JSON Configuration File Format

With the implementation of the Common Component Specification (CCS) compliant AppServerStartupManager we have introduced a new JSON based session configuration file format. The file supports inheritance (a customer's specific configuration may be inheriting defaults provided by Consultingwerk). The file format supports configuration of:

- service.xml Files loaded during the session startup
- factory.xml Files providing configuration to the default IFactory implementation
- CCS Managers loaded by the IStartupManager
- Database Aliases
- Name of the Application Config file (.applicationsettings, .restapplicationsettings)
- Static Properties (e.g. FrameworkSettings:DebugMode)
- Custom Log Entries
- SESSION:EXPORT

The AppServerStartupManager uses exactly one configuration file during session initialization. But this configuration file may be "basedOn" another configuration file, which recursively may also be based on another configuration file.

Configuration objects like the list of the services.xml files, CCS managers, database aliases, static properties and custom log entries may be set only in a configuration file high up in the stack or be inherited from the configuration it is based on. In which case array entries or nested properties are merged.

Sample configuration files

Consultingwerk/Framework/Server/startup_rest_appserver.json

```
{
        "basedOn": "",
        "inheritStaticProperties": false,
        "staticProperties": {},
        "applicationSettings": ".restapplicationsettings",
        "inheritLoadServices": false,
        "loadServices": [
        "Consultingwerk/Framework/Server/rest_services.xml",
        "Consultingwerk/SmartFramework/services_server.xml"],
    "loadFactories": [
        "Consultingwerk/Framework/factory.xml"],
        "inheritManagers": false,
        "managers": {
        "Ccs.Common.IServiceManager": "Consultingwerk.Framework.CcsServiceManager"},
    "inheritCustomLogEntries": false,
    "customLogEntries": [],
        "inheritAliases": false,
        "aliases": null,
        "sessionExport":  null
}
```

Consultingwerk/SmartComponentsDemo/Web2/startup_rest_appserver.json

```
{
        "basedOn": "Consultingwerk/Framework/Server/startup_rest_appserver.json",
        "inheritStaticProperties": true,
        "staticProperties": {
                "Consultingwerk.Framework.FrameworkSettings:DebugMode": true,
                "Consultingwerk.OERA.DataAccess:LogFetchDataDetails": true
        },
        "inheritLoadServices": true,
        "loadServices": [
            "Consultingwerk/SmartComponentsDemo/Web2/demo_services.xml",
        "Consultingwerk/OERA/TableStatistics/services_request_monitor.xml"],
    "inheritCustomLogEntries": true,
    "customLogEntries": [
        "ServiceInterface",
        "DataAccess",
        "ServiceLoader",
        "ConfigurationProvider",
        "ServiceNameMappingService",
        "SmartWebHandlerRequest",
        "SmartHybridRealm",
        "SmartRepositoryService",
        "ContextDatasetStore"],
    "inheritAliases": true,
    "aliases": {
            "sports2000": ["dictdb", "appdb"],
            "icfdb": "afdb"
    }
}
```

Resulting configuration

```
{
  "staticProperties": {
    "Consultingwerk.Framework.FrameworkSettings:DebugMode": true,
    "Consultingwerk.OERA.DataAccess:LogFetchDataDetails": true
  },
  "applicationSettings": ".restapplicationsettings",
  "loadServices": [
    "Consultingwerk\/Framework\/Server\/rest_services.xml",
    "Consultingwerk\/SmartFramework\/services_server.xml",
    "Consultingwerk\/SmartComponentsDemo\/Web2\/demo_services.xml",
    "Consultingwerk\/OERA\/TableStatistics\/services_request_monitor.xml"
  ],
  "loadFactories": [
    "Consultingwerk/Framework/factory.xml"],
  "managers": {
    "Ccs.Common.IServiceManager":
  },
  "customLogEntries": [
    "ServiceInterface",
    "DataAccess",
    "ServiceLoader",
    "ConfigurationProvider",
    "ServiceNameMappingService",
    "SmartWebHandlerRequest",
    "SmartHybridRealm",
    "SmartRepositoryService",
    "ContextDatasetStore"
  ],
  "aliases": {
    "sports2000": [
      "dictdb",
      "appdb"
    ],
    "icfdb": "afdb"
  },
  "sessionExport": null
}
```

Application Settings

The Application Settings (ConfigurationProvider) can be configured in two ways:

- Either as a single JSON string value, providing the name of the JSON configuration file (e.g. .applicationsettings or .restapplicationsettings)
- As a JSON Object, providing name/value pairs with the configuration values, basically embedding the .applicationsettings/.restapplicationsettings file into the JSON configuration file.

Services

The loadServices array can define services in two ways, combinations are supported:

- as JSON Strings referencing the names of service.xml files
- as JSON Objects directly referencing services, see below. Services in the JSON Object configuration in one JSON Configuration file may override settings for the same Interface in a base JSON Configuration file thus allowing overriding individual service definitions - which is not supported by service.xml files.

```
"loadServices": [
  {
    "Package.Subpackage.IInterfaceName1": "Package.Subpackage.ServiceImplementation1",
    "Package.Subpackage.IInterfaceName2": "Package.Subpackage.ServiceImplementation2"
  }
]
```

AppServer Startup

The procedure *Consultingwerk/Framework/Server/ccs_startup.p* implements an AppServer startup based on the Common Component Specification (CCS) compliant AppServerStartupManager.

The AppServer session startup procedure parameter points to the JSON Configuration file.