SRP_HashTable

SRP Hash Tables are high-performance, in-memory tables.

Method	Description
Create	Creates an SRP Hash Table
Contains	Determines whether or not a key exists in an SRP Hash Table
Count	Gets the number of key-value pairs in an SRP Hash Table
Get	Get the value paired with a given key in an SRP Hash Table
GetKeys	Gets all the keys of an SRP Hash Table in no particular order
GetValues	Gets all the values stored in an SRP Hash Table in no particular order
GetKeyValuePairs	Gets all the keys and their paired values of an SRP Hash Table in no particular order
Release	Releases the handle to an SRP Hash Table
Remove	Removes a key and it's paired value from an SRP Hash Table
Set	Pairs a value with a given key in an SRP Hash Table

Hash Table Defined

In this documentation, the term Hash Table refers to an in-memory collection of elements, which are referenced by a key. This is also known as a key-value-pair collection, or a dictionary.

RTP65

OpenInsight does support in-memory linear hash tables, which they call the Fast Filing System. The method for using these tables is RTP65, and OpenInsight uses it to cache frequently used tables. It is available for use by developers, but documentation is scarce and it's not the most intuitive. It also has the limitation that RTP65 supports only a limited amount of these in-memory tables.

SRP Hash Table

SRP Hash Tables offer the same functionality as the Fast Filing System in an easy to use API that offers more flexibility and performance. While the Fast Filing System is always case sensitive, SRP Hash Tables can be case sensitive or case insensitive. SRP Hash Tables also allow you to predict the number of elements it will contain so it can allocate it's in-memory data structures to keep it super efficient.

To create a case insensitive with a modest amount of pre-allocated space, simply do this:

```
// Create a case-insensitive hash table
Handle = SRP_HashTable("Create")
```

If, however, you need a case sensitive hash table, set the first parameter to 1:

```
// Create a case-sensitive hash table
Handle = SRP_HashTable("Create", 1)
```

And if you expect your table to be very big (tens of thousands of elements or more), then set the second parameter to an approximation of the number of elements your table will contain. You don't have to be accurate, and you can overshoot the number just in case:

```
// Create a case-sensitive hash table that will contain roughly 250,000 elements
Handle = SRP_HashTable("Create", 1, 250000)
```

Once you have your table created, you can add key-value pairs to it like this:

```
// Set the chores to be done on each day of the week
SRP_HashTable("Set", Handle, "Sunday", "")
SRP_HashTable("Set", Handle, "Monday", "Vacuum")
SRP_HashTable("Set", Handle, "Tuesday", "Take out garbage")
SRP_HashTable("Set", Handle, "Wednesday", "Mow the lawn")
SRP_HashTable("Set", Handle, "Thursday", "Clean bathroom")
SRP_HashTable("Set", Handle, "Friday", "Dust")
SRP_HashTable("Set", Handle, "Saturday", "Have fun!")
```

Notice that you can pass empty keys and empty values. Later, when you need to get the value associated with a key, you do this:

```
// Get today's (Tuesday's) chore
TodaysChore = SRP_HashTable("Get", Handle, "Tuesday")
```

If the key does not exist within the table, then "" is returned. But wait! What if the key does exist and the value is blank? We can use the following method to determine if a key truly exists in the table:

```
// See if the key exists in the table
KeyExists = SRP_HashTable("Contains", Handle, "Someday")
```

The above call would return 0. However, if we specified Sunday (which returns "" as well), we would have gotten a 1.

Now, the most important thing to remember is that, when we're done with an SRP Hash Table, we need to release the memory tied to it.

```
SRP_HashTable("Release", Handle)
```

That's it. If you forget to release a SRP Hash Table handle, the memory will get cleaned up when OpenInsight closes. However, it's a good habit to release SRP Hash Tables you no longer need since leaving them in memory can cause memory leaks, especially if your application runs all day.

Other SRP_HashTable Services

If you simply want to know how many elements are in your hash table, you make one simple call:

```
// How many elements are in my hash table?
NumElements = SRP_HashTable("Count", Handle)
```

You can get a list of all the keys in your table, delimited by any character. If you omit the delimiter, then @FM will be used. Note also that if your SRP Hash Table is case-insensitive, then the keys will always be upper case.:

```
// Get all the keys, comma delimited
KeyList = SRP_HashTable("GetKeys", Handle, ",")
```

You can also get just the list of values in your hash table. Again, you can specify any delimiter character or leave it blank to use @FM:

```
// Get all the values, @FM delimited
ValueList = SRP_HashTable("GetValues", Handle)
```

Lastly, you can get both your keys and values, and the delimiters are up to you:

```
// Get all the keys and values
KeyValuePairs = SRP_HashTable("GetKeyValuePairs", Handle, @TM, @STM)
```

When to Use SRP Hash Tables

Hash tables are most useful when you are running a process that needs to keep track of unique values. Normally, developers use the BASIC+ Locate statement, but this get very slow for large processes. SRP Hash Tables are easier and faster to use in this regard and have much less overhead than using an OpenInsight table.

Points to Remember

Don't forget to release your SRP Hash Table handles. Always.