Mit der Veröffentlichung des Plugin 1 zu EEP16 wurde die Grundlage zur Einführung der Exchange-Speicherklasse geschaffen.

Eine Instanz der Klasse (Objekt) wird bei Initialisierung des SpDrS60 erzeugt. Voraussetzung ist die entsprechende Konfiguration in der "SpDrS60_InitPara_RL2.lua".

Dazu wird der Parameter "_SPDRSS60_ExtDataSet" mit "true" definiert.

Die Bezeichnung des Objekts lautet *_SpDrS60_Exchange* und stellt verschiedene Methoden zur Speicherung von Strings in einer zur "Anl3" korrespondierenden Lua-Datei zur Verfügung.

Die externe Datei gliedert sich grundsätzlich in zwei Abschnitte. Die Sektion *[SpDrStatic]* beinhaltet dabei die statischen Daten.

Die Sektion *[SpDrDynamic]* wird unter anderem zur Speicherung remanenter dynamischer Daten des SpDrS60 genutzt. *Das führt dazu, dass Teile dieser Sektion beim Speichern der Anl3-Datei ständig überschrieben werden.*

Die Methoden des SpDrS60 Exchange-Objekts

_SpDrS60_Exchange.PushDynamicData(_Str)
schreibt einen String in die dynamische Sektion *[SpDrDynamic]*

_SpDrS60_Exchange.PopDynamicData(_Str)
löscht einen einzelnen String aus der dynamische Sektion *[SpDrDynamic]*

_SpDrS60_Exchange.PushStaticData(_Str)
schreibt einen String in die statische Sektion *[SpDrStatic]*

_SpDrS60_Exchange.PopStaticData(_Str)
löscht einen einzelnen String aus der statische Sektion *[SpDrStatic].*

_SpDrS60_Exchange.getDynamicData()
gibt den Inhalt des dynamischen Exchange-Buffer als Stringtable zurück

_SpDrS60_Exchange.getStaticData()
gibt den Inhalt des statischen Exchange-Buffer als Stringtable zurück

StringTable-Verarbeitung

Um einfache Stringtabellen zu verarbeiten, besteht die Möglichkeit, die Push- und Pop-Methoden in eine Iteration einzubinden. Hier ein Beispiel mit "PushDynamicData(*_Str*)":

```
function PushDynTable(_t)
  for k,v in pairs(_t) do _SpDrS60_Exchange.PushDynamicData(v) end;
end;

local MyStrTbl = {"eins","zwei","drei"};

PushDynTable(MyStrTbl);
```