支持数据集的新App框架SDK接口设计

一 支持数据集的App框架简述

支持数据集的App框架配置接口主要包括2部分：数据集的配置和App的profile的配置。

1、数据集的配置

App要支持以数据集形式配置输入参数，因此要首先配置数据集，默认数据集为设备+资源名称的一个集合，每个集合成员由设备名+资源名唯一标识。一个数据集可以支持多个设备，每个设备内支持不少于一个资源。因此配置数据集就是列出设备名称和设备下面所有的资源列表，从中选择需要的设备和资源名。配置数据集时应该添加描述注明数据集的内容、特征、面向的需求方等等，以便使用时通过描述挑选数据集。

2、App profile的配置

参考原C++驱动的App的profile，支持数据集的App的profile需要支持以数据集形式的输入参数，同时保留对设备+资源名的引用形式的数据资源的输入参数的支持。增加数据类型的定义，支持DataSet类型。

连接参数设置暂时保留原配置格式，暂不考虑手动配置，其中input由sdk通过zmq发送和接收验证，output和parameter等同样通过sdk提供接口进行查询和配置，所以用户不需要关注连接参数的配置，暂定不通过用户接口配置。

Output部分的配置和Parameter部分的配置暂不变动，由sdk内部增加接口支持查询和参数的配置。

二 SDK初始化接口

接口定义：func NewNSPlusSdk(appNameKey string) (\*NSPlusSdk, error)

调用示例：

var err error  
nsplusdemo.NsplusSdk, err = nsplus.NewNSPlusSdk(“demo”)  
if err != nil {  
 logBeforeInit(err)  
}

参数说明：

返回说明：如果sdk初始化失败，则返回error，否则，返回正常的sdk指针；

三 回调函数设置接口

定义三个回调函数，分别用于接收input数据，接收parameter更新，初始化app自定义的业务client，定义如下：

**type** InputRecvCallback **func**(recData \*DataDistroWithIndex) error  
  
**type** ParameterRecvCallback **func**(recvParam \*AppParams) error  
  
**type** InitClientsCallback **func**() error

设置回调函数接口定义：

func (sdk \*NSPlusSdk) SetInitClientsCBFunc(f InitClientsCallback)

func (sdk \*NSPlusSdk) SetInputRecvCBFunc(f InputRecvCallback)

func (sdk \*NSPlusSdk) SetParameterRecvCBFunc(f ParameterRecvCallback)

参数说明：参数为函数，参考调用实例

返回说明：

调用实例：如定义如下三个处理函数：

**func** InputCBFuns(recvParam \*nsplus.DataDistroWithIndex) error {  
 NsplusSdk.LoggingClient.Error(fmt.Sprintf(**"nnnnnnnnnnn InputCBFuns recv data[%v]"**, recvParam))  
 *//close(errChan)* **return** nil  
}  
**func** ParameterCBFuns(recvParam \*nsplus.AppParams) error {  
 NsplusSdk.LoggingClient.Error(fmt.Sprintf(**"mmmmmmmmmmm ParameterCBFuns recv data[%v]"**, recvParam))  
 *//close(errChan)* **return** nil  
}  
**func** InitClientsCBFuns() error {  
 NsplusSdk.LoggingClient.Error(fmt.Sprintf(**"kkkkkkkkkk InitClientsCBFuns now time=[%v]"**, time.Now()))  
 *//close(errChan)* **return** nil  
}

则设置回调函数为：

internal.NsplusSdk.SetInitClientsCBFunc(internal.InitClientsCBFuns)  
internal.NsplusSdk.SetInputRecvCBFunc(internal.InputCBFuns)  
internal.NsplusSdk.SetParameterRecvCBFunc(internal.ParameterCBFuns)

注：设置回调函数需要在NewNSPlusSdk()调用之后，在SdkRun()执行之前调到

四 Input、Output、Parameter查询接口

4.1 Input查询接口

接口定义：(sdk \*NSPlusSdk)GetInput(pName string) []AppInputs

调用示例：sdk.GetInput("inputname")

参数说明：pName为空的时候查询所有Input，否则按名称查询；

返回说明：返回数组形式[]AppInputs，如果数组长度为0，表示无结果或查询异常，其中，AppInputs的定义如下：

**type** AppInputs **struct**{  
 Param InputsInfo **`json:"param"`** *// input\output parameter define* DevResValues **map**[string]ResValues **`json:"devResValues"`** *// key devname, value res name and values*}

**type** ResValues **struct**{  
 ResValues **map**[string]ValueDefine **`json:"resValues"`** *// key resname, value res type and values*}

**type** ValueDefine **struct**{  
 Value string **`json:"value"`** Type string **`json:"type"`**}

4.2 Output查询接口

接口定义：(sdk \*NSPlusSdk)GetOutput(pName string) []AppOutputs

调用示例：sdk.GetOutput("paraName")

参数说明：pName为空的时候查询所有Output，否则按名称查询；

返回说明：返回数组形式[]AppOutputs，如果数组长度为0，表示无结果或查询异常，其中，AppOutputs的定义如下：

**type** AppOutputs **struct**{  
 Param OutputsInfo **`json:"param"`** *// output define* Values ValueDefine **`json:"values"`** *// value and type*}

**type** ValueDefine **struct**{  
 Value string **`json:"value"`** Type string **`json:"type"`**}

4.3 Parameters查询接口

接口定义：(sdk \*NSPlusSdk)GetParams(pName string) []AppParams

调用示例：sdk.GetParams("paraName")

参数说明：pName为空的时候查询所有Parameter，否则按名称查询；

返回说明：返回数组形式[]AppParams，如果数组长度为0，表示无结果或查询异常，其中，AppParams的定义如下：

**type** AppParams **struct**{  
 Param ParameterInfo **`json:"param"`** *// parameter define* Values ValueDefine **`json:"values"`** *// value and type*}

**type** ValueDefine **struct**{  
 Value string **`json:"value"`** Type string **`json:"type"`**}

五 Output输出接口

4.1 业务Port查询接口

接口定义：func (sdk \*NSPlusSdk)SendAppOutput(data []NameValues) error

调用示例：Port := sdk.GetPortConfig ()

参数说明：参数为输出的Name-Value数组

type NameValues struct {  
 Name string `json:"name"`  
 Value interface{} `json:"value"`  
}

返回说明：返回错误信息

调用实例：

Outputs := make([]NameValues, 0)  
for \_, param := range sdk.AppParamsCtrl.Output {  
 var onepara NameValues  
 onepara.Name = param.Name  
 onepara.Value = fmt.Sprintf("%v\_%v", "outValue", sdk.AppParamsCtrl.OutputSqid)  
  
 log.Printf("debugOutputSendLoop for Output [%v] ", onepara)  
 Outputs = append(Outputs, onepara)  
}  
  
err := sdk.SendAppOutput(Outputs)  
if err != nil {  
 log.Printf("SendAppOutput for Output [%v] fail, error=[%v] ", Outputs, err.Error())  
}

六 业务Port和挂载路径查询接口

4.1 业务Port查询接口

接口定义：(sdk \*NSPlusSdk)GetPortConfig() string

调用示例：Port := sdk.GetPortConfig ()

参数说明：nil；

返回说明：返回profile中设定的端口号

4.2 业务路径查询接口

接口定义：(sdk \*NSPlusSdk)GetDirConfig() []string

调用示例：Paths := sdk.GetDirConfig()

参数说明：nil；

返回说明：返回profile中设定的业务路径（应为绝对路径），可能是多个

附件

原C++sdk的app profile定义规范



Profile实例

