文件保密等级	[V]对外开放
	[ ]对外保密
	[ ]对内保密

接口大师
接口插件开发技术手册

版本：20200104

目录

1 开发环境

- Windows 10
- Visual Studio 2015 及以上
- .Net Framework 4.5.2 及以上

2 接口说明

2.1 命名空间 ioMaster.Intf

- 从来源读取数据接口 IGetJsonData

```
// 根据 webDsDataEntity 中的连接设置和 formEntity 中的表单设置，
// 读取数据后并以传输实体对象形式返回
public TranReadEntity GetJsonData(WebDsDataEntity webDsDataEntity,
WebFormDataEntity formEntity)
```

- 写数据到目标接口 ISaveJsonData

```
// 根据 webDsDataEntity 中的连接设置和 formEntity 中的表单设置，
// 把 saveReadEntity 中的结果写入目标，并把执行结果以日志对象形式返回
public LogEntity SaveJsonData(TranReadEntity saveReadEntity
                              , WebDsDataEntity webDsDataEntity
                              , WebFormDataEntity formEntity
                              , FieldMapEntity[] fieldMaps)
```

注意：以上两个方法的传入参数已在接口大师界面上维护好，仅供读取，无需修改。

- PostgreSql、Oracle、SQL Server 的读写已实现，并内置于程序中。
- 京极的上传已实现。
- 向金蝶 K3 Cloud 的回写需引用 `Kingdee.K3Cloud.dll`，根据实际需求实现各种表单的读写方法。具体用法见后面的示例。

2.2 命名空间 ioMaster.BO.Web

- `WebDsDataEntity`，数据连接配置实体类,一般供调用者读取。

属性名称	数据类型	描述
ds_id	string	数据源 ID
ds_name	string	数据源名称
ds_class	string	数据源分类,如 PostgreSql、SQL Server
IntfID	string	接口任务 ID
sys_id	string	系统 ID
sys_host	string	服务器地址,如数据库服务器的 IP、金蝶云接口的 URL 等
sys_port	string	端口,如数据库服务器的端口、
sys_user_no	string	登陆数据库或 URL 的用户名,如金蝶云账号、京极的用户 ID
sys_user_pswd	string	登陆数据库或 URL 的密码,如金蝶云密码、京极的密钥
sys_db_name	string	数据库实例名、京极的企业 ID、
sys_type_no	string	系统类型编号,0 标准,1 自定义
sys_type_name	string	系统类型名称,如标准、自定义
sys_full_name	string	系统名称,如安达发 Ax10
sys_class	string	系统分类,如 CRM、SRM、MES 等
per_read	int	每次读取多少条,如 100
per_save	int	每次写入多少条,如 20

- WebDsDataEntity 主要参数示例

属性名称	Postgre Sql	SqlServ er	金蝶 K3 Cloud	京极
sys_host	192.168. 1.220	192.168. 1.220	http://192.168.88. 5/K3cloud	https://www.56008.c om/yun/api
sys_port	5432	1433	2052,指界面为简体中文	无
sys_user_ no	postgres	sa	zhangsan	5de9baa8a71ec10ab c2fa123
sys_user_ pswd	123321	abcabc	888888	xoaeoa4qQ459adrjd aR
sys_db_n ame	srm_db	crm_db	5cb23dfeeff865	5de9b234dsooao10 e506d0004

- WebFormDataEntity ,表单实体类,包含表单的配置信息,一般供调用者读取。字段属性说明如下:

属性名称	数据类型	描述
xt_id	string	表单所在的系统 ID
FormId	string	表单 ID
FormType	string	表单类型,如 dll、sql
FormName	string	表单名称，如简单生产入库单
FullFormName	string	表单全称，如 金蝶 K3 Cloud 7.2 简单生产入库单
dllming	string	dll 名称,如果表单类型为 dll 时，此处为 dll 的名称及类名,格式为:dll 文件名(不含扩展名)/类名。如 JingJi/JingJiUpStd
IsSqlForm	bool	是否 SQL 表单,没有 dll 名称则是 SQL 表单
duqusql	string	当表单是 SQL 类型,且作为读取数据来源时的 SQL 语句。 如 :select mo_id,mo_no,part_no,upd_time from mo
xietable	string	写入表名。当表单是 SQL 类型,且作为写入目标时,写入的目标(临时)表名

属性名称	数据类型	描述
xieccgc	string	存储过程名。当表单是 SQL 类型,且作为写入目标时,写入目标表完成之后,调用此存储过程对数据进行再处理
beizhu	string	备注,表单的附加注释信息
fenlei	string	表单的自定义分类,如仓储、设备、品质
wb_bd_id	string	外部表单 ID。如金蝶的表单 ID ENG_BOM、京极的接口 ID:5cb82d08a71ec12f747c7941 等
pk_field	string	主键字段名称。当表单是 SQL 类型,且作为读取数据来源时的主键字段名。如 mo_id
update_field	string	更新时间字段名称。当表单是 SQL 类型,且作为读取数据来源时的更新时间字段。如 upd_time
dllmulu	string	第三方开发 dll 文件目录
dllurl	List<DllUrlEntity>	第三方插件文件列表

2.3 命名空间 ioMaster.BO.Pub

- FieldMapEntity,字段映射实体类

属性名称	数据类型	描述
is_enabled	string	是否启用。0 未启用, 1 已启用
from_field_name	string	来源字段名
to_field_name	string	目标字段名

- TranReadEntity，数据传输实体类，用于存放读取到的结果

属性名称	数据类型	描述
Code	string	返回代码
Message	string	返回信息
ResultTable	DataTable	读取到的结果(数据表格式)
JsonData	string	读取到的结果(Json 格式)

属性名称	数据类型	描述
Exception	Exception	返回的异常对象

2.4 命名空间 ioMaster.Up

- DllUrlEntity ,表单第三方插件文件实体类

属性名称	数据类型	描述
type	int	是否主程序。0 主程序 1 不是主程序
url	string	dll 的下载地址
filename	string	dll 文件名

2.5 命名空间 ioMaster.Log

- LogEntity , 日志实体类

属性名称	数据类型	描述
LogId	string	日志 ID
IntfId	string	任务 ID
FromSysId	string	来源系统 ID
FromSysName	string	来源系统名称
FromFormId	string	来源表单 ID
FromFormName	string	来源表单名称
ToSysId	string	目标系统 ID
ToSysName	string	目标系统名称
ToFormId	string	目标表单 ID
ToFormName	string	目标表单名称
CodeNo	string	返回代码
Msg	string	返回信息
LogLevel	string	日志级别。 分为 debug、info、error、warn、fatal。默认为

属性名称	数据类型	描述
		info
IsSuccessful	bool	是否成功。默认当 Code 为'0'时表示成功
IsMemDbLog	bool	是否在界面日志中显示
IsWebLog	bool	日志是否上传到服务器
RowCount	int	同步数据行数
DataSize	long	同步数据大小
NetFlow	long	同步数据所用流量
IsPosted	int	日志是否已上传
Exception	Exception	异常对象
ResultObject	object	用于存放结构复杂的返回结果，便于分析

- AppLog，日志帮助类

```
// 根据 logEntity 中的设置，输出日志内容
public static void Info(LogEntity logEntity)
```

示例：

- 输出"Hello World!"

```
AppLog.Info(new LogEntity(){Msg="Hello World!"})
```

- 在 try…catch…中使用

```
try{
// do something
return new LogEntity(){
  CodeNo="0",
  Msg="执行成功"
```

```
}
} catch(Exception ex){
return new LogEntity(){
  CodeNo="-1",
  Msg = $"异常:{ex.Message}",
  Exception = ex
}
}
```

2.6 命名空间 ioMaster.Helper

- JsonHelper ,Json 帮助类

```
// 把对象序列化后，返回 Json 格式字符串
public static string SerializeObject(object o)
// 把 Json 格式的字符串反序列化成 T 类型对象实例
public static T DeserializeJsonToObject<T>(string json) where T : class
```

2.7 命名空间 Kingdee.K3Cloud.BO.pub

- UserIdField 类，对应 Json 格式 API 中带 FUserID 属性的字段,如

```
"FForceCloserId": {
  "FUserID": ""
}
```

在 C#中定义为

```
public UserIdField FForceCloserId{get;set;}
```

- NumberField 类，对应 Json 格式 API 中带 FUserID 属性的字段,如

```
"FStockerId":{
  "FNUMBER":""
}
```

在 C#中定义为

```
public NumberField FStockerId{get;set;}
```

3 示例

某企业需要从安达发 Ax 中读取数据，然后写入到金蝶 K3 Cloud 7.2 中的工单工时登记单。开发步骤如下：

一、在金蝶 K3 Cloud 中搜索 API，找到 工单工时登记单，查看 保存 方法的 Json 格式。

类似如下:

```
{
    "Creator":"",
    "NeedUpDateFields":[],
    "NeedReturnFields":[],
    "IsDeleteEntry":"true",
    "SubSystemId":"",
    "IsVerifyBaseDataField":"false",
    "IsEntryBatchFill":"true",
    "ValidateFlag":"true",
    "NumberSearch":"true",
    "InterationFlags":"",
    "IsAutoSubmitAndAudit":"false",
    "Model":{
```

```
        "FID": 0,
        "FBillNo": "",
        "FCreatorId": {
            "FUserID": ""
        },
        "FCreateDate": "1900-01-01",
        "FModifierId": {
            "FUserID": ""
        },
        "FModifyDate": "1900-01-01",
        "FAPPROVEDATE": "1900-01-01",
        "FAPPROVERID": {
            "FUserID": ""
        },
        "FDate": "1900-01-01",
        "FPrdOrgId": {
            "FNumber": ""
        },
        "F_ora_Text": "",
        "FSumFProductionhour": 0,
        "FEntity": [
            {
                "FEntryID": 0,
                "FWORKSHOPID": {
                    "FNUMBER": ""
                },
                "FProductionhour": 0,
                "FMaterialId": {
                    "FNUMBER": ""
                },
                "FPRODUCTNO": ""
            }
        ]
    }
}
```


二、根据字段说明及 Json 格式，可以看出大致分为三段：

- **保存选项部分**
- **单据头 FBillHead，也就是 Model**
- **单据体 FEntity，即表身明细。这是一个 Json 数组**

三、新建 dll 项目。根据 Json 格式编写 C#类，便于后续以对象方式对数据进行操作，再把对象序列化为 Json 格式。

1、通过观察各种表单保存 API 发现，所有选项都是一样的，因此这部分可定义成一个通用类，以后所有金蝶 K3 Cloud 的表单保存功能都从此类继承。代码示例如下(为缩小篇幅，省略部分代码):

```
// StdHeaderFormSave.cs
using System.Collections.Generic;

namespace Kingdee.K3Cloud.BO.pub
{
    public class StdHeaderFormSave
    {
        public StdHeaderFormSave()
        {
            IsDeleteEntry = "true";
            // 略
            NeedUpDateFields = new List<string>().ToArray();

        }
        /// <summary>
        /// 需要更新的字段，数组类型，格式：[key1,key2,...]（非必录） 注（更新单据体字段得加上单据体 key）
        /// </summary>
        public string[] NeedUpDateFields { get; set; }
        /// <summary>
        /// 是否删除已存在的分录，布尔类型，默认 true（非必录）
        /// </summary>
        public string IsDeleteEntry { get; set; }
        // 略
    }
}
```

2、虽然手上只有保存方法的 API，但我们一般采用批量保存的方法。批量保存与保存的区别在于 Json 格式：保存时，Model 为单个对象；在批量保存中，Model 为对象数组。以下为批量保存的类对象代码(请仔细观察 C#类和 Json 格式是如何对应的):

```
using System;
using System.Collections.ObjectModel;
using Kingdee.K3Cloud.BO.pub;

namespace Kingdee.K3Cloud.FormEntity
{
```

```
/// <summary>
/// 工单工时登记单 BillHour_Registry
/// </summary>
public class BH_Reg_BatchSave : StdHeaderFormSave
{
    public ObservableCollection<BH_Reg_Save_FBillHead> Model { get; set; }
    public BH_Reg_BatchSave()
    {
        Model = new ObservableCollection<BH_Reg_Save_FBillHead>();
    }
}

// 单据头
public class BH_Reg_Save_FBillHead : ICloneable
{
    public BH_Reg_Save_FBillHead()
    {
        FCreatorId = new UserIdField();
        FPrdOrgId = new NumberField();
        FEntity = new ObservableCollection<BH_Reg_Save_FEntity>();
    }
    public object Clone()
    {
        BH_Reg_Save_FBillHead billHead = (BH_Reg_Save_FBillHead)MemberwiseClone();
        billHead.FCreatorId = (UserIdField)FCreatorId.Clone();
        billHead.FPrdOrgId = (NumberField)FPrdOrgId.Clone();
        billHead.FEntity = new ObservableCollection<BH_Reg_Save_FEntity>();
        return billHead;
    }

    /// <summary>
    /// 实体主键
    /// </summary>
    public int FID { get; set; }

    /// <summary>
    /// 单据编号(必填项)
    /// </summary>
    public string FBillNo { get; set; }

    /// <summary>
    /// 创建人
    /// </summary>
    public UserIdField FCreatorId { get; set; }
```

```
/// <summary>
/// 创建日期
/// </summary>
public string FCreateDate { get; set; }

/// <summary>
/// 日期
/// </summary>
public string FDate { get; set; }

/// <summary>
/// 生产组织(必填项)
/// </summary>
public NumberField FPrdOrgId { get; set; }

/// <summary>
/// 总人数
/// </summary>
//public int FSumpeople { get; set; }

/// <summary>
/// 总工时(小时)
/// </summary>
public double FSumFProductionhour { get; set; }

/// <summary>
/// 数据状态
/// </summary>
//public string FDocumentStatus { get; set; }

/// <summary>
/// 修改人
/// </summary>
//public string FModifierId { get; set; }

/// <summary>
/// 修改日期
/// </summary>
//public string FModifyDate { get; set; }

/// <summary>
/// 审核日期
/// </summary>
```

```
        //public string FAPPROVEDATE { get; set; }

        /// <summary>
        /// 审核人
        /// </summary>
        //public string FAPPROVERID { get; set; }

        /// <summary>
        /// 备注
        /// </summary>
        //public string F_ora_Text { get; set; }

        public ObservableCollection<BH_Reg_Save_FEntity> FEntity { get; set; }
        public string hid { get; set; }
    }

    // 单据明细
    public class BH_Reg_Save_FEntity : ICloneable
    {
        public BH_Reg_Save_FEntity()
        {
            FWORKSHOPID = new NumberField();
            FMaterialId = new NumberField();
        }
        public object Clone()
        {
            BH_Reg_Save_FEntity entity = (BH_Reg_Save_FEntity)MemberwiseClone();
            entity.FWORKSHOPID = (NumberField)FWORKSHOPID.Clone();
            entity.FMaterialId = (NumberField)FMaterialId.Clone();
            return entity;
        }
        /// <summary>
        /// 实体主键
        /// </summary>
        public int FEntryID { get; set; }

        /// <summary>
        /// 车间,必须填
        /// </summary>
        public NumberField FWORKSHOPID { get; set; }

        /// <summary>
        /// 工时(小时)
        /// </summary>
```

```
        public double FProductionhour { get; set; }

        /// <summary>
        /// 物料编码,必须填
        /// </summary>
        public NumberField FMaterialId { get; set; }

        /// <summary>
        /// 生产编号,必须填
        /// </summary>
        public string FPRODUCTNO { get; set; }

        /// <summary>
        /// 计件工资
        /// </summary>
        public double FPieceworkwage { get; set; }
    }
}
```

3、因为要往金蝶回写数据，所以要实现 ISaveJsonData 接口中 SaveJsonData 方法。代码如下：

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using ioMaster.BO.Pub;
using ioMaster.BO.Web;
using ioMaster.Intf;
using ioMaster.Log;
using ioMaster.Res;
using Kingdee.K3Cloud;
using Kingdee.K3Cloud.FormEntity;
using Kingdee.K3Cloud.Common;
using Newtonsoft.Json;

// 一定要写在 ioMaster.Plugins 命名空间中
namespace ioMaster.Plugins
{
    /// <summary>
    /// 员工计件工资 BillHour_Registry
    /// </summary>
    public class BillHour_Reg_WB : ISaveJsonData
    {
```

```
        public LogEntity SaveJsonData(TranReadEntity saveReadEntity, WebDsDataEntity
webDsDataEntity, WebFormDataEntity formEntity, FieldMapEntity[] fieldMaps)
        {
            // 如果想查看以上参数内容，可以输出到日志
            // AppLog.Info(new LogEntity()
            // {
            //       Msg = $"saveReadEntity: {JsonConvert.SerializeObject(saveReadEntity)}"
            // });
            return K3CloudBatchSave(saveReadEntity, webDsDataEntity, formEntity, fieldMaps);
        }

        private K3CloudHttpClient k3CloudHttpClient;
        public LogEntity K3CloudBatchSave(TranReadEntity saveReadEntity, WebDsDataEntity
webDsDataEntity,
            WebFormDataEntity formEntity, FieldMapEntity[] fieldMaps)
        {
            try
            {
                DataTable dtSource = saveReadEntity.ResultTable;

                string strHostUrl = webDsDataEntity.sys_host;
                string strUserName = webDsDataEntity.sys_user_no;  // 账号
                string strDbName = webDsDataEntity.sys_db_name;     // 账套 ID
                string strUserPswd = webDsDataEntity.sys_user_pswd;// 密码
                string strPort = webDsDataEntity.sys_port;
                int lcid = Convert.ToInt32(strPort); // 语言，一般是 2052，简体中文

                string formId = "abbc4bf4861254daa9046bex4c7b1e70";// 表单 ID
                string strFilter = string.Empty;                    // 过滤条件
                string strFieldKeys = formEntity.xieruziduan;       // 逗号隔开的字段名
"FName,FShortName"。
                string strResult = string.Empty;

                string strPkField = formEntity.pk_field;
                if (string.IsNullOrEmpty(strPkField))
                    strPkField = "hid";

                //登录
                if (null == k3CloudHttpClient)
                {
                    k3CloudHttpClient = new K3CloudHttpClient(strHostUrl, strDbName,
strUserName, strUserPswd, lcid);
                    bool bOk = k3CloudHttpClient.ValidateUser();
```

```
                if (!bOk)
                    return new LogEntity
                    {
                        CodeNo = StrRes.s_failure,
                        Msg = $"失败:{k3CloudHttpClient.CheckMsg}"
                    };
            }

            BH_Reg_BatchSave tmpSave = new BH_Reg_BatchSave();
            #region 把数据写入临时对象
            foreach (DataRow dr in dtSource.Rows)
            {
                BH_Reg_Save_FBillHead billHead = new BH_Reg_Save_FBillHead();
                //表头字段。
                billHead.hid = dr[strPkField].ToString();
                billHead.FDate = dr["FDate"].ToString();
                billHead.FBillNo = dr["FBillNo"].ToString();                    // 单据编号
                billHead.FPrdOrgId.FNumber = dr["FPrdOrgId"].ToString();        // 生产
组织,即工厂编号
                //billHead.FCreatorId.FUserID = Convert.ToInt32(dr["FUserID"]);     // 创建
人，非必填，略

                BH_Reg_Save_FEntity billBody = new BH_Reg_Save_FEntity();
                billBody.FWORKSHOPID.FNumber = dr["FWORKSHOPID"].ToString();
// 车间
                billBody.FProductionhour = Convert.ToDouble(dr["FProductionhour"]);// 工
时(小时)
                billBody.FMaterialId.FNumber = dr["FMaterialId"].ToString();    // 物料编
码
                billBody.FPRODUCTNO = dr["FPRODUCTNO"].ToString();              //
生产编号
                billBody.FPieceworkwage = Convert.ToDouble(dr["FPieceworkwage"]);    //
计件工资
                billHead.FEntity.Add(billBody);
                tmpSave.Model.Add(billHead);
            }
            #endregion 把数据写入临时对象

            BH_Reg_BatchSave objSave = new BH_Reg_BatchSave();
            #region 临时对象转为正式对象;
            BH_Reg_Save_FBillHead tmpHead = tmpSave.Model.FirstOrDefault();
            BH_Reg_Save_FBillHead head = null;
            if (null != tmpHead)
            {
```

```
                    head = (BH_Reg_Save_FBillHead) tmpHead.Clone();
                    List<BH_Reg_Save_FBillHead> allModel = tmpSave.Model.Where(r => r.hid
== head.hid).ToList();
                    foreach (BH_Reg_Save_FBillHead objHead in allModel)
                    {
                        foreach (BH_Reg_Save_FEntity objEntity in objHead.FEntity)
                            head.FEntity.Add((BH_Reg_Save_FEntity) objEntity.Clone());
                    }

                    objSave.Model.Add(head);
                }
                #endregion 临时对象转为正式对象

                string strSaveJson = JsonConvert.SerializeObject(objSave);

                strResult = k3CloudHttpClient.Execute(formId, strSaveJson,
K3CloudActionEnum.BatchSave);
                if (strResult.Contains("IsSuccess\":true"))
                    return new LogEntity
                    {
                        CodeNo = StrRes.s_ok,
                        Msg = "成功",
                        ResultObject = strResult
                    };
                return new LogEntity
                {
                    CodeNo = StrRes.s_failure,
                    Msg = $"失败",
                    ResultObject = strResult
                };
            }
            catch (Exception ex)
            {
                return new LogEntity
                {
                    CodeNo = StrRes.s_failure,
                    Exception = ex
                };
            }
        }
    }
}
```

4、因为安达发 Ax 使用的是 PostgreSql，读写功能已实现并内置于程序中，只需要在接口大师界面上配置即可，无需另外开发，无需实现 IGetJsonData 中的 GetJsonData 方法。具体如何配置请自行查阅接口大师帮助文档。

5、至此，回写插件开发完毕。

4 发布