

# SRP\_JsonX\_BeginString

Creates a new document that must be built in order.

## Syntax

```
SRP_JsonX_BeginString(Name, Init, IsPretty)
```

## Returns

Nothing. SRP\_JsonX\_BeginString always succeeds, but it can produce a warning if you don't pass "{" or "[" as the starting token. See [SRP\\_JsonX\\_Error](#).

## Parameters

| Parameter | Description                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------|
| Name      | The name of the new document, used for debugging purposes only                                                             |
| Init      | Initial partial json. Must at lest be "{" or "[", but can be more hardcoded json to start. <b>Optional.</b> Default is "{" |
| IsPretty  | Set to 1 to make the output human readable, set to 0 to make it compact. <b>Optional.</b> Default is 0                     |

## Remarks

SRP\_JsonX\_BeginString creates a new document. If there was already an active document, that one is placed on the stack and this one becomes the active document.

The Init parameter allows you to initialize the new document. At the very least, you should pass "{" to start the document with an object as the root or "[" to start the document with an array. However, you can pass as much json as you like to initialize the new document. The only thing you can't do is pass complete json since this routine is for starting a document you plan to build, not parse json. If, for example, you had a bunch of hardcoded json members you plan to return in a package, it is faster to pass json than to make a bunch of calls to [SRP\\_JsonX](#) to do the same thing. In the following example, we create a response initialized with some hardcoded members.

```
SRP_JsonX_Begin('ErrorResponse', '{"status":200,"phrase":"OK","method":"GET","URL":"https:\\www.examples.com"}')
  If Len(Errors) then
    SRP_JsonX('errors', '[')
    For Each Error in Errors using @FM
      SRP_JsonX(Error)
    Next Error
    SRP_JsonX(']')
  end
ErrorResponse = SRP_JsonX_End('Pretty')
```

This routine is only for situations where you will be building a json string in order. In fact, [SRP\\_JsonX](#) and [SRP\\_JsonX\\_End](#) are the only other routines you can use with this kind of document. Each call to [SRP\\_JsonX](#) behaves exactly the same as it always does, but instead of creating json structures in memory, each call appends json text to a string buffer. This is why you must specify the formatting via the IsPretty parameter at the beginning since formatting will be done with each call to [SRP\\_JsonX](#). When you end the document by calling [SRP\\_JsonX\\_End](#), the FormatOptions parameter will be ignored, and the json you built will be returned. If you know you are building json in order, this will be faster than using [SRP\\_JsonX\\_Begin](#).

The Name parameter can be anything you want as it is only used for debugging purposes. The name will appear when calling [SRP\\_JsonX\\_State](#) or [SRP\\_JsonX\\_Trace](#).

**Note:** The IsPretty parameter is just a boolean. Setting it to 1 makes the output pretty, and setting it to 0 makes it concise. If you include the SRPJSONX insert in your code, you may also use JsonxPretty\$ or JsonxConcise\$ for more code readability.

## Examples

```
$insert SRPJSONX
```

```
SRP_JsonX_BeginString('DirectToString', '{', JsonxPretty$)
  SRP_JsonX('employees', '[')
    SRP_JsonX('{')
      SRP_JsonX('firstname', 'John')
      SRP_JsonX('lastname', 'Doe')
      SRP_JsonX('age', 21)
    SRP_JsonX('}')
    SRP_JsonX('{')
      SRP_JsonX('firstname', 'Anna')
      SRP_JsonX('lastname', 'Smith')
      SRP_JsonX('age', 32)
    SRP_JsonX('}')
    SRP_JsonX('{"firstname":"Peter", "lastname":"Jones", "age":43}')
  SRP_JsonX(']')
  SRP_JsonX('count', 4)
  SRP_JsonX('active', 1, 'Bool')
  SRP_JsonX('alwaysnull')
  SRP_JsonX('alwaysstring', 4.321, 'String')
Json = SRP_JsonX_End()
```