

Softland Software & Service

芍园IRAP制造卓越支撑平台

WEB API接口规范

━━OPC系统分册

Web API Specification for Softland IRAP-MESP

OPC System

（版本号：V6.02）

江苏芍园科技有限责任公司

（本稿完成日期：?）

**前 言**

本规范是依据标准化基本原理和方法结合制造业信息化实际而制定的。

本规范规定了IRAP-MESP (Manufacturing Excellence Support Platform)的OPC系统对外提供数据交换并实现自动化控制的统一接口。

本规范由江苏芍园科技有限公司具体承担制定并负责解释。

本规范的版本修订历史如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 主要更改内容 | 更改日期 | 更改人 | 更改页号 |
| 6.01 | 首次发行 | 2017-12-20 | 李智颖 | All |
| 6.02 | 增加“设备指定标签数据回写并读取”接口；  修改调用地址 | 2018-2-23 | 李智颖 | 20-21  12 |
| 6.03 | 修改“GetDevices”和“GetDeviceTags”接口；  停用“GetKepServList”、 “UpdateKepServ”、 “UpdateDeviceTags”接口 | 2018-11-5 | 李智颖 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 总则 1](#_Toc529190303)

[1.1 适用范围 1](#_Toc529190304)

[1.2 规范引用文件 1](#_Toc529190305)

[1.3 术语和定义 1](#_Toc529190306)

[1.3.1 Web Service 2](#_Toc529190307)

[1.3.2 RESTful 4](#_Toc529190308)

[1.3.3 ESB 6](#_Toc529190309)

[2 设计原则 9](#_Toc529190310)

[2.1 数据类型定义 9](#_Toc529190311)

[2.2 标识项定义 9](#_Toc529190312)

[2.2.1 工序代码 Operation Code 10](#_Toc529190313)

[2.2.2 在制品站位代码 WIP Station Code 10](#_Toc529190314)

[2.2.3 在制品标识代码 WIP ID Code 11](#_Toc529190315)

[2.3 接口概述 11](#_Toc529190316)

[3 接口说明 12](#_Toc529190317)

[3.1 接口属性 12](#_Toc529190318)

[3.1.1 调用地址 12](#_Toc529190319)

[3.2 系统信息接口 System Information 12](#_Toc529190320)

[~~3.2.1~~ ~~获取KepwareServer列表 GetKepServList~~ 12](#_Toc529190321)

[~~3.2.2~~ ~~更新KepwareServer UpdateKepServ~~ 13](#_Toc529190322)

[3.3 设备信息接口 Device Information 15](#_Toc529190323)

[3.3.1 获取设备列表 GetDevices 15](#_Toc529190324)

[3.3.2 获取设备标签（Tag）列表 GetDeviceTags 16](#_Toc529190325)

[~~3.3.3~~ ~~更新设备标签 Update Device Tags~~ 18](#_Toc529190326)

[3.3.4 设备指定标签数据读取与回写 DeviceTagValueRW 20](#_Toc529190327)

[4 附录 23](#_Toc529190328)

[4.1 预定义的错误信息 23](#_Toc529190329)

# 总则

## 适用范围

本规范规定了IRAP-制造卓越支撑平台(IRAP-MESP)的OPC系统对外提供数据交换并实现自动化控制的统一网络服务接口。

本规范适用于所有使用IRAP-MESP系统的各类制造企业，包括离散制造、流程制造以及混合型制造。

## 规范引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注明日期的引用文件，其随后所有的修改单（不包括勘误的内容）或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注明日期的引用文件，其最新版本适用于本标准。

江苏芍园IRAP网络接口(IRAP-WebAPI)将Web Service、RESTFul API、Json RPC、WCF以及ESB入队、ESB出队、分布式缓存存取整合成统一的网路服务平台，服务启动、停止、异常均及时反馈到zookeeper服务器；可以分布式部署，支持nginx负载均衡；支持身份认证与数据通信加密；支持跨平台、跨语言接口调用；提供集中配置与集中监控控制台。

有关Web Service、RESTFul API、Json RPC、WCF等有关公共的协议，请从互联网上获取有关标准文档。

## 术语和定义

下列术语和定义适用于本规范。

### Web Service

#### Web Service基本概念

Web service是一个平台独立的，松耦合的、自包含的、基于可编程的web的应用程序，可使用开放的XML标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

因为 Web Service 符合公认的标准，所以能够以这种通常可访问的方式来公开其资源。Web Service 使用的标准技术包括：

通过 WSDL 文件公开描述其自身功能通过 XML 消息（通常使用 SOAP 格式）与其他应用程序进行通信使用标准网络协议，如 HTTP。

#### 什么是WSDL

Web Service 描述语言 (Web Service Description Language，简称WSDL）使用标准 XML 格式描述 Web Service。WSDL 文件描述了特定的Web Service，使其他软件应用程序可以与它进行交互。WSDL 通常可公开访问并且提供了充分的信息，潜在客户端仅根据 WSDL文件即可了解如何来操作此 Web Service。WSDL 文件用于在 Web Service 提供者和使用者之间交换接口信息。通过 WSDL 描述，客户端可以使用 Web Service 的功能，而无需了解Web Service 的实现细节。

WSDL 文件包含了客户端调用 Web Service 的方法所必需的所有信息：用作方法参数或返回值的数据类型单个的方法名和签名（WSDL 将方法称为“操作”）每个方法可以使用的协议和消息格式用来访问 Web Service 的 URL。

#### XML 和 SOAP

可扩展标记语言（Extensible Markup Language，简称 XML）提供了一种公共语言，通过它，不同的应用程序可以在网络上彼此进行交互。大多数 Web Service 通过 XML 进行通信。客户端将包含请求的 XML 消息发送到 Web Service，然后 Web Service 使用包含操作结果的 XML消息进行响应。大多数情况下，这些 XML 消息将根据 SOAP 语法设置格式。

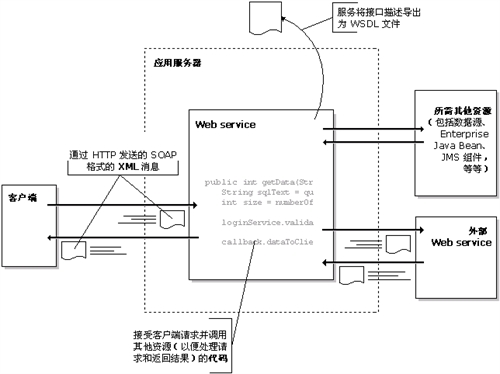
Simple Object Access Protocol (SOAP) 指定了一种标准格式，使应用程序可以调用彼此的方法，在彼此之间传递数据。请注意，Web Service 可以使用非 SOAP 格式的 XML 消息进行通信。特定的 Web Service 支持的消息类型在其 WSDL 文件中进行描述。

#### 网络协议

Web Service 使用广泛使用的协议，如超文本传输协议（HyperText Transfer Protocol，简称 HTTP）和 Java 消息服务（Java Message Service，简称 JMS），接收请求并发送响应。Web Service 可以支持多个协议，它的不同方法可以支持不同的协议。Web Service 支持的协议在 WSDL 文件中进行发布。本规范使用HTTP协议。

#### Web Service 体系结构

下图显示了 Web Service（在中央）、其客户端软件应用程序（在左侧）及其使用的资源（包括数据库、其他 Web Service 等）（在右侧）之间的关系。Web Service 通过使用标准协议（如 HTTP）交换 XML 消息来与客户端和各种资源进行通信。在 WebLogic Server 上部署 Web Service 后，由 WebLogic Server 负责将传入的 XML 消息路由到您编写的 Web Service 代码。Web Service 将导出 WSDL 文件，以描述其接口，其他开发人员可以使用此文件来编写访问此 Web Service 的组件。



### RESTful

#### RESTful基本概念

REST（英文：Representational State Transfer，简称REST）描述了一个架构样式的网络系统，比如 web 应用程序。它首次出现在 2000 年 Roy Fielding 的博士论文中，他是 HTTP 规范的主要编写者之一。在目前主流的三种Web服务交互方案中，REST相比于SOAP（Simple Object Access protocol，简单对象访问协议）以及XML-RPC更加简单明了，无论是对URL的处理还是对Payload的编码，REST都倾向于用更加简单轻量的方法设计和实现。值得注意的是REST并没有一个明确的标准，而更像是一种设计的风格。

在REST中的一切都被认为是一种资源。每个资源由URI标识。使用统一的接口。处理资源使用POST，GET，PUT，DELETE操作类似创建，读取，更新和删除（CRUD）操作。

无状态。每个请求是一个独立的请求。从客户端到服务器的每个请求都必须包含所有必要的信息，以便于理解。通信都是通过展现。例如XML，JSON以状态为角度，提出将状态移植到客户端处理的新思路。 提出一个既适于客户端应用又适于服务端的应用的、统一的Web视图。适合B/S ,C/S, S/S。 HTTP客户端与HTTP服务器之间的差别，对架构来说无所谓。一个软件应可以既充当Web客户端又充当Web服务器，而无须采用两套完全不同的APIs。

提供资源操作方法的统一：POST, GET, PUT, DELETE ，以超文本或超媒体驱动(hypertext/Hypermedia)的状态转移是REST架构核心。 操作带来状态变化，状态转移遍历使用链接导航方式实现。

#### 原则条件

REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。Hypermedia 是应用程序状态的引擎，资源表示通过超链接互联。

#### 分层系统

另一个重要的 REST 原则是分层系统，这表示组件无法了解它与之交互的中间层以外的组件。通过将系统知识限制在单个层，可以限制整个系统的复杂性，促进了底层的独立性。

当 REST 架构的约束条件作为一个整体应用时，将生成一个可以扩展到大量客户端的应用程序。它还降低了客户端和服务器之间的交互延迟。统一界面简化了整个系统架构，改进了子系统之间交互的可见性。REST 简化了客户端和服务器的实现。

#### 统一接口

REST要求，必须通过统一的接口来对资源执行各种操作。对于每个资源只能执行一组有限的操作。以HTTP/1.1协议为例，HTTP/1.1协议定义了一个操作资源的统一接口，主要包括以下内容：

7个HTTP方法：GET/POST/PUT/DELETE/PATCH/HEAD/OPTIONS

HTTP头信息（可自定义）

HTTP响应状态代码（可自定义）

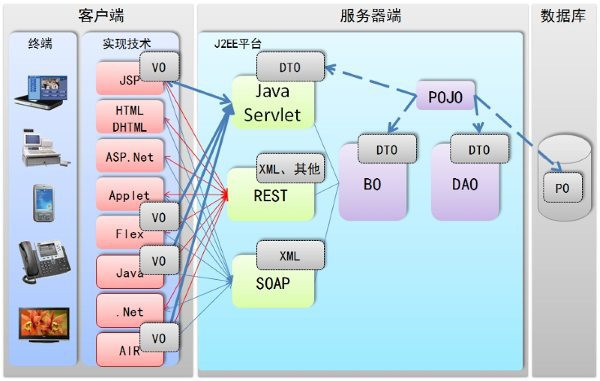
一套标准的内容协商机制

一套标准的缓存机制

一套标准的客户端身份认证机制

REST还要求，对于资源执行的操作，其操作语义必须由HTTP消息体之前的部分完全表达，不能将操作语义封装在HTTP消息体内部。这样做是为了提高交互的可见性，以便于通信链的中间组件实现缓存、安全审计等等功能。 **本文档接口主要使用POST方法**。

#### REST体系结构



### ESB

#### ESB基本概念

ESB全称为Enterprise Service Bus，即企业服务总线。它是传统[中间件技术](http://baike.so.com/doc/6540720.html)与XML、Web服务等技术结合的产物。ESB提供了网络中最基本的连接中枢，是构筑企业神经系统的必要元素。ESB的出现改变了传统的[软件架构](http://baike.so.com/doc/5924712.html)，可以提供比传统中间件产品更为廉价的[解决方案](http://baike.so.com/doc/5901020.html)，同时它还可以消除不同应用之间的技术差异，让不同的应用服务器协调运作，实现了不同服务之间的通信和整合。从功能上看，ESB提供了[事件驱动](http://baike.so.com/doc/357634.html)和文档导向的处理模式，以及分布式的运行管理机制，它支持基于内容的路由和过滤，具备了复杂数据的传输能力，并可以提供一系列的标准接口。

#### ESB的特性

根据维基百科的定义，企业服务总线具有如下特性：

1. 它是面向服务架构的实现。
2. 它通常是操作系统和编程语言无关的；它应能在Java和.Net应用程序之间工作。
3. 它使用XML（可扩展标识语言）作为标准通信语言。
4. 它支持Web服务标准。
5. 它支持消息传递（同步、异步、点对点、发布-订阅）。
6. 它包含基于标准的适配器（如J2C/JCA），用于集成传统系统。
7. 它包含对服务编制（orchestration）和编排（choreography）的支持。
8. 它包含智能、基于内容的路由服务（itinerary路由）。
9. 它包含标准安全模型，用于ESB的认证、授权和审计。
10. 它包含转换服务（通常是使用XSLT），在发送应用和接收应用之间转换格式，简化数据格式和值的转换。
11. 它包含基于模式（schema）的验证，用于发送和接收消息。
12. 它可以统一应用业务规则，充实其它来源的消息，分拆和组合多个消息，以及处理异常。
13. 它可以条件路由，或基于非集中策略的消息转换，即不需要集中规则引擎。
14. 它可监视不同SLA（服务级别合约）的消息响应门限，以及在SLA中定义的其它特性。
15. 它（常常）简化“服务类别”，向更高或更低优先级用户做出适当的响应。
16. 它支持队列，在应用临时不可用时用来保存消息。
17. 它由（地理）分布式环境中的选择性部署应用适配器组成

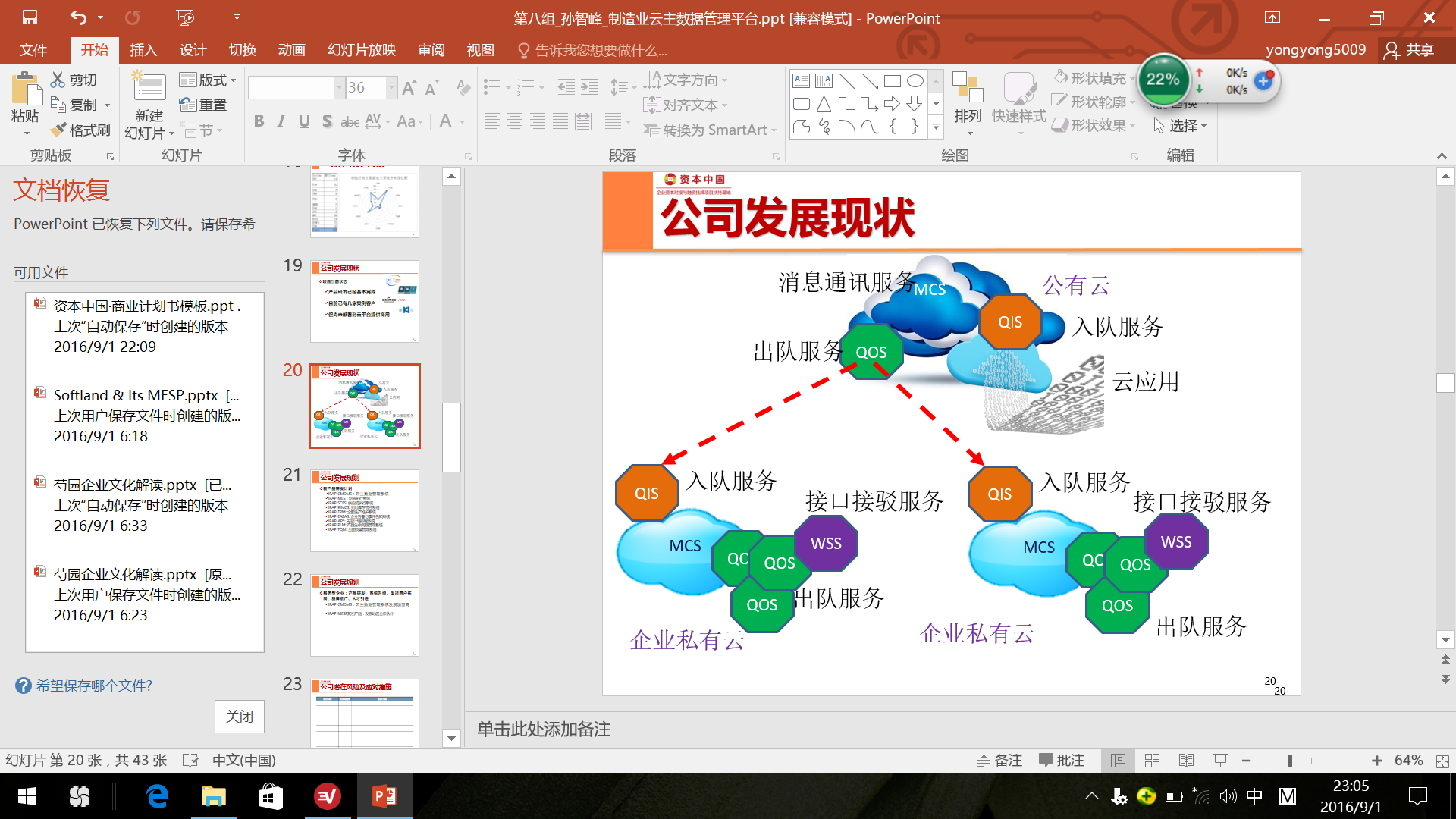
#### ESB的组件构成

ESB相当于一条信息高速公路，公路上划分若干车道，称之为队列。



ESB作为信息高速公路

要实现应用系统之间的集成，还需要入队服务和出队服务组件，甚至有Web Service接驳服务。



ESB的组件构成

#### ESB的消息队列结构

<Queue>

<head>

<Name>*User-Defined Queue Name*</Name>

<CorrelationID>*User-Defined Message ID*</CorrelationID>

<Persistence>*True or False*</Persistence>

<Priority>*User-Defined Priority(1-10)*</Priority>

<Type>*xml or text or zip*</Type>

</head>

<body>

<Message>*User-Defined Message Content*</Message>

</body>

<Properties>

<ExCode>*User-Defined ExCode*</ExCode>

<Filter>*User-Defined Filter, Usually for Designation of QOS*</Filter>

<ESBServerAddr>*User-Defined ESB Server Address*</ESBServerAddr>

*… User-Defined More Properties*

</Properties>

</Queue>

ESB上传输的消息报文也是三段结构：

<Softland>

<Head>

<ExCode>*User-Defined ExCode*</ExCode>

<CorrelationID>*User-Defined Message ID*</CorrelationID>

<CommunityID>*User-Defined CommunityID*</CommunityID>

<SysLogID>*User-Defined SysLogID*</SysLogID>

<UserCode>*UserCode*</UserCode>

<AgencyLeaf>*AgencyLeaf*</AgencyLeaf>

<RoleLeaf>*RoleLeaf*</RoleLeaf>

<StationID>*StationID*</StationID>

<UnixTime>*UnixTime*</UnixTime>

</Head>

<Body>

*User-Defined Message Body*

</Body>

<Log>

*User-Defined Log*

</Log>

</Softland>

# 设计原则

## 数据类型定义

表1给出了本规范中使用的数据类型定义。

1. 数据类型定义

|  |  |
| --- | --- |
| 标识符 | 数据类型 |
| N | 数字 |
| tinyint | 字节型，可以表示0─255 |
| smallint | 短整型，可以表示-32768─32767 |
| int | 整型，可以表示-2147483648─2147483647 |
| bigint | 长整型，可以表示-9223372036854775808  ─9223372036854775807 |
| decimal | 精确数字型 |
| float | 单精度浮点型 |
| double | 双精度浮点型 |
| AN | 包含0－9的数字和不包括汉字在内的其他字符 |
| char | 定长字符型 |
| varchar | 变长字符型 |
| text | 文本型 |
| ANC | 包含汉字在内的所有字符 |
| nchar | 定长字符型 |
| nvarchar | 变长字符型 |
| ntext | 文本型 |

## 标识项定义

本规范涉及如下标识项：

### 工序代码 Operation Code

在一个制造企业内唯一标识一个工序(Operation of Process)的代码，在IRAP-MDM中，工序代码用8位字符表示：

工段代码好

工序附加码

工序序号

工段代码：4位

工序序号：2位，支持99个

工序附加码：1位字母数字，用以区分设备型号以及人力配备带来的产能与工艺差异性，产能以及工艺标准、检查标准、测试标准、工装标准、装料标准、卸料标准、包装标准、生产程序、作业标准、失效模式、防错规则、工艺文档如果没有任何差异，附加码用“0”。

### 在制品站位代码 WIP Station Code

在一个制造企业内唯一标识一个在制品站位的代码，在IRAP-MDM中，制品站位代码用10位字符表示：

产线

工位

工序类型

车间或生产区域

厂区

厂区编号：从1-9,A-Z,共支持35个厂区

车间或生产区域编号：2位，支持99个

产线编号：3位，支持999个

在制品站位编号：3位，支持999个

工序类型：M─有料生产工序 W─无料加工工序 I─人工检查工序

T─机器测试工序 P─成品包装工序 H─物料审查工序

A─维修分析工序 R─故障维修工序 C─修复检查工序

Q─有料生产+人工检查工序 X─无料生产+人工检查工序

Y─有料生产+机器测试工序 Z─成品包装+附件配套工序

V─虚拟复合工序

### 在制品标识代码 WIP ID Code

在一个制造企业内唯一标识一个在制品个体或批次的代码，在IRAP-MES中，在制品代码用10-12位字符表示：

大类编号

小类编号

年份

年内序号

个体/群体标识

在IRAP-MES应用实践中通过自动识别技术识别的条码、RFID等可以是在制品的标识代码，也可以是产品的序列号、批号以及解析无歧异的容器编号、看板编号。

## 接口概述

本接口规范帮助实现第三方应用系统与IRAP-MESP之间的双向数据交换，接口逻辑示意图如下：

IRAP-MESP Databases

IRAP-MES 模块1

IRAP-MES 模块2

IRAP-MES 模块n

IRAP-MESP Web API

IRAP-MESP

生产设备系统

测试设备系统

信息追溯系统

# 接口说明

## 接口属性

### 调用地址

http://ServerIP:16912/OPCGateway/Exchange

## 系统信息接口 System Information

### ~~获取KepwareServer列表 GetKepServList~~

**接口标识：GetKepServList**

**接口描述：**本接口用于查看当前OPC系统中注册的KepwareServer列表。

**输入格式：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **GetKepServList** | 注意大小写敏感 |
| CommunityID | int | 社区标识 | 根据URL或社区切换 |

**输出格式：**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **GetKepServList** |  |
| ErrCode | int | 错误代号 |  |
| ErrText | nvarchar(400) | 错误文本 | 当ErrCode!=0时需弹出错误提示窗口，提示ErrText的内容。 |
| ----以下为行集 | | | |
| Ordinal | Int | 序号 | 前台排序用 |
| KepServAddr | String | KepServer的IP地址 |  |
| KepServName | String | KepServer的服务器名称 |  |

**调用举例：**

输入参数：

<Parameters>

<Param

Excode="GetKepServList"

CommunityID="60010" />

</Parameters>

输出参数：

<Result>

<Param

ExCode="GetKepServList"

ErrCode="0"

ErrText="KepwareServer列表获取完成。" />

<ParamXML>

<Row

Ordinal="1"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5" />

<Row

Ordinal="2"

KepServAddr="192.168.57.53"

KepServName="Kepware.KEPServerEX.V5" />

</ParamXML>

</Result>

**调用说明：**

1. 如果认证通过，ErrCode返回0，否则ErrCode!=0, ErrText返回错误信息，预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 121001 | 数据库链接失败。 | 16 | 否 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |

### ~~更新KepwareServer UpdateKepServ~~

**接口标识：UpdateKepServ**

**接口描述：**本接口用于更新当前OPC系统中注册的KepwareServer信息。

**输入格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **UpdateKepServ** | 注意大小写敏感 |
| CommunityID | int | 社区标识 | 根据URL或社区切换 |
| UpdateType | Int | 更新模式 | 1-新增  3-删除 |
| KepServAddr | String | KepServer的IP地址 |  |
| KepServName | String | KepServer的服务器名称 |  |

**输出格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **UpdateKepServ** |  |
| ErrCode | int | 错误代号 |  |
| ErrText | nvarchar(400) | 错误文本 | 当ErrCode!=0时需弹出错误提示窗口，提示ErrText的内容。 |

**调用举例：**

输入参数：

<Parameters>

<Param

Excode="UpdateKepServ"

CommunityID="60010"

UpdateType="1"

KepServAddr="127.0.0.1"

KepServName="TestServerName"/>

</Parameters>

输出参数：

<Result>

<Param

ExCode="UpdateKepServ"

ErrCode="0"

ErrText="KepServer信息更新完成。" />

</Result>

**调用说明：**

1. 如果认证通过，ErrCode返回0，否则ErrCode!=0, ErrText返回错误信息，预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 121001 | 数据库链接失败。 | 16 | 否 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |

## 设备信息接口 Device Information

### 获取设备列表 GetDevices

**接口标识：GetDevices**

**接口描述：**本接口用于查看指定KepwareServer中注册的所有设备。

**输入格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **GetDevices** | 注意大小写敏感 |
| KepServAddr | String | Kepware Server 的IP地址 |  |
| KepServName | String | Kepware Server的名称 |  |

**输出格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | varchar(50) | **GetDevices** |  |
| ErrCode | int | 错误代号 |  |
| ErrText | nvarchar(400) | 错误文本 | 当ErrCode!=0时需弹出错误提示窗口，提示ErrText的内容。 |
| ----以下为行集 | | | |
| Ordinal | Int | 序号 | 前台排序用 |
| KepServAddr | String | KepServer的IP地址 |  |
| KepServName | String | KepServer的服务器名称 |  |
| KepServChannel | String | KepServer中定义的Channel |  |
| KepServDevice | String | KepServer中定义的Device |  |

**调用举例：**

输入参数：

<Parameters>

<Param

Excode="GetDevices"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5" />

</Parameters>

输出参数：

<Result>

<Param

ExCode="GetDevices"

ErrCode="0"

ErrText="设备列表获取完成。" />

<ParamXML>

<Row

Ordinal="1"

DeviceCode="1001001001"

DeviceName="生产设备1"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5"

KepServChannel="Channel1"

KepServDevice="Device1" />

</ParamXML>

</Result>

**调用说明：**

1. 如果认证通过，ErrCode返回0，否则ErrCode!=0, ErrText返回错误信息，预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |
| 900101 | KepServer[%1!(%2!)]未在系统中注册 | 1 | 否 |
| 900103 | KepServer[%1!(%2!)]连接失败 | 1 | 否 |

### 获取设备标签（Tag）列表 GetDeviceTags

**接口标识：GetDeviceTags**

**接口描述：**本接口用于查看指定设备的标签（Tag）清单，如果ErrCode=0，返回的清单有效，否则无效，ErrText给出错误信息。

**输入格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | String | **GetDeviceTags** | 注意大小写敏感 |
| KepServAddr | String | Kepware Server的IP地址 |  |
| KepServName | String | Kepware Server的名称 |  |
| KepServChannel | String | Kepware Server的Channel名称 |  |
| KepServDevice | String | Kepware Server的Device名称 |  |

**输出格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用指南 |
| ExCode | String | **GetDeviceTags** |  |
| ErrCode | Int | 错误代号 |  |
| ErrText | String | 错误文本 |  |
| KepServAddr | String | KepServer的IP地址 |  |
| KepServName | String | KepServer的服务器名称 |  |
| KepServChannel | String | KepServer定义的Channel名称 |  |
| KepServDevice | String | KepServer定义的Device名称 |  |
| ----以下为行集 | | | |
| Ordinal | Int | 序号 | 前台排序用 |
| TagName | String | 标签名称 |  |

**调用举例：**

输入参数：

<Parameters>

<Param

ExCode="GetDeviceTags"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5"

KepServChannel="Channel1"

KepServDevice="Device1" />

</Parameters>

输出参数：

<Result>

<Param

ExCode="GetDeviceTags"

ErrCode="0"

ErrText="设备标签清单获取完成。"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5"

KepServChannel="Channel1"

KepServDevice="Device1" />

<ParamXML>

<Row

Ordinal="1"

TagName="Channel1.Device1.COMPOS\_01" />

<Row

Ordinal="2"

TagName="Channel1.Device1.FREE\_1" />

</ParamXML>

</Result>

**调用说明：**

1．如果获取成功，ErrCode返回0，否则ErrCode!=0, ErrText返回错误信息，预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 121001 | 数据库链接失败。 | 16 | 否 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |
| 900101 | KepServer[%1!(%2!)]未在系统中注册 | 1 | 否 |
| 900103 | KepServer[%1!(%2!)]连接失败 | 1 | 否 |
| 903321 | DeviceName[%1!]不存在 | 1 | 否 |

### ~~更新设备标签 Update Device Tags~~

**接口标识：UpdateDeviceTags**

**接口描述：**本接口用于更新设备的标签清单，如果ErrCode=0，更新设备标签成功；否则失败，ErrText给出错误信息。

**输入格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用说明 |
| ExCode | String | **UpdateDeviceTags** |  |
| CommunityID | Int | 社区标识 |  |
| UpdateType | Int | 更新模式 | * 1. 新增   2-修改  3-删除 |
| DeviceCode | String | 设备代码 | 注意大小写敏感 |
| DeviceName | String | 设备名称 |  |
| KepServAddr | String | KepServer的IP地址 |  |
| KepServName | String | KepServer的服务器名称 |  |
| KepServChannel | String | KepServer定义的Channel名称 |  |
| KepServDevice | String | KepServer定义的Device名称 |  |
| ----以下为行集 | | | |
| Ordinal | Int | 序号 |  |
| TagName | String | KepServer定义的标签名称 |  |
| DataType | String | KepServer定义的标签数据类型 |  |
| Description | String | KepServer定义的标签描述 |  |

**输出格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用说明 |
| ExCode | String | **UpdateDeviceTags** |  |
| ErrCode | String | 错误代号 |  |
| ErrText | String | 错误文本 |  |

**调用举例：**

输入参数：

<Parameters>

<Param

ExCode="UpdateDeviceTags"

CommunityID="60010"

UpdateType="1"

DeviceCode="1001001001"

DeviceName="生产设备1"

KepServAddr="127.0.0.1"

KepServName="Kepware.KEPServerEX.V5"

KepServChannel="Channel1"

KepServDevice="Device1" />

<ParamXML>

<Row

Ordinal="1"

TagName="COMPOS\_01"

DataType="Boolean"

Description="" />

<Row

Ordinal="2"

TagName="COMPOS\_02"

DataType="Boolean"

Description="" />

</ParamXML>

</Parameters>

输出参数：

<Result>

<Param

ExCode="UpdateDeviceTags"

ErrCode="0"

ErrText="设备标签更新成功。" />

</Result>

**调用说明：**

* 1. 如果ErrCode=0，返回的清单有效，否则无效，ErrText给出错误信息, 预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 121001 | 数据库链接失败。 | 16 | 否 |
| 101439 | 无效的设备代码 | 1 | 否 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |
| 900001 | 请求报文解析失败，原因：[%1!] | 1 | 否 |
| 900201 | 编号为[%1!]的设备未在系统中注册 | 1 | 否 |
| 900202 | 设备号[%1!]已经存在，对应的设备名称[%2!]，不能新增 | 1 | 否 |

### 设备指定标签数据读取与回写 DeviceTagValueRW

**接口标识：DeviceTagValueRW**

**接口描述：**本接口用于读取和回写设备Tag值内容，如果ErrCode=0，回写和读取成功；否则失败，ErrText给出错误信息。

**输入格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用说明 |
| ExCode | String | **DeviceTagValueRW** |  |
| CommunityID | Int | 社区标识 |  |
| KepServName | String | KepServer的服务器名称 |  |
| ----以下是Write行为的行集 | | | |
| Ordinal | Int | 序号 |  |
| TagName | String | KepServer定义的标签名称 |  |
| TagValue | String | 需要写入设备标签的值 |  |
| ----以下是Read行为的行集 | | | |
| Ordinal | Int | 序号 |  |
| TagName | String | KepServer定义的标签名称 |  |
| ----以下是Flag行为的属性 | | | |
| TimeOut | Int | 等待满足Flag行集中定义的所有标签值的时间，单位：秒 |  |
| ----以下是Flag行为的行集 | | | |
| Ordinal | Int | 序号 |  |
| TagName | String | KepServer定义的标签名称 |  |
| TagValue | String | 标签值 | 需要满足标签值的时候，获取Read行为行集中定义的标签值 |

**输出格式**：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 参数类型 | 参数说明 | 调用说明 |
| ExCode | String | **DeviceTagValueRW** |  |
| ErrCode | String | 错误代号 |  |
| ErrText | String | 错误文本 |  |
| ----以下是Read行为的结果行集 | | | |
| Ordinal | Int | 序号 |  |
| TagName | String | KepServer定义的标签名称 |  |
| TagValue | String | 读取到的标签值 |  |

**调用举例：**

输入参数：

<Parameters>

<Param

ExCode="DeviceTagValueRW"

CommunityID="60002"

KepServName="Kepware.KEPServerEX.V5" />

<ParamXML>

<Write>

<Tag Ordinal="1"

TagName="Channel1.Device1.COMPOS\_01"

TagValue="True" />

<Tag Ordinal="2"

TagName="Channel1.Device1.PARAM\_P8\_25"

TagValue="200" />

</Write>

<Read>

<Tag Ordinal="1"

TagName="Channel1.Device1.PARAM\_P6\_25" />

</Read>

<Flag TimeOut="30">

<Tag Ordinal="1"

TagName="Channel1.Device1.PART\_NUMBER"

TagValue="1" />

<Tag Ordinal="2"

TagName="Channel1.Device1.IN\_BATCH\_NUMBER"

TagValue="2" />

</Flag>

</ParamXML>

</Parameters>

输出参数：

<Result>

<Param

ExCode="DeviceTagValueRW"

ErrCode="0"

ErrText="执行成功。" />

<ParamXML>

<Read>

<Tag Ordinal="1"

TagName="Channel1.Device1.PARAM\_P6\_25"

TagValue="120" />

</ParamXML>

</Result>

**调用说明：**

* 1. 如果ErrCode=0，回写并读取成功，否则失败，ErrText给出错误信息, 预定义的错误信息包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
| 903341 | 设备标签回写失败 | 1 | 否 |
| 903342 | 设备标签读取超时 | 1 | 否 |

# 附录

## 预定义的错误信息

|  |  |  |  |
| --- | --- | --- | --- |
| 错误号 | 错误信息 | 严重级别 | 是否报告消息总线 |
|  |  |  |  |
| 101439 | 无效的设备代码 | 16 | 是 |
| 101448 | 无效的产品编号 | 16 | 是 |
| 101449 | 无效的生产程序 | 16 | 是 |
| 101450 | 无效的控制通道 | 16 | 是 |
| 101451 | 无效的开关编号 | 16 | 是 |
| 101452 | 无效的工装物料代码 | 16 | 是 |
| 101453 | 无效的工装失效代码 | 16 | 是 |
| 101454 | 无效的设备故障代码 | 16 | 是 |
| 101455 | 生产换型事件数据采集失败 | 16 | 是 |
| 101455 | 无效的工序代码 | 16 | 是 |
| 101456 | 工序报工事件数据采集失败 | 16 | 是 |
| 101456 | 无效的产品编号 | 16 | 是 |
| 101456 | 无效的生产工单号 | 16 | 是 |
| 101457 | 工序生产开始事件数据采集失败 | 16 | 是 |
| 101458 | 未获取到该设备的点检事项清单 | 1 | 否 |
| 101459 | 无效的社区标识 | 1 | 否 |
| 101461 | 无效的产线代码 | 16 | 是 |
| 101462 | 无效的工单编号 | 16 | 是 |
| 121001 | 数据库链接失败 | 16 | 否 |
| 121002 | 无效的用户代码 | 1 | 否 |
| 121003 | 用户密码校验失败 | 1 | 否 |
| 121004 | 验证码输入错误 | 1 | 否 |
| 121005 | 用户已经被注销 | 1 | 否 |
| 121006 | 未经允许的接入地址 | 1 | 否 |
| 121007 | 信息站点未注册 | 16 | 是 |
| 121008 | 用户已经登录系统，禁止重复登录 | 16 | 是 |
| 121009 | 用户已经退出系统 | 1 | 否 |
| 121011 | 无效的在制品标识 | 1 | 否 |
| 121012 | 工艺路由错误（%1!工位） | 1 | 否 |
| 121013 | 在制品标识有歧义 | 1 | 否 |
| 121014 | 测试数据采集失败 | 16 | 是 |
| 121015 | 尚未获取到在制品扫描信息 | 1 | 否 |
| 121016 | 无效的部件位置代号 | 1 | 否 |
| 121017 | 未找到原材料加料记录 | 1 | 否 |
| 121018 | 未找到原材料收料记录 | 1 | 否 |
| 121019 | 未找到原材料特性采集记录 | 1 | 否 |
| 121020 | 测量数据采集失败 | 16 | 否 |
| 121021 | 无效的工位代码（%1!） | 16 | 否 |
| 121022 | 无效的产品编号（%1！） | 16 | 否 |
| 121023 | 未找到质量参数 | 16 | 否 |
| 121025 | 测量数据不符合质量标准 | 0 | 否 |
| 121026 | 无效的设备代码（%1！） | 16 | 否 |
| 121027 | 设备（%1!）不在生产中 | 16 | 否 |
| 121028 | 未找到工单%1!在%2!的生产开始记录 | 16 | 否 |
| 121032 | 无效的产品编号 | 16 | 否 |
| 121033 | 无效的产线代码 | 16 | 否 |
|  | 检查数据采集失败 | 16 | 否 |
|  | 工艺参数采集失败 | 16 | 是 |
| 900000 | 报文体中的交易代码和报文头中的交易代码不一致 | 1 | 否 |
| 900001 | 请求报文解析失败，原因：[%1!] | 1 | 否 |
| 900101 | KepServer[%1!(%2!)]未在系统中注册 | 1 | 否 |
| 900102 | KepServer[%1!(%2!)]已在系统中注册 | 1 | 否 |
| 900103 | KepServer[%1!(%2!)]连接失败 | 1 | 否 |
| 903321 | DeviceName[%1!]不存在 | 1 | 否 |
| 900201 | 编号为[%1!]的设备未在系统中注册 | 1 | 否 |
| 900202 | 设备号[%1!]已经存在，对应的设备名称[%2!]，不能新增 | 1 | 否 |
| 903341 | 设备标签回写失败 | 1 | 否 |
| 903342 | 设备标签读取超时 | 1 | 否 |