SRP_Decode

Decodes printable-encoded data into a variable.

Syntax

```
Result = SRP_Decode(String, Algorithm, CharacterSet)
```

Returns

The decoded data.

Parameters

Parameter	Description
String	The data to be decoded. Required.
Algorithm	The decoding algorithm to be used. Optional.
CharacterSet	The character set used when the String was originally encoded. Optional.

Remarks

The SRP_Decode function decodes data that was previously encoded into a printable format. There are three decoding algorithms available:

Algorithm	Description	Version Added
HEX	The input contains hexadecimal pairs for each byte	
BASE32	The input is encoded in BASE32 printable format	
BASE64	The input is encoded in BASE64 printable format	
URL	The input is a URL that might have character escape sequence, such as %20	2.1.5
BASE64URL	The input is BASE64 encoded using URL safe characters	2.1.5

If you leave the Algorithm parameter blank or pass an unrecognized value, then BASE64 is assumed.

OpenInsight 7.1 and later includes a Base64Decode function. The output will be the same as SRP_Decode, though the C++ algorithms used by the SRPUtility.dll will likely perform faster. This is negligible for small amounts of data. SRP_Decode will work for OpenInsight 7.0 and earlier.

OpenInsight also includes it's own "HEX" decoder via the IConv function. However, this decoder does not handle delimiters. This may be useful if you're decoding elements in an array. However, if you are decoding binary data, then the SRP_Decode "HEX" algorithm will serve more useful.

Use SRP Encode when you need to convert binary data into a printable format.

The CharacterSet parameter allows you to customize the characterset used for BASE32 or BASE64. Leaving this blank uses the standard character set recognized by most decoders, including the one built into OpenInsight. If you run into a scenario in which the built-in character set causes issues, then you can customize the character set to fit your needs. Since this is the decoder, then you need to make sure the character set perfectly matches the one used to originally encode the string.

Example

```
* decode using default (BASE64)
Data = SRP_Decode(EncodedData)

* HEX encode some printable data
Data = SRP_Decode(EncodedData, "HEX")

* BASE32 encode some printable data
Data = SRP_Decode(EncodedData, "BASE32")

* BASE64 decode some printable data
Data = SRP_Decode(EncodedData, "BASE64")
```