the same for all workstations, or different, for greater security. | Test1 |
| StadisPassword | Stadis Password. Used in web service connection. | password |
| Override Setting ID | ComponentID of any overrides to the standard settings. Leave blank unless this workstation has a setting different from the “RPro9WS” settings. | OverrideWS |
| OPOSPrinterName | Printer for BalanceCheck receipt printing. |  |
| RasterPrinterName | Printer for BalanceCheck receipt printing. |  |
| WindowsPrinterName | Printer for BalanceCheck receipt printing. |  |
| Log | Debug option. Whether logging is enabled. | True or False |
| NetworkChecking | Debug option. Whether network checking is enabled. | True or False |

Installation Settings:

|  |  |  |
| --- | --- | --- |
| **Setting Name** | **Function** | **Example** |
| AllowReturnCreditToCard | Can a return be credited to an existing card? | True or False |
| AskForTicketOnIssue | Whether to prompt for scan when issuing gift cards. | True or False |
| AskForTicketOnRedeem | Whether to prompt for scan on non-Stadis transactions. | True or False |
| DefaultCustomerID | Number to use if customer declines to scan a ticket. | 99999 |
| FeeOrTenderForIssueOffset | Whether to use a fee or a negative tender as offset. | Fee, Tender |
| FormLogoImage | Location of logo to be used on forms. | C:\RetailPro9\Plugins\abc.jpg |
| GiftCardEvents | Used to identify scans from gift cards. | 9999;8888;7777;6666;5555 |
| ImageTransparentColor | RGB value of logo transparent color – defaults to magenta. | 255,255,255 (white) |
| IsMergeFunctionEnabled | Balance Check – Is Merge button visible? | True or False |
| IsPrintingEnabled | Balance Check – Can receipt copies be printed? | True or False |
| IssueGiftCardForReturn | On a return, are funds paid out as a gift card? | True or False |
| PostNonStadisTransactions | Controls whether transactions without a Stadis issue or redeem will be written to the Stadis database. | True or False |
| ReturnGiftCardALU | If IssueGiftCardForReturn = True, what is the ALU of the gift card to be used? | GiftCardALUR |
| ScanPattern | Used to validate ticket numbers. See following section for full explanation. | 1[47][0-9]{12}$|[01][0-9]{7}$ |
| ShowStadisActionGrid | Balance Check – Should StadisActions be shown? | True or False |
| TenderTypeForStadis | Tender type used for Stadis redeems. | GiftCard, Check or ForeignCheck |
| VendorID | Value that will appear in the VendorID field of the Transaction and StadisAction. | ARAR |
| BalChkButtonActive | Controls whether button appears or not. | True or False |
| BalChkButtonCaption | Wording that appears on the button. | Gift Card $ Check |
| BalChkButtonImage | Icon that appears on the button. | Stadis32.bmp |
| BalChkButtonHint | Text in pop-up balloon help. | Check balance on gift card / ticket. |
| IssueButtonActive | See explanation for BalChkButton. | True or False |
| IssueButtonCaption | “ “ “ “ | Issue Gift Card |
| IssueButtonImage | “ “ “ “ | Stadis32.bmp |
| IssueButtonHint | “ “ “ “ | Issue STADIS Gift Card(s). |
| RedeemButtonActive | “ “ “ “ | True or False |
| RedeemButtonCaption | “ “ “ “ | Gift Card / STADIS |
| RedeemButtonImage | “ “ “ “ | Stadis32.bmp |
| RedeemButtonHint | “ “ “ “ | Redeem a STADIS gift card or ticket. |
| ReturnButtonActive | “ “ “ “ | True or False |
| ReturnButtonCaption | “ “ “ “ | Return Gift Card / STADIS |
| ReturnButtonImage | “ “ “ “ | Stadis32.bmp |
| ReturnButtonHint | “ “ “ “ | Return STADIS gift card or ticket. |
| ReloadButtonActive | “ “ “ “ | True or False |
| ReloadButtonCaption | “ “ “ “ | Reload Gift Card |
| ReloadButtonImage | “ “ “ “ | Stadis32.bmp |
| ReloadButtonHint | “ “ “ “ | Add money to STADIS gift card. |

# GiftCardInfo table

This table contains settings for each type of card that will be sold. If gift cards are being sold, there must be at least one entry. If there is more than one entry, buttons for the cards will appear on the Issue screen. Cards that have been issued/activated, or cards waiting to be activated, will have the RProLookupALU in the Ticket UDF3 field, so that we can do lookups in the table for issues, reloads, etc.

|  |  |  |  |
| --- | --- | --- | --- |
| **Setting Name** | **Type** | **Function** | **Example** |
| GiftCardInfoID | GUID | Unique identifier for record |  |
| GiftCardName | Varchar(50) | Full name of gift card | Mega Bucks $50 Gift Card |
| RProLookupALU | Varchar(50) | This will be used to lookup the corresponding item in Retail Pro inventory. An inventory item needs to be defined for each different card type. | GiftCardALU1 |
| EventID | Varchar(50) | EventID to be associated with this card. | 9999 |
| ButtonPosition | Integer | Left to right position of the button for this gift card on the Issue screen. | 1 through 6 |
| ButtonCaption | Varchar(50) | Description of this card as it will appear on its corresponding button. | MegaBucks $50 |
| DropdownCaption | Varchar(50) | Description of this card as it will appear in the dropdown box in the grid. **Note: If possible, make the first character unique, to facilitate autocomplete.** | 50$ MBuck |
| ReceiptCaption | Varchar(20) | Description of this card as it will appear on the receipt. | Mega Bucks Gift Card |
| FixedOrVariable | Char(1) | Whether the amount is fixed or customer-selectable. | F or V |
| AllowIssue | Bit / Boolean | Can this card type be issued, or does it have to preexist in the database? | True or False |
| AllowActivate | Bit / Boolean | Can this card type be activated? | True or False |
| IAMinAmount | Money / Decimal | Fixed – Amount of card  Variable- Minimum amount for card issue. | 50.00  20.00 |
| IAMaxAmount | Money / Decimal | Fixed – Same as IAMinAmount  Variable – Maximum amount for card issue | 50.00  1000.00 |
| AllowReload | Bit / Boolean | Can the customer add money to an existing gift card via a reload? | True or False |
| RMinAmount | Money / Decimal | The minimum amount of money that can be put on a card in a reload. | 10.00 |
| RMaxAmount | Money / Decimal | The maximum amount that can be put on the card through reloads (*not* the maximum amount for a reload). Reloads can’t take the card beyond this amount. | 1000.00 |

# Scan Patterns

The V8 plugin had a setting called “CustomerBarcodeLength” that was used to detect incomplete scans. However, customer requests quickly escalated beyond checking length. The Mets wanted us to allow only 14-digit numbers that started with 14 or 17. The Texans wanted only 14-digit numbers that started with 14 or 17, *or* 8-digit numbers that started with 0 or 1. This would have required us to hard-code all of these tests and weigh the code down with special-exception routines along the lines of “If site = “Mets” do this, else if site = “Texans” do that, else …”. To avoid this, and to give us more flexibility, the StadisPOS and the V9 plugin use regular expression pattern matching.

What is a regular expression? According to Wikipedia:

In [computing](http://en.wikipedia.org/wiki/Computing), **regular expressions**, also referred to as **regex** or **regexp**, provide a concise and flexible means for matching [strings](http://en.wikipedia.org/wiki/String_(computer_science)) of text, such as particular characters, words, or patterns of characters. A regular expression is written in a [formal language](http://en.wikipedia.org/wiki/Formal_language#Programming_languages) that can be interpreted by a regular expression processor, a program that either serves as a [parser generator](http://en.wikipedia.org/wiki/Parser_generator) or examines text and identifies parts that match the provided [specification](http://en.wikipedia.org/wiki/Specification_(technical_standard)).

The following examples illustrate a few specifications that could be expressed in a regular expression:

* The sequence of characters "car" in any context, such as "car", "cartoon", or "bicarbonate"
* The word "car" when it appears as an isolated word
* The word "car" when preceded by the word "blue" or "red"
* A dollar sign immediately followed by one or more digits, and then optionally a period and exactly two more digits (for example "$10", or "$245.99").

Regular expressions can be much more complex than these examples.

In other words, regular expressions are like wild cards on steroids. Regex patterns can look fairly intimidating, but we have two things going for us: (1) the part of the pattern that has to be changed for a particular site isn’t too complicated, and (2) we have a tool that can be used to quickly test new patterns.

For a good explanation of regular expressions, see <http://www.regular-expressions.info/>. It has a full tutorial as well as a one-page quick-start.

Regex operators: **. $ ^ { [ ( | ) \* + ? \**

To use any of the regex operators as characters, put a backslash in front of them. For example, an \* in a pattern means “zero or more times”. To indicate an asterisk in that position, you would use \\* instead.

|  |  |  |  |
| --- | --- | --- | --- |
| **Regex** | **Meaning** | **Example** | **Description** |
| Quantifiers | | | |
| **\*** | Zero or more matches |  |  |
| **+** | One or more matches | A+ | Matches A, AA or AAA, but ABA would result in two matches. |
| **?** | Zero or one match |  |  |
| **{N}** | Exactly N matches | [01]{4} | Matches any four-digit binary number, such as 1010 or 1001. |
| **{N,}** | At least N matches | [0-9]{3,} | Matches 123 or 1234, but not 12. |
| **{N,M}** | N to M matches | [0-9]{1,3} | Matches 1, 12 or 123. For 1234 you would get two matches, 123 and 4. |
| Character matching | | | |
| **.** | Matches any character |  |  |
| **[aeiou]** | Matches any character between the brackets | [aeiou]  gr[ae]y | Matches any vowel  Matches gray or grey |
| **[^aeiou]** | Any character except the ones between the brackets | [^aeiou] | Matches any non-vowel |
| **[a-zA-Z]** | The dash lets you specify a range of characters | [0-9]  [a-zA-Z] | Any digit  Any upper or lower case letter |
| **(abc)** | The characters between the parentheses are a group. Usually used with an or character or a quantifier. | (ab){1,3}  (ab|ax) | Matches ab, abab or ababab  Matches ab or ax |
| **\w** | A word character. Same as [a-zA-Z\_0-9]. |  |  |
| **\W** | A non-word character. Same as [^a-zA-Z\_0-9]. |  |  |
| **\s** | A white-space character (space, tab, newline, etc.) |  |  |
| **\S** | A character other than a white space. |  |  |
| **\d** | A decimal digit. Same as [0-9] |  |  |
| **\D** | A non-digit character. Same as [^0-9]. |  |  |
| Miscellaneous | | | |
| **|** | Or. | (AA|BB) | AA or BB. |
| **(?# comment)** | Comment |  |  |
| Character escapes | | | |
| **\t** | Tab. Same as \x09. |  |  |
| **\r** | Carriage return. Same as \x0D. |  |  |
| **\n** | Newline character. Same as \x0A. | \r\n | Matches end of line in a .txt file. |
| **\x20** | An ASCII character in Hex notation. | \x20 | Matches space character. |
| **\cC** | An ASCII control character. | \cC | Matches control-C. |
| Zero-width matches | | | |
| **^** | The beginning of the string. | ^The | Matches “The” at the start of a string. |
| **$** | The end of the string. | xyz\.$ | Matches the end of a string that ends with “xyz” and a period. |
| **\b** | Word boundary. The first or last letter of a word. | \bA | Matches the A in Axe or USA, but not in CAP or ace. |
| **\B** | Not on a word boundary. | \BA | Matches the A in CAP, but not in Axe or USA. |
| Zero-width look-ahead and look-behind | | | |
| **(?=subexpr)** | Positive look-ahead. | \w+(?=,) | Matches a word followed by a comma, without matching the comma. |
| **(?!subexpr)** | Negative look-ahead. | \w+\b(?![,:;]) | Matches a word that isn’t followed by a comma, colon or semi-colon. |
| **(?<=subexpr)** | Positive look-behind. | (?<=[,;])\w+ | Matches a word that follows a comma or semi-colon. |
| **(?<!subexpr)** | Negative look-behind. | (?<!,)\b\w+ | Matches a word that doesn’t follow a comma. |

There are other commands. These are just the ones you’re most likely to run into or need.

Let’s examine a scan pattern:

**1[47][0-9]{12}$**

Breaking it down:

**1 A one**

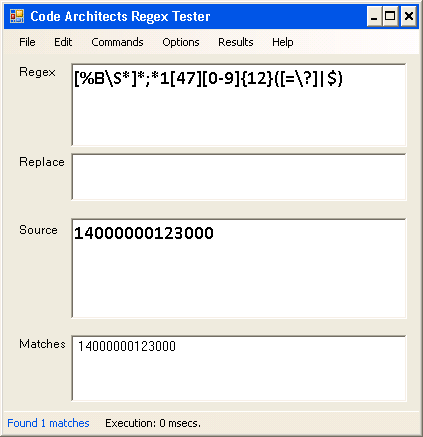
**[47] followed by a four or a seven**

**[0-9] followed by a digit**

**{12} twelve times**

**$ the end of the string (so we don’t match strings longer than 14 chars)**

The Regex test tool is located at M:\Hardware and Software\SOFTWARE\RegexTester. The following is a screen shot:



The Regex pattern goes at the top, scan or type the input into the Source box, and hit F5. If the input fits the pattern it will appear in Matches.