Registration Model

To use Dynamic Registration at Handango, log in to Handango and click **Add A Product** or **Modify** the product you wish to set to dynamic registration. Scroll down to the third section under your product details and click **Modify Registration Information**. Change the registration method to ?Dynamic? and enter your RPN string.

General Definitions

The C/DUI (Customer/Device Unique Identifier) - The handheld device user selects their C/DUI when they first set up their handheld device. Most users choose their name or some variation thereof. Device-specific examples of C/DUIs are the **Palm User Name** on Palm OS handheld devices, the **Owner Name on Pocket PC** handheld devices, and the **IMEI** on Symbian OS devices. For Symbian OS devices, customers will be asked to enter their IMEI in the following format: **350443-61-001241-9 (including dashes)**. Please be mindful of this fact when creating your application and code verification process. The IMEI will be run through the RPN string with the dashes included.

For example: Will's Palm User Name is ?Will P?, so his C/DUI is also "Will P". The C function call you will use to get the C/DUI off of a handheld device is **DlkGetSyncInfo (NULL,NULL,NULL,name,NULL,NULL);** where *name* is a char array of at least 41 characters (i.e char name[41]). When this C function completes, name will contain the C/DUI as a null terminated (ASCIIZ) string.

The RPN String (Reverse Polish Notation String) -- This is a string you give us along with your application that enables us to generate valid dynamic registration codes for customers who buy your application.

Dynamic Registration Code -- The five-digit numeric code that the user must enter to unlock your application on his or her handheld device.

Summary

Dynamic Registration protects you, the author, because the dynamic registration code that the user must enter to unlock your application depends on what the C/DUI on their handheld device is. Since different users select different C/DUI's for some handheld devices, a dynamic registration code that unlocks your application on one handheld device probably won't work on another handheld device. The only way the same dynamic registration code will work on two different handheld devices is if those two handheld devices have the same C/DUI. As you can see, this helps prevent piracy because the only way a pirate could copy your software from his friend would be to make sure that the C/DUI on his handheld device is the same as the C/DUI on his friend?s. It becomes more tedious and impractical for the pirate to change his C/DUI each time he wants to run a different application.

Step-by-Step

Here is an example of how Handango uses the RPN string you give us to compute the correct dynamic registration code for a user based on the C/DUI on that user's handheld device.

For our example:	**C/DUI** = "Will P" **RPN string** = "i 0 == 111 * key + c 2 * +"

1. We convert each character of the C/DUI to ASCII value.
   W = 87 decimal
   i = 105 decimal
   l = 108 decimal
   l = 108 decimal
   [space] = 32 decimal

P = 80 decimal

2. We apply the RPN string to the ASCII value of each character in the C/DUI. The RPN string is actually a mathematical formula written in reverse Polish notation. Where 'i' is the position of the character in the C/DUI starting at 0, and 'c' is the ASCII value of the character in the C/DUI. (We will discuss 'key' later in this example.)
3. We start out by setting the variable ?key? = 0.
4. Then we apply the RPN string to the first character, 'W'. For 'W' i=0 since 'W' is the first character of the C/DUI and we start counting at 0, and c=87 since 'W' has an ASCII value of 87.
5. Now apply the RPN string. The RPN string is evaluated by placing each operand in it on a stack, when an operator is reached, it is applied to the last two operands on the stack. Those operands are popped from the stack and the result is pushed back onto the stack.
6. We process the first operand 'i' in the RPN string by pushing it onto the stack, now stack = i.
7. Then we process the next operand in the RPN string, '0' by pushing it onto the stack, now stack = i 0.
8. Next we come to the '==' which is an operator that works just like the C operator. We pop the last two operands from the stack which were i and 0 and we apply the operator '==' to them. Since i equals 0, the result is true. So, 1 is pushed back onto the stack, now stack = 1.
9. Then we process the next operand in the RPN string which is '111' by pushing it onto the stack, making stack = 1 111.
10. Next, we come to the '*' which is an operator representing multiplication. We pop the last two operands from the stack, which were 1 and 111, and we multiply them. The result is 111 which is pushed back onto the stack, making stack = 111.
11. Then we process the next operand in the RPN string, which is 'key', by pushing it onto the stack, making stack = 111 key.
12. Next we come to the '+' which is an operator representing addition. We pop the last two operands from the stack, which are 111 and ?key?, and we add them together. Since key=0, and 111 + 0 = 111, the result is 111 which is pushed back onto the stack, making stack = 111.
13. Then we process the next operand in the RPN string which is 'c' by pushing it onto the stack, making stack = 111 c.
14. Then we process the next operand in the RPN string, which is '2', by pushing it onto the stack, making stack = 111 c 2.
15. Next we come to the '*' which is an operator representing multiplication. We pop the last two operands from the stack, which are c and 2, and we multiply them. Since c=87 and 2*87=174, the result is 174 which is pushed back onto the stack, making stack = 111 174.
16. Next we come to the '+' which is an operator representing addition. We pop the last two operands from the stack, which were 111 and 174, and we add them together. 111+174=285. The result is 285, which is pushed back onto the stack, making stack = 285.
17. Now we have reached the end of the RPN string so the result of applying the RPN string to the first character 'W' of the C/DUI is 285.
18. We set key=285. **This concludes applying the RPN string to 'W'**.
19. **Now we apply the RPN string to the next character 'i'**, which is the second character in the C/DUI, so i=1 since we start counting at 0. The ASCII value of 'i' is 105. So, c=105 and from processing 'W' key=285.
20. We process the first operand 'i' in the RPN string by pushing it onto the stack, now stack = i.
21. Then we process the next operand in the RPN string, '0' by pushing it onto the stack. Now stack = i 0.
22. Next we come to the '==' which is an operator that works just like the C operator. We pop the last two operands from the stack which were i and 0 and we apply the operator '==' to them. Since i equals 1, the result is false, so 0 is pushed back onto the stack. Now stack = 0
23. Then we process the next operand in the RPN string, which is '111', by pushing it onto the stack giving, stack = 0 111.
24. Next we come to the '*' which is an operator representing multiplication. We pop the last two operands from the stack, which were 0 and 111, and we multiply them. The result is 0 which is pushed back onto the stack, making stack = 0.
25. Then we process the next operand in the RPN string, which is 'key', by pushing it onto the stack, making stack = 0 key.
26. Next we come to the '+' which is an operator representing addition. We pop the last two operands from the stack, which were 0 and key, and we apply addition to them. Since key=285, and 285 + 0 = 285,the result is 285 which is pushed back onto the stack, making stack = 285.
27. Then we process the next operand in the RPN string, which is 'c', by pushing it onto the stack, making stack = 285 c.
28. Then we process the next operand in the RPN string, which is '2', by pushing it onto the stack, making stack = 285 c 2.
29. Next we come to the '*' which is an operator representing multiplication. We pop the last two operands from the stack, which were c and 2, and we multiply them. Since c=105 and 2*105=210, the result is 210, which is pushed back onto the stack, making stack = 285 210.

30. Next we come to the '+' which is an operator representing addition. We pop the last two operands from the stack, which were 285 and 210, and we add them together. 285+210=495. The result is 495, which is pushed back onto the stack, making stack = 495.
31. Now we have reached the end of the RPN string so the result of applying the RPN string to the second character 'i' of the C/DUI is 495.
32. We set key=495. **This concludes applying the RPN string to 'i'**
33. **Now we apply the RPN string to the next character 'l'**, which is the third character in the C/DUI, so i=2 since we start counting at 0. The ASCII value of 'l' is 108. So, c=108 and from processing 'l' key=495.
34. Applying the RPN string as in the previous step, we get key=495 + 2*108 = 711.
35. We set key=711. **This concludes applying the RPN string to the first 'l'**.
36. **Now we apply the RPN string to the next character 'l'**, which is the fourth character in the C/DUI, so i=3 since we start counting at 0. The ASCII value of 'l' is 108. So, c=108 and from processing 'l' key=711.
37. Applying the RPN string as in the previous step, we get key=711 + 2*108 = 927.
38. We set key=927. **This concludes applying the RPN string to the second 'l'**.
39. **Now we apply the RPN string to the next character** [space], which is the fifth character in the C/DUI, so i=4 since we start counting at 0. The ASCII value of [space] is 32. So, c=32 and from processing [space] key=927.
40. Applying the RPN string as in the previous step, we get key=927 + 2*32 = 991.
41. We set key=991. **This concludes applying the RPN string to [space]**.
42. **Now we apply the RPN string to the next character** 'P', which is the sixth character in the C/DUI, so i=5 since we start counting at 0. The ASCII value of 'P' is 80. So, c=80 and from processing 'P' key=991.
43. Applying the RPN string as in the previous step, we get key=991 + 2*80 = 115.
44. Since we are at the end of the C/DUI, we get key = 1151. Next we add a 0 to the beginning to make it a five-digit number, thus our dynamic registration code for a C/DUI of 'Will P' is 01151.
    - So for C/DUI = "Will P' and RPN String = "i 0 == 111 * key + c 2 * +"
    - The dynamic registration code = "01151"

Note: The RPN String "i 0 == 111 * key + c 2 * +" Adds up the ASCII values of all the characters in the C/DUI, multiplies that by 2 and adds 111 to that.

- For "Will P": 87 + 105 + 108 + 108 + 32 + 80 = 520; 520 * 2 = 1040; 1040 + 111 = 1151.
- An RPN string of "i 0 == 111 * key + c 3 * +" would add up the ASCII values of all the characters in the C/DUI. Multiply that by 3 and add 111 to that.
- So, for "Will P": 520*3=1560; 1560 + 111 = 1671.
- An RPN string of "i 0 == 987 * key + c +" would add up the ASCII values of all the characters in the C/DUI and add 987 to that.
- So for "Will P": 520 + 987 = 1507.

RPN Strings can be much more complex than this. Just about any operator is allowed:

- Logical operators such as &&, ||, !, ==, >=
- Bitwise operators such as << (Shift left), >> (Shift right), ~ (Invert), & (AND), | (OR).
- Arithmetic operators such as +, -, *, /, % (Modulo)

Other Things to Remember

- The only three variables allowed in an RPN string: **key**, **i**, and **c**. There is no variable for RPN string length.
- Dynamic registration code generation is done using 32-bit signed integers, and the final result is converted to a 16-bit unsigned integer. The 32-bit signed integer result is converted to a 16-bit unsigned integer by masking off all but the lowest order 16 bits. For example, if the final result is -3 (0xFFFFFFFD), then the last or lowest order 16 bits would be 0xFFFD hex which is unsigned decimal 65533. The variable c is considered to be an unsigned 8-bit integer. The smallest possible registration code is 0 and the largest possible registration code is 2^16 - 1 = 65535. If a registration code is fewer than five digits, 0s are added to the front to make it five digits. So, 763 becomes 00763. Because only integers are used in the dynamic registration code generation, 763 / 10 = 76 not 76.3. Consequently, decimals never appear in a dynamic registration code.
- Also, the program that processes RPN strings puts a limit on the number of characters then processed. If a handheld device ID exceeds 10 characters, only the first five and last five characters are used to generate the unlock code.

- Non-cumulative operations are evaluated from left to right, so 7 3 - = 4 (not -4).
- Currently, 16-bit characters such as japanese, chinese, greek, or hebrew characters are not supported in this process as the variable c is treated as an 8-bit unsigned integer.

If you have any questions about this process, feel free to contact partners@handango.com or your Business Development representative.