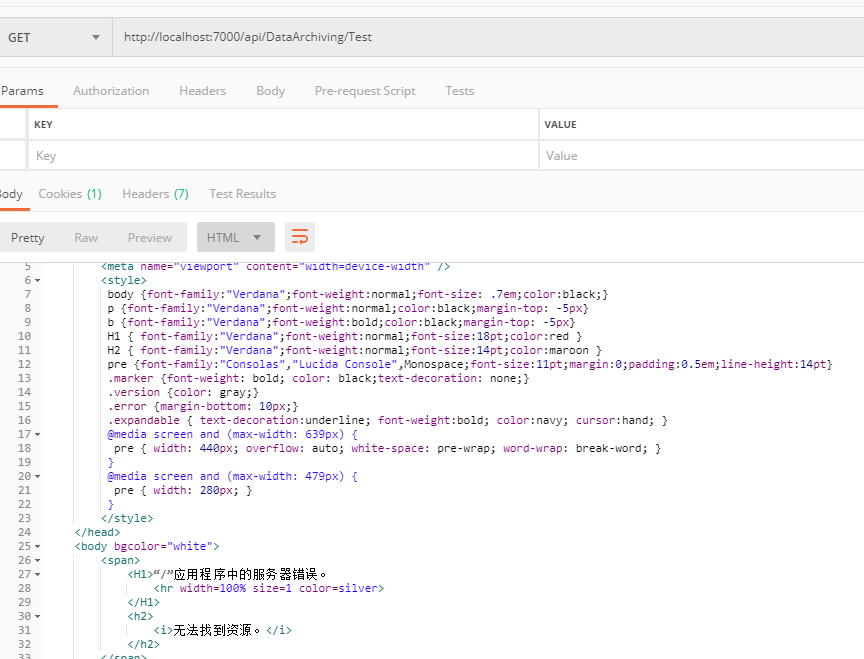
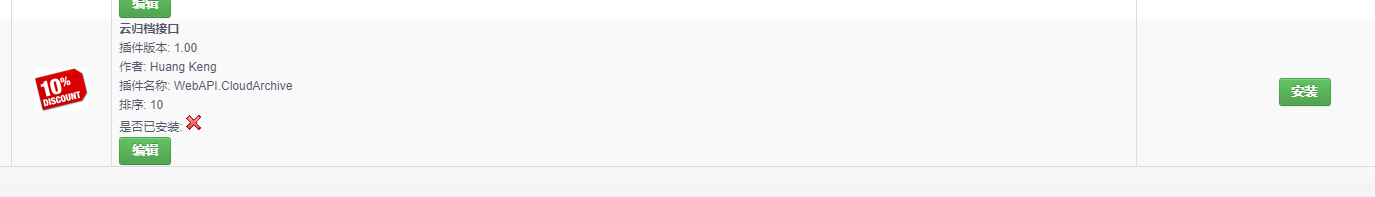
能力开放平台——正式动工

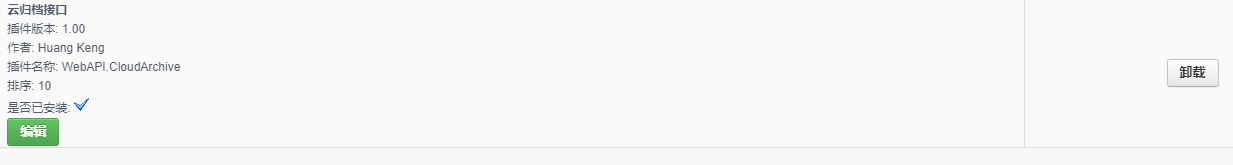
1.

接口无法找到资源，

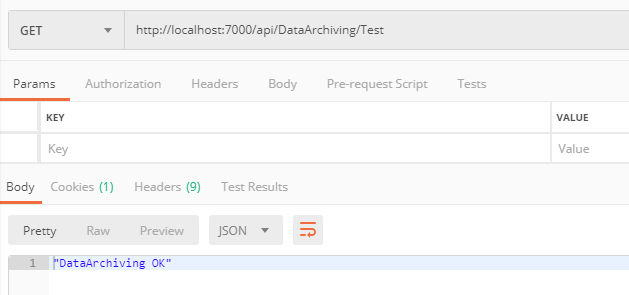


安装插件：





接口通了，开森。



2.

安装代码检查器。

除了eWorld.Entity之外，都给加上。

3.

添加第一个方法：

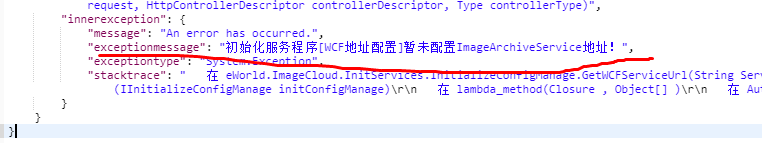
"error:System.NullReferenceException: 未将对象引用设置到对象的实例。\r\n 在 eWorld.Plugin.WebAPI.CloudArchive.Controllers.PublicController.Hua()"

应该是云归档的wcf没有配置地址。感觉。

4.

调用crm的wcfClient没有问题，ImageArchive就有问题了。

查了错误，是没有配置。



直接去源代码里面找：

/// <summary>

/// 获取WCF地址，如CRMService

/// </summary>

/// <param name="ServiceName">服务名称</param>

/// <param name="ReturnCompleteWCFUrl">是否返回完整的WCF地址，false仅返回IP</param>

/// <returns></returns>

public string GetWCFServiceUrl(string ServiceName, bool ReturnCompleteWCFUrl = true)

{

if (\_cacheWCFConnectSetting == null || \_cacheWCFConnectSetting.Count <= 0)

\_cacheWCFConnectSetting = InvokeApiGetWCFConnectString();

var currentWCFItem = \_cacheWCFConnectSetting.FirstOrDefault(s => s.Name == ServiceName);

if (currentWCFItem == null || string.IsNullOrEmpty(currentWCFItem.Url))

没有配置，是这个报错

throw new Exception(string.Format("初始化服务程序[WCF地址配置]暂未配置{0}地址！", ServiceName));

Uri hostUri = new Uri(currentWCFItem.Url);

if (!ReturnCompleteWCFUrl)

return hostUri.ToString();

else

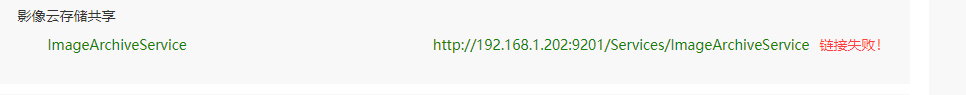
{

Uri retUri = new Uri(hostUri, string.Format("/Services/{0}.svc", ServiceName));

return retUri.ToString();

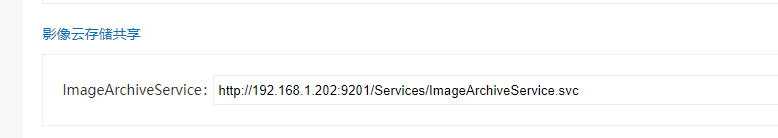
}

}



应该是这个的缘故，我改下配置。

加了svc



去202获取这个记录数据的xml文件，对的啊，有值的。

http://192.168.1.202:9201/Services/ImageArchiveService.svc

//初始化配置工具、WCF客户端调用

builder.RegisterType<InitializeConfigManage>().As<IInitializeConfigManage>().WithParameters(new List<NamedParameter>()

{

new NamedParameter("systemCode",1100),

new NamedParameter("initServiceSystemKey","bac11f3bd919f9fb")

}).SingleInstance();

知道原因了：

public enum SystemCodeEnum

{

门户 = 10,

公共服务 = 20,

远程医疗 = 100,

网络医院 = 200,

影像存储 = 300,

医学教育 = 400,

典型病例 = 500,

影像质控 = 600,

科研协同 = 700,

智能诊断 = 800,

影像大数据 = 900,

客户关系管理 = 1100,

运营管理 = 1200,

机构管理 = 1210,

在线监控 = 1300,

数据挖掘 = 1400,

能力开放 = 9100

}

这里是根据code的编码获取的，我这里给的是1100，所以我要换成300才行。

用军哥的1100获取的结果：

{

"data": [

{

"Name": "CommonService",

"Url": "http://192.168.1.202:9002/Services/CommonService.svc"

},

{

"Name": "MediaService",

"Url": "http://192.168.1.202:9002/Services/MediaService.svc"

},

{

"Name": "CRMService",

"Url": "http://192.168.1.202:9101/Services/CRMService.svc"

},

{

"Name": "PaymentService",

"Url": "http://192.168.1.202:9101/Services/PaymentService.svc"

}

],

"code": 0,

"msg": "操作成功"

}

用云归档的300获取的结果：

{

"data": [

{

"Name": "CommonService",

"Url": "http://192.168.1.202:9002/Services/CommonService.svc"

},

{

"Name": "MediaService",

"Url": "http://192.168.1.202:9002/Services/MediaService.svc"

},

{

"Name": "CRMService",

"Url": "http://192.168.1.202:9101/Services/CRMService.svc"

},

{

"Name": "ImageArchiveService",

"Url": "http://192.168.1.202:9201/Services/ImageArchiveService.svc"

}

],

"code": 0,

"msg": "操作成功"

}

开始报新的错：



两个方法重名了：

"exceptionmessage": "An exception was thrown while invoking the constructor 'Void .ctor(eWorld.ImageCloud.InitServices.IInitializeConfigManage)' on type 'eWorldWCFClient`1'. ---> 同一个协定中不能存在两个名称相同的操作，类型为 eWorld.IDCAS.WCFServerContract.ServerContract.IImageArchiveService 的方法 GetPatientExamDocumentList 和 GetPatientExamDocumentList 违反了此规则。可以通过更改方法名称或使用 OperationContractAttribute 的 Name 属性更改其中一个操作的名称。 (See inner exception for details.)",

暂时在方法后面加一个WithShort

/// <summary>

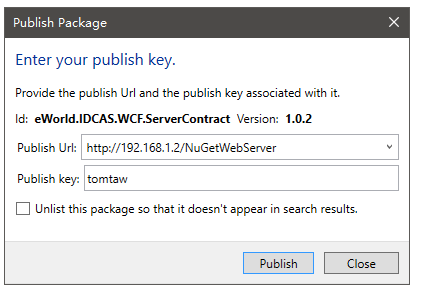
/// 获取文档信息(使用短链)

/// </summary>

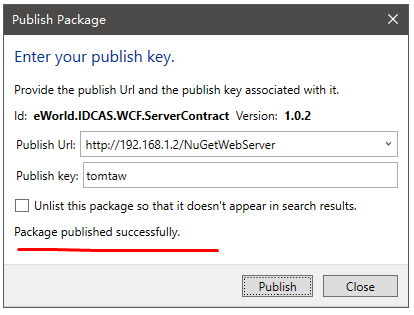
/// <param name="query">请求参数</param>

/// <returns>检查文档列表信息</returns>

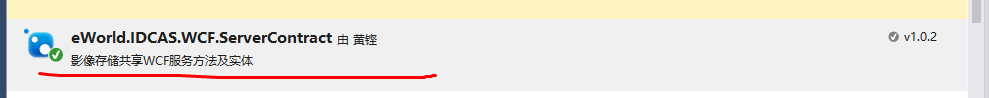
BaseListReturnResult<DocumentViewDTO> GetPatientExamDocumentListWithShort (DocumentQuery query);



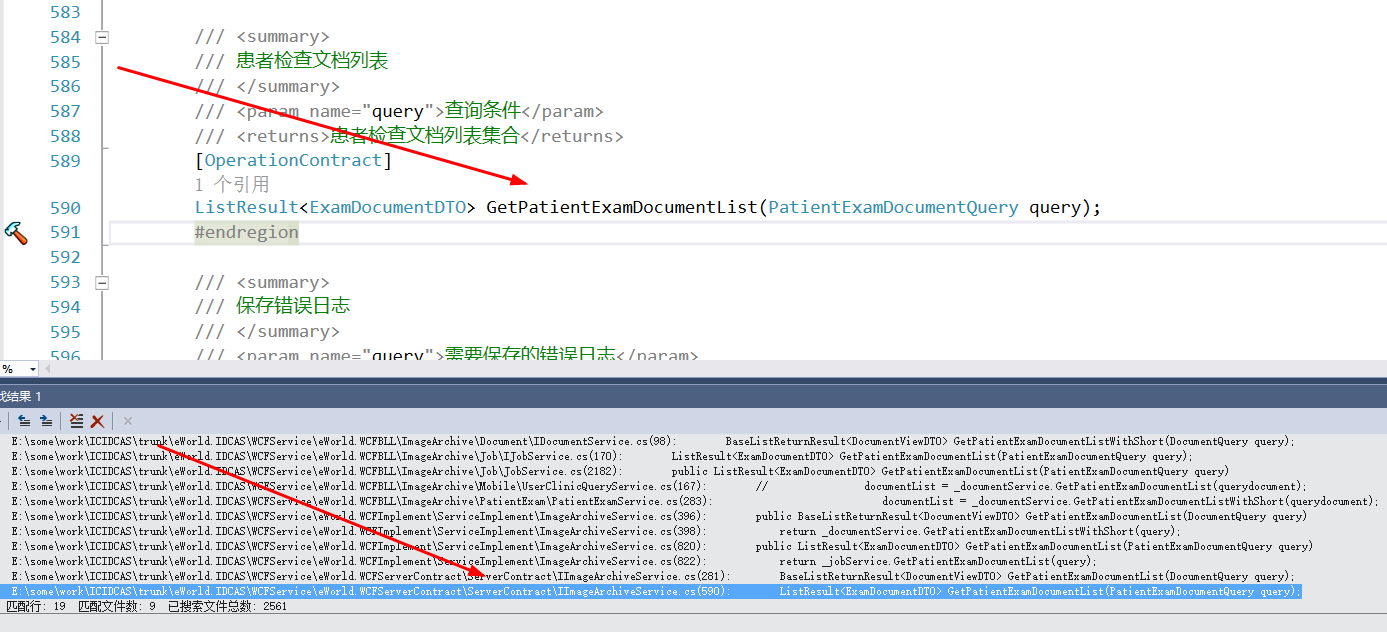
应该是成功了：



好了

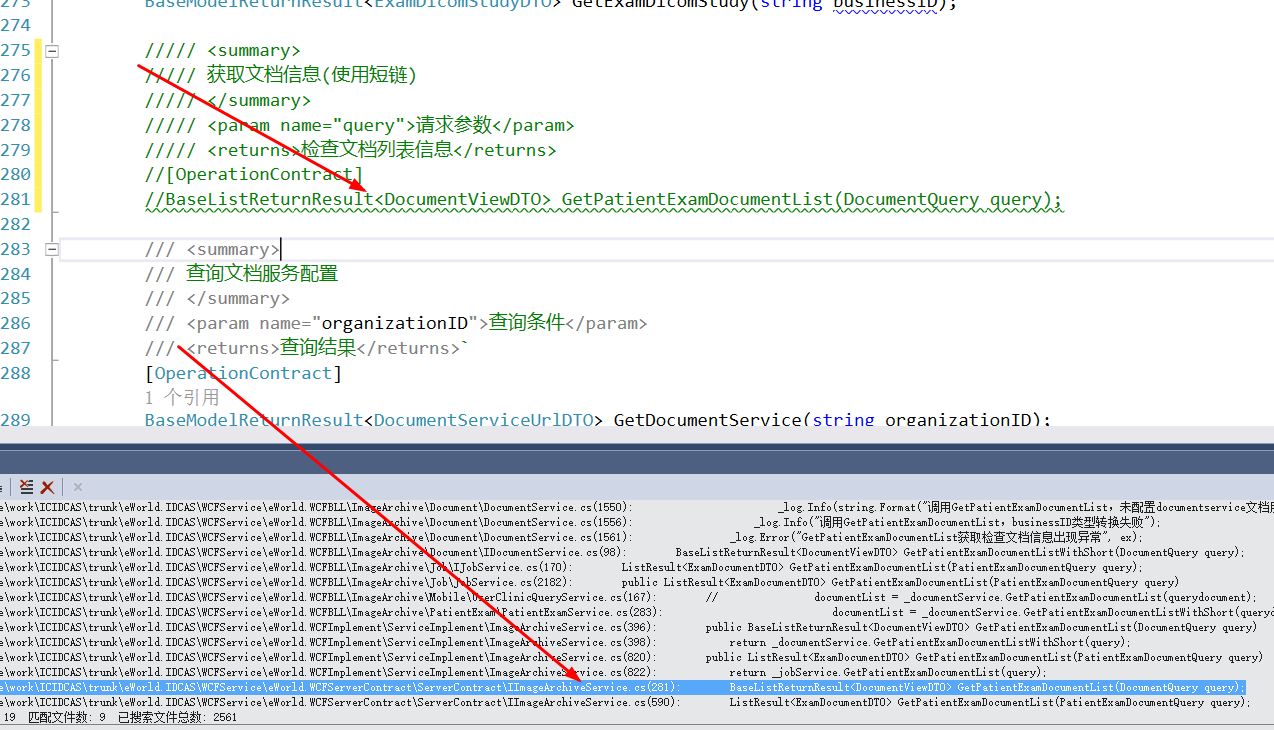


还有重名的，在ImageArchiveService中。先注释掉了。

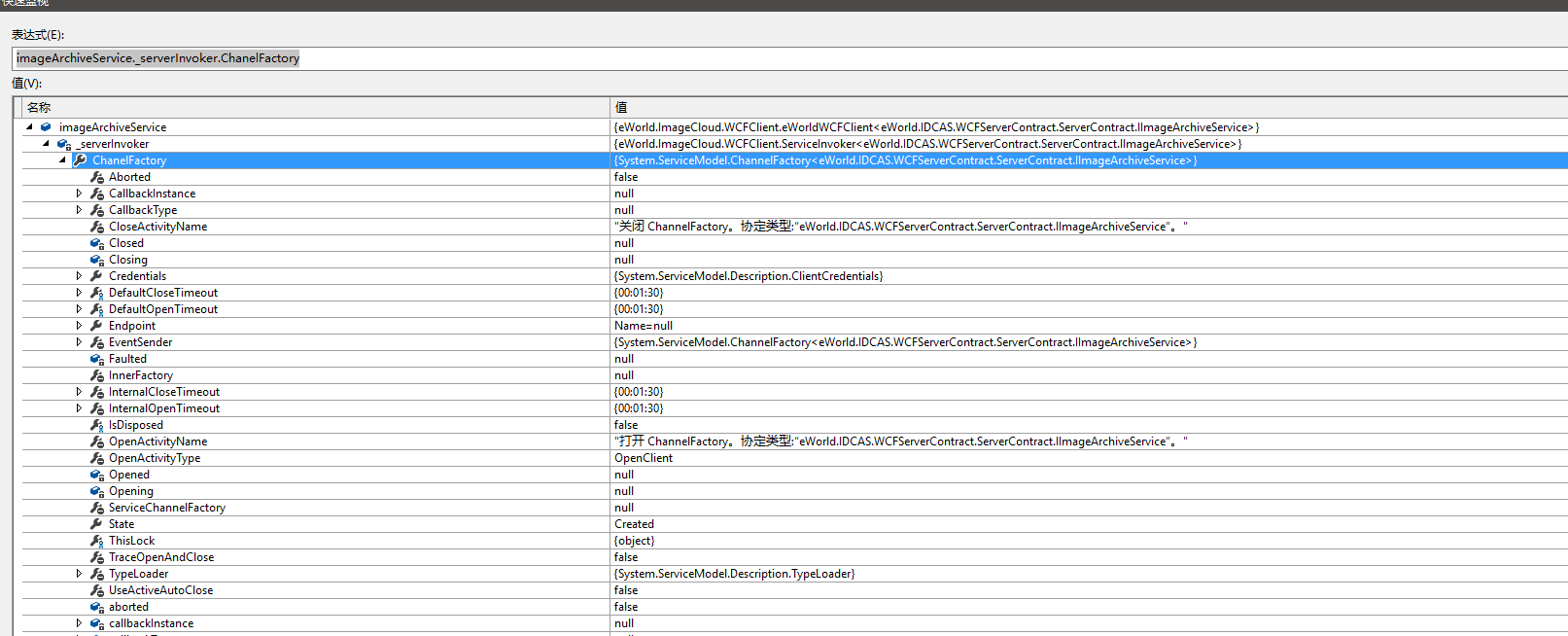


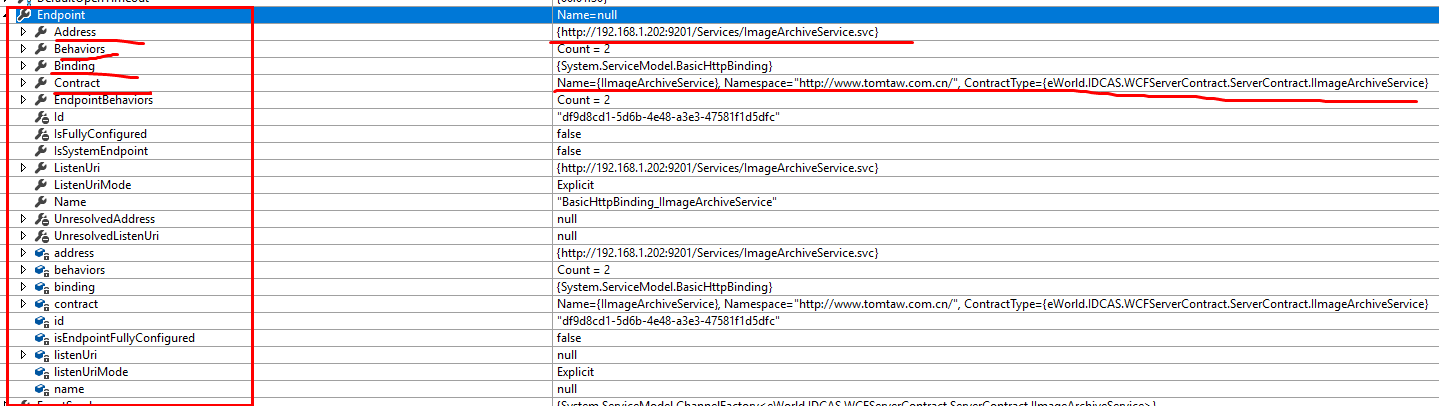
一共两个，一个是调用之前的DocumentService中的GetPatientExamDocumentList(DocumentQuery query)方法，（获取文档信息(使用短链)）

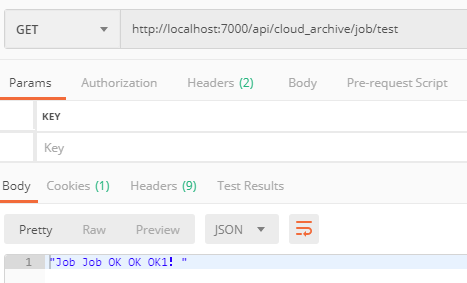
还有一个是调用JobService中的GetPatientExamDocumentList(PatientExamDocumentQuery query)方法。



来了来了，开森







Good Job！

调用的时候发生报错：



修改一下地址：

<http://localhost:9301/Services/ImageArchiveService.svc>

<http://192.168.1.202:9201/Services/ImageArchiveService.svc>

地址变成了本地，但是报错啊：

None of the constructors found with

'Autofac.Core.Activators.Reflection.DefaultConstructorFinder'

on type 'eWorld.IDCAS.WCF.BLL.Common.Configuration.DataDicInfoService'

can be invoked with the available services and parameters:Cannot resolve parameter 'eWorld.Core.Data.ICommonRepository`1[eWorld.Entity.Domain.Common.Configuration.DataDicInfo] datadicinfoRepository'

of constructor

'Void .ctor(eWorld.Core.Data.ICommonRepository`1[eWorld.Entity.Domain.Common.Configuration.DataDicInfo]

使用可用的服务和参数可以调用在

'eWorld.IDCAS.WCF.BLL.Common.Configuration.DataDicInfoService'类型中使用'Autofac.Core.Activators.Reflection.DefaultConstructorFinder'找到的构造函数：

无法解析参数

'eWorld.Core.Data.ICommonRepository`1 [eWorld.Entity.Domain.Common.Configuration.DataDicInfo] datadicinfoRepository'的构造函数

'Void.ctor（eWorld.Core.Data.ICommonRepository .1

[eWorld.Entity.Domain.Common.Configuration.DataDicInfo]]

"None of the constructors found with 'Autofac.Core.Activators.Reflection.DefaultConstructorFinder' on type 'eWorld.IDCAS.WCF.BLL.Common.Configuration.DataDicInfoService' can be invoked with the available services and parameters:\r\nCannot resolve parameter 'eWorld.Core.Data.ICommonRepository`1[eWorld.Entity.Domain.Common.Configuration.DataDicInfo] datadicinfoRepository' of constructor 'Void .ctor(eWorld.Core.Data.ICommonRepository`1[eWorld.Entity.Domain.Common.Configuration.DataDicInfo])'."

修改了这个：



5.

升级到Cloud 1.4后的报错：

几个方法：

（1）

/// <summary>

/// 获取患者平台检查列表

/// </summary>

/// <param name="query">查询条件</param>

/// <returns>获取患者平台检查列表的结果</returns>

public PageResult<PatientRelatedExamListDTO> GePatientPlatformExamList(ArchiveRecordQuery query)

（2）

#region Job触发器（JobTriggerMst）

/// <summary>

/// 根据条件查询Job触发器列表信息

/// </summary>

/// <param name="query">查询条件</param>

/// <returns>查询结果</returns>

PageResult<JobTriggerInfo> GetJobTriggerList(JobTriggerQuery query);

（3）

契约当中有些没加标签

#region Job触发器（JobTriggerMst）

/// <summary>

/// 根据条件查询Job触发器列表信息

/// </summary>

/// <param name="query">查询条件</param>

/// <returns>查询结果</returns>

PageResult<JobTriggerInfo> GetJobTriggerList(JobTriggerQuery query);

/// <summary>

/// 查询Job触发器及Job定义列表信息

/// </summary>

/// <param name="serviceUID">采集服务唯一号</param>

/// <param name="jobUID">作业唯一号</param>

/// <param name="groupUID">分组唯一号</param>

/// <returns>查询结果</returns>

ListResult<JobTriggerWithJobDefineInfo> GetJobTriggerWithJobDefineList(Guid? serviceUID, Guid? jobUID, Guid? groupUID);

/// <summary>

/// 保存Job触发器信息

/// </summary>

/// <param name="jobTriggerInfo">需要保存的Job触发器信息</param>

/// <returns>保存结果</returns>

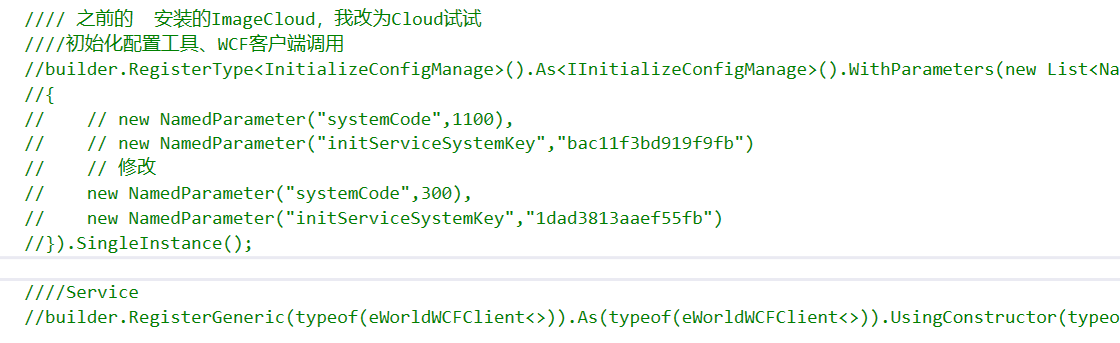
DataResult<Guid> SaveJobTrigger(JobTriggerInfo jobTriggerInfo);

"找不到方法:

“eWorld.Cloud.Result.ListResult`1<eWorld.IDCAS.WCF.ServerContract.Models.DTO.Info.DocumentViewDTO> eWorld.IDCAS.WCFServerContract.ServerContract.IImageArchiveService.GetExamDocumentList(eWorld.IDCAS.WCF.ServerContract.Models.DTO.Query.DocumentQuery)”。"

（4）

把注入，之前的ImageCloud改成Cloud试试。

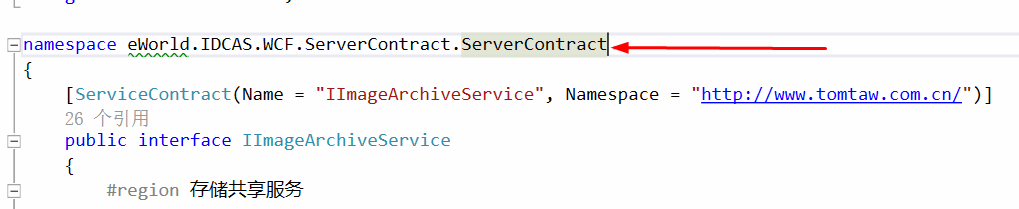


删除了，换成Cloud



修改了数据库的配置。

（5）



{"未能从程序集

“eWorld.IDCAS.WCF.ServerContract, Version=1.0.2.6694, Culture=neutral, PublicKeyToken=null”

中加载类型“eWorld.IDCAS.WCF.ServerContract.IImageArchiveService”。":"eWorld.IDCAS.WCF.ServerContract.IImageArchiveService"}

（6）

crm中的wcf



（7）

注入改回去了：

//// 之前的 安装的ImageCloud，我改为Cloud试试

////初始化配置工具、WCF客户端调用

//builder.RegisterType<InitializeConfigManage>().As<IInitializeConfigManage>().WithParameters(new List<NamedParameter>()

//{

// // new NamedParameter("systemCode",1100),

// // new NamedParameter("initServiceSystemKey","bac11f3bd919f9fb")

// // 修改

// new NamedParameter("systemCode",300),

// new NamedParameter("initServiceSystemKey","1dad3813aaef55fb")

//}).SingleInstance();

////Service

//builder.RegisterGeneric(typeof(eWorldWCFClient<>)).As(typeof(eWorldWCFClient<>)).UsingConstructor(typeof(IInitializeConfigManage)).InstancePerHttpRequest();

//要删除

//初始化配置工具、WCF客户端调用

builder.RegisterType<InitializeConfigManage>().As<IInitializeConfigManage>().WithParameters(new List<NamedParameter>()

{

//new NamedParameter("systemCode",1100),

//new NamedParameter("initServiceSystemKey","bac11f3bd919f9fb")

//修改

new NamedParameter("systemCode",300),

new NamedParameter("initServiceSystemKey","1dad3813aaef55fb")

}).SingleInstance();

//Service

builder.RegisterGeneric(typeof(eWorldWCFClient<>)).As(typeof(eWorldWCFClient<>)).UsingConstructor(typeof(IInitializeConfigManage)).InstancePerHttpRequest();

// builder.RegisterType<ImageArchiveService>().As<IImageArchiveService>()

//.WithParameter(ResolvedParameter.ForNamed<ICacheManager>("nop\_cache\_static"))

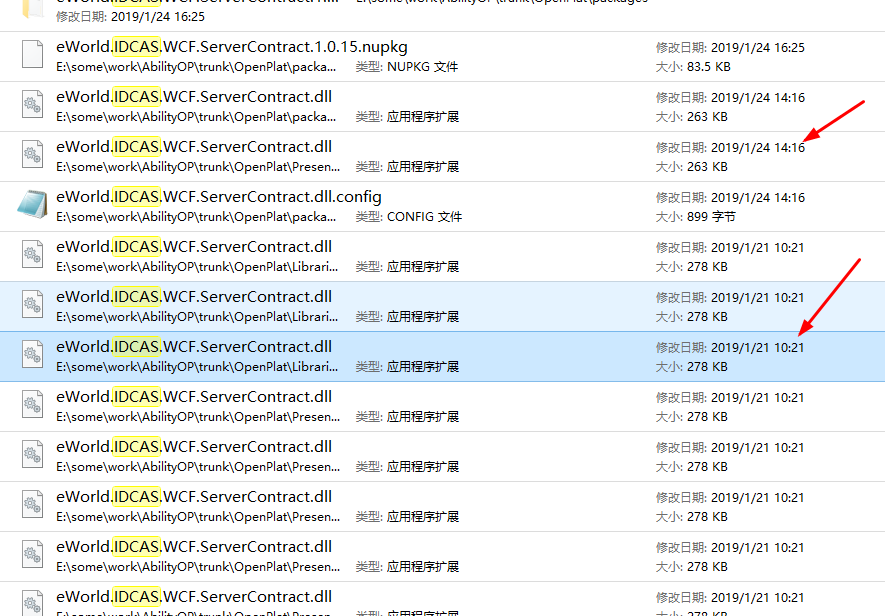
//.InstancePerHttpRequest();

////要删除

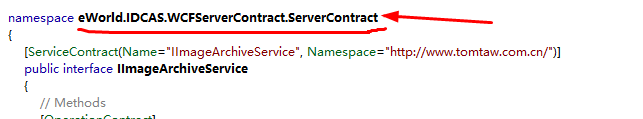
（8）

把重复的，之前的dll去除试试。

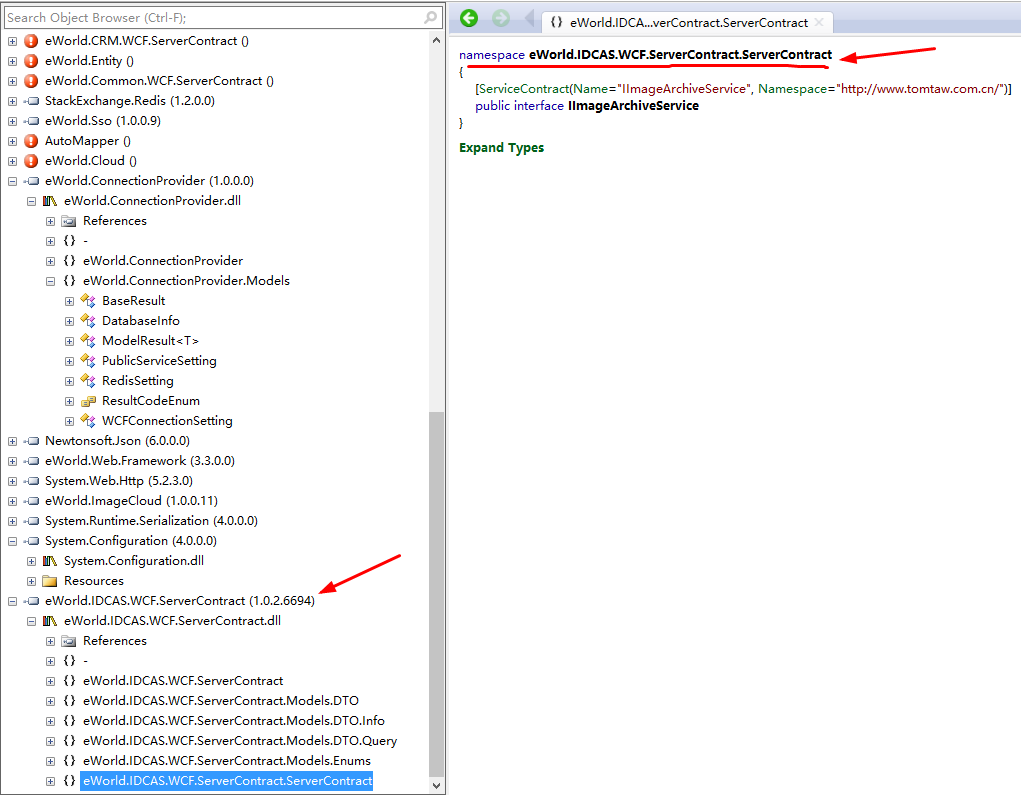
发现删除了，但编译之后，重新生成的还是之前版本的契约dll



反编译之后：



新的，多一个点的。



但为什么版本号1.02.6694是一样的？

（9）

搞定了，就是陈旧的dll错乱的问题，把eWorld.Core里面的引用去除后，再把所有的dll删除后，就好了。

控制器的注入依旧用了 ImageCloud

public ThirdPartyController(

ImageCloud.WCFClient.eWorldWCFClient<IImageArchiveService> imageArchiveService,

ImageCloud.InitServices.IInitializeConfigManage initializeConfigManage,

ImageCloud.WCFClient.eWorldWCFClient<ICRMService> crmClient)

{

this.\_imageArchiveService = imageArchiveService;

this.\_initializeConfigManage = initializeConfigManage;

this.\_crmClient = crmClient;

}

但暂时切换到本地的9301调用，因为可以调试，而且数据库也正常。

测试wawa

<http://localhost:7000/api/cloud_archive/ThirdParty/wa_wa>

get请求，函数体：

[HttpGet]

public object wa\_wa()

{

string wcf = null;

try

{

DocumentQuery param = new DocumentQuery()

{

BusinessID = "16b7e171-6e3d-45a7-90d0-925d6907275d"

};

var result = this.\_imageArchiveService.Invoke(i => i.GetExamDocumentList(param));

return result;

//return null;

//var customer = this.CurrentCustomerGuid;

//return "customer" + customer + "Public OK OK OK1！";

}

catch (Exception e)

{

return $@"wcf:{wcf}\r\nerror:{e}";

}

}

返回结果：

{

"data": [

{

"fileuid": "94440b6d-1a75-4bc3-b36b-9bb235313241",

"businessid": "16b7e171-6e3d-45a7-90d0-925d6907275d",

"mimetype": "image/jpeg",

"businesstype": "Exam",

"uniqueid": null,

"classcode": "Exam",

"typecode": "ExamResult",

"formatcode": null,

"filesha": "dd2e8ca104a4ce60f29ac9841465c0eaecaacc01",

"filesize": 1165368,

"filecreatetime": "2018-12-18T09:25:52",

"originalfilename": "b0f33923-10e4-4b3a-849f-19c1eb1c4212.jpg",

"filedownloadurl": "http://192.168.1.202:9200/DocumentService/Download/Document/?fileUid=94440b6d-1a75-4bc3-b36b-9bb235313241&token=d44b4ee73edc072798df8f3cfe65578f",

"dicomdownloadlist": null,

"fileviewurl": "http://192.168.1.202:9200/DocumentService/Viewer/?businessID=16b7e171-6e3d-45a7-90d0-925d6907275d&businessType=Exam&classCode=Exam&typeCode=ExamResult&fileUid=94440b6d-1a75-4bc3-b36b-9bb235313241&organizationID=47000326-5&print=&download=&businessView=false&token=635c910899c69b7ae23612780630f9bd&reportimgpost=&Anonymous=",

"filebusinessviewurl": "http://192.168.1.202:9200/DocumentService/Viewer/?businessID=16b7e171-6e3d-45a7-90d0-925d6907275d&businessType=Exam&classCode=Exam&typeCode=ExamResult&fileUid=94440b6d-1a75-4bc3-b36b-9bb235313241&organizationID=47000326-5&print=&download=&businessView=true&token=635c910899c69b7ae23612780630f9bd&reportimgpost=&Anonymous=",

"eworldviewerurl": null,

"filepath": null,

"filerelativepath": null,

"modality": null

},

{

"fileuid": "7f4d657d-3126-479f-b510-7f5aa88f33c5",

"businessid": "16b7e171-6e3d-45a7-90d0-925d6907275d",

"mimetype": "application/dicom",

"businesstype": "Exam",

"uniqueid": null,

"classcode": "Exam",

"typecode": "ExamImage",

"formatcode": "DICOMDIR",

"filesha": "089138ae459ec17085984c1c908867fc663a4fcb",

"filesize": 7902304,

"filecreatetime": "2018-12-18T09:25:53",

"originalfilename": "75ae6a62-cc8a-43c5-b2f8-944030112160.zip",

"filedownloadurl": "http://192.168.1.202:9200/DocumentService/Download/Document/?fileUid=7f4d657d-3126-479f-b510-7f5aa88f33c5&token=9e35d2f8a9d86016da669a957f3263d6",

"dicomdownloadlist": null,

"fileviewurl": "http://192.168.1.202:9200/DocumentService/Viewer/?businessID=16b7e171-6e3d-45a7-90d0-925d6907275d&businessType=Exam&classCode=Exam&typeCode=ExamImage&fileUid=7f4d657d-3126-479f-b510-7f5aa88f33c5&organizationID=47000326-5&print=&download=&businessView=false&token=9836f40e0224e9b06e4667b57016c58f&reportimgpost=&Anonymous=",

"filebusinessviewurl": "http://192.168.1.202:9200/DocumentService/Viewer/?businessID=16b7e171-6e3d-45a7-90d0-925d6907275d&businessType=Exam&classCode=Exam&typeCode=ExamImage&fileUid=7f4d657d-3126-479f-b510-7f5aa88f33c5&organizationID=47000326-5&print=&download=&businessView=true&token=9836f40e0224e9b06e4667b57016c58f&reportimgpost=&Anonymous=",

"eworldviewerurl": null,

"filepath": null,

"filerelativepath": null,

"modality": null

}

],

"code": 0,

"msg": "OK"

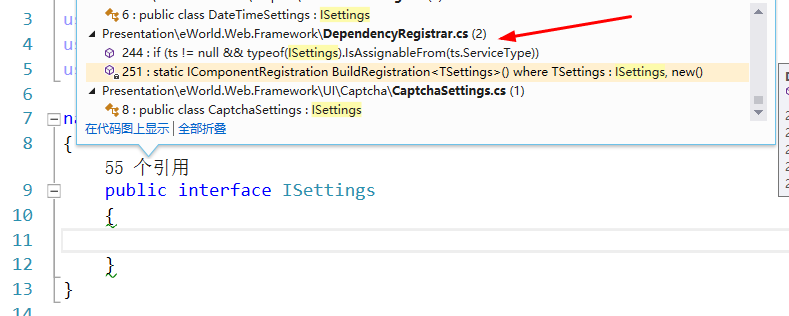
}

居然直接把地址拼接好了，不用我再做data的转换？

（10）

6.

配置表的批量注入：



不知道是不是就是这个：

public IEnumerable<IComponentRegistration> RegistrationsFor(

Service service,

Func<Service, IEnumerable<IComponentRegistration>> registrations)

{

var ts = service as TypedService;

if (ts != null && typeof(ISettings).IsAssignableFrom(ts.ServiceType))

{

var buildMethod = BuildMethod.MakeGenericMethod(ts.ServiceType);

yield return (IComponentRegistration)buildMethod.Invoke(null, null);

}

}

static IComponentRegistration BuildRegistration<TSettings>() where TSettings : ISettings, new()

{

return RegistrationBuilder

.ForDelegate((c, p) =>

{

var currentWebSiteId = c.Resolve<IWebSiteContext>().CurrentWebSite.Id;

//uncomment the code below if you want load settings per WebSite only when you have two WebSites installed.

//var currentWebSiteId = c.Resolve<IWebSiteService>().GetAllWebSites().Count > 1

// c.Resolve<IWebSiteContext>().CurrentWebSite.Id : 0;

//although it's better to connect to your database and execute the following SQL:

//DELETE FROM [Setting] WHERE [WebSiteId] > 0

return c.Resolve<ISettingService>().LoadSetting<TSettings>(currentWebSiteId);

})

.InstancePerHttpRequest()

.CreateRegistration();

}

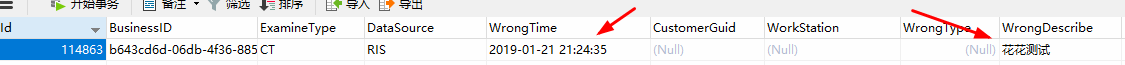
7.

所有的log暂时注释

// \_log.Debug("Endpoint地址为代理模式，将转换为：" + endpoint);

8.

测试SaveLogError方法成功。



9.

把参数验证移植到云归档里面，而不是放在能力开放平台。

if (triggerUID == Guid.Empty)

{

result.code = -1;

result.msg = "TriggerUID不能为空";

\_log.Info("查询任务触发的数据配置列表失败，错误信息：TriggerUID不能为空");

return result;

}

10.

{"DicomInfo":{"OrganizationID":null,"DICOMIP":"192.168.1.9","DICOMPort":109,"DICOMAE":"IA9","DICOMCMovePort":0},"StudyInstanceUIDList":["1.2.826.0.1.3680043.2.1505.97676.20160101093257.0"],"PatientID":"GY92260","AccessionNumber":"SD-53259","Anonymize":false}

1.2.826.0.1.3680043.2.1505.97676.20160101093257.0

{"RequestFileList":[],"ReportFileList":[{"DownloadUrl":"http://192.168.2.47:8787/DocumentService/Download/Document?FileUID=140fac5e-6ba2-4c2a-85dd-c57659c38148&token=2b5b070f839b35329a8b1f505a309b5c","TempFileName":"140fac5e-6ba2-4c2a-85dd-c57659c38148.Jpg","ImageFormatCode":null,"FilmFormatCode":null}],"ImageFileList":[],"FilmFileList":[],"BusinessTime":"2018/11/24 9:32:57"}

修改解决方案

Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "eWorld.Entity", "..\..\..\Data\trunk\entity\eWorld.Entity\eWorld.Entity.csproj", "{BD066F4B-E1AB-423E-BC3D-B32688CE6EF9}"

11.

预留，job控制器中的注册和获取检查列表，两个接口没做，入参实体变了，而且nuget上没发新包。

12.

原来的控制器的方法，现在舍去，保留下。

//public PublicController(

// eWorldWCFClient<IImageArchiveService> imageArchiveService,

// IInitializeConfigManage initializeConfigManage,

// eWorldWCFClient<ICRMService> crmClient)

//{

// this.\_imageArchiveService = imageArchiveService;

// this.\_initializeConfigManage = initializeConfigManage;

// this.\_crmClient = crmClient;

//}

13.

public：

1)获取dicom信息

2)获取影像浏览地址

3)获取检查列表(hide\_name没有处理，所以没做)

4)

5)

.ForMember(dest => dest.ICC\_Web\_Client\_Url, opt => opt.MapFrom(p => CommonUtil.TransLocalhostForClient(p.ICCWebClientUrl, \_idcasurl)))

Job控制器：

1）获取采集服务信息

2）保存采集服务

组织id 123308244720006231

3)

5）注册

a.结果里面的观察结果序列号，有必要传？最后没有传。

b.最后的数据处理，在注册成功之后，分枝版本。

I：保存检查校验信息

II：根据工号和机构ID查询医生信息，如果能查到用户，就调用接口发送消息

还要发送：APP报告状态修改消息

//调用消息推送

var url = ConfigDefine.MessagePushUrlByAPP;

这个我还没处理。

问题：

1.关于必选的参数验证，并没有做处理。

2.关于visit的处理

如果参数里面没有VisitUID，就new一个

if (!query.VisitUID.HasValue)

{

query.VisitUID = Guid.NewGuid();

这个处理是肯定有问题的，因为如果是更新，没有传，但ObservationRequest的数据存在，可以直接点出来当前的VisitUID的

没问题，这后面有做处理的，当存在数据，就不映射：

if (obs.VisitUID.HasValue)

{

mapExpObs = mapExpObs.ForMember(t => t.VisitUID, dto => dto.Ignore());

}

映射后又做这样的处理：

obs = Mapper.Map(query, obs);

if (!obs.VisitUID.HasValue)

{

obs.VisitUID = query.VisitUID;

}

if (query.VisitUID.HasValue && obs.VisitUID.HasValue && obs.VisitUID.Value != query.VisitUID.Value)

{

obs.VisitUID = query.VisitUID;

}

发现哪怕是重复数据，都是新建了一条ObservationRequest请求，那现在的逻辑是，只要不传ObservationUID，就是重新建一条数据，而不再覆盖？

而如果传了ObservationUID，就覆盖了。

{

"patient": {

"patient\_id": "0000040",

"pid\_assigning\_authority": "2402.PS",

"name": "花花",

"sex": "男",

"birth\_date": "2014-11-9",

"id\_card\_no": "330724xxxxx1020312",

"contact\_phone\_no": "13xxxxx5512",

"organization\_id": "2402"

},

"observation\_request": {

"patient\_id": "0000040",

"observation\_uid": "efbd0ffc-8472-444a-813d-31b74f6a7f58",

"pid\_assigning\_authority": "2402.PS",

"patient\_class": "门诊",

"organization\_id": "2402",

"organization\_name": "余杭区第一人民医院",

"accession\_number": "2415565",

"service\_id": "CGX334833",

"procedure\_name": "左踝关节正侧位",

"service\_sect\_id": "DR",

"reg\_time": "2019-01-02 09:36:33",

"observation\_date": "2019-01-02 09:36:33",

"audit\_date": "2019-01-02 09:36:33",

"result\_date": "2019-01-02 09:36:33",

"data\_source": "RIS",

"age": "28",

"age\_unit": "岁"

},

"observation\_result\_list": [

{

"set\_id": "1",

"observation\_id": "GDT",

"value\_text": "两肺纹理略增多，肺野内未见明显病灶；两肺门影无明显增大。肝脏表面光正，隔夜比例未见失调，干裂无增宽，干是指未见异常密度灶。肝门结构清晰，蛋刀系统未见扩张，胆囊不大，囊壁无 增厚，囊内未见异常密度早，一线形态密度未见异常，脾脏不大，其实质内未见异常密度影，两侧肾上腺、肾脏邢台密 度未见异常。"

},

{

"set\_id": "2",

"observation\_id": "IMP",

"value\_text": "222未见异常组织密度影。"

}

]

}

结果：

{

"data": "5fba1eb6-a750-49d9-a232-d8c898b0dd37",

"code": 0,

"msg": "OK"

}

{

"data": "efbd0ffc-8472-444a-813d-31b74f6a7f58",

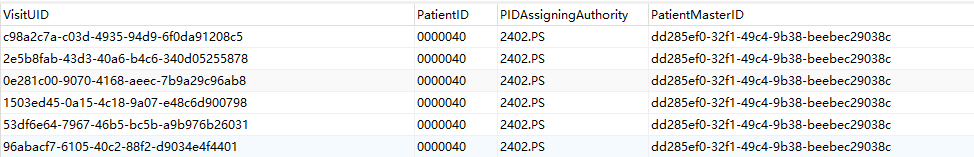
"code": 0,

"msg": "OK"

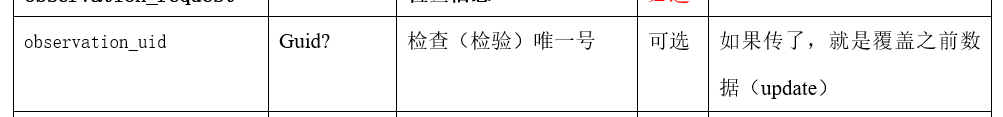
}

这里的问题是，当传入ObservationUID，覆盖的同时，Visit是不覆盖的，所以又多了一条Visit表的数据，而不是更新原来的记录。

无论更新还是insert，都新生成了一条visit数据。



修改了字段说明：



自我感觉，改成我开发的温岭那个版本比较好。

6)保存采集任务

7)获取采集任务

采集标记在这里有些应用

// 干部优先

if (!string.IsNullOrEmpty(query.InteractionIdExt1))

{

query.InteractionIdExt1 = query.InteractionIdExt1.Trim();

string sqlcodefirstcount = sqlcount + sqlOrganizationwhere + sqlwhere + " AND InteractionIdExt1='" + query.InteractionIdExt1 + "' ";

string sqlcodefirstdaycount = sqlcodefirstcount + " AND PerformTime>='" + nowdaystart + "' " + sqlcountorder;

\_log.Debug("GetIntegrationTaskList采集服务获取任务列表，查询是否有优先采集执行语句:" + sqlcodefirstdaycount);

resultcount = \_integrationTaskRepository.SqlQuery<long>(sqlcodefirstdaycount).ToList();

if (resultcount.Count > 0 && resultcount[0] > 0)

{

// 当天数据有优先采集任务

collecttype = 1101;

}

else

{

resultcount = \_integrationTaskRepository.SqlQuery<long>(sqlcodefirstcount).ToList();

if (resultcount.Count > 0 && resultcount[0] > 0)

{

// 有优先采集任务

collecttype = 1100;

}

else

{

// 没有优先采集数据

collecttype = 1000;

}

}

}

if (collecttype > 1099)

{

if (collecttype == 1101)

{

sqlordere = " ORDER BY PerformTime ";

sqlwhere += " AND InteractionIdExt1='" + query.InteractionIdExt1 + "' AND PerformTime>='" + nowdaystart + "' ";

}

else if (collecttype == 1100)

{

sqlordere = " ORDER BY PerformTime DESC ";

sqlwhere += " AND InteractionIdExt1='" + query.InteractionIdExt1 + "' ";

}

}

采集标识：

兵哥 这个CollectionMark 采集状态 这些1000 1144 1111是怎么定义的？

曹兵 2019/2/26 10:21:14

1000是未采集 初始化状态

1144 是检查变更需要重新采集检查

1111是采集完成的

1是变更需要采集 0是未采集 4是不存在

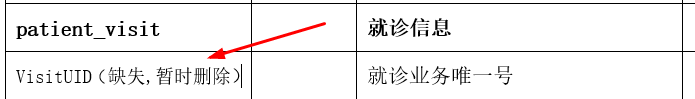
第一位是占位符 第二位是检查 第三位是影像 第四位是胶片

自己的扩展：1044，那就是没有影像和胶片，只有检查，还未采集？

8)获取版本

9)云到云检查列表

就诊表Visit信息中，VisitUID的缺失，不过在ObservationRequest中是有VisitUID的。



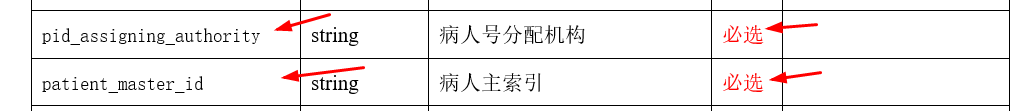
10) 云到云检查详情

DataArchiving：

1）患者信息注册

2）观察申请信息注册

这里的必选，和采集里面的检查注册是不一样的的，那边不是必选，因为可以从病人信息注册成功后获取，但这里，必须直接传进去。类似温岭的接口。



剩下的问题，改造这个接口！

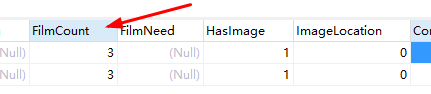
3）观察申请结果注册

4)

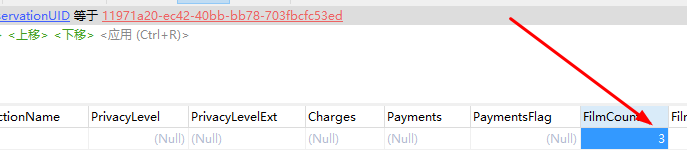
！！！将新版本的wcf发布到202，结果数据库配置没改，又连到77上面去了。之后要注意！！！

！！！还有，我这边的能力开放平台，最后的发布配置也要改成200的数据库！！！

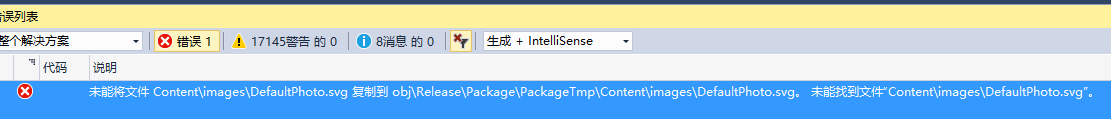
改到77上去了：



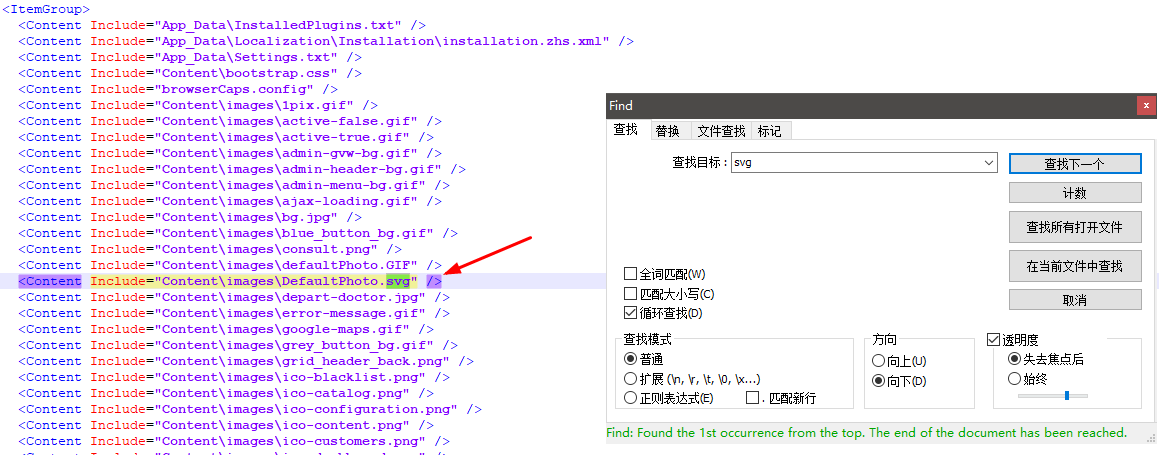
成功，修改了初始化配置中，200的数据库。



发布报错：

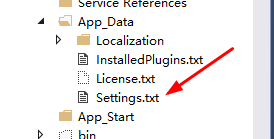


把程序集里面这个删了：

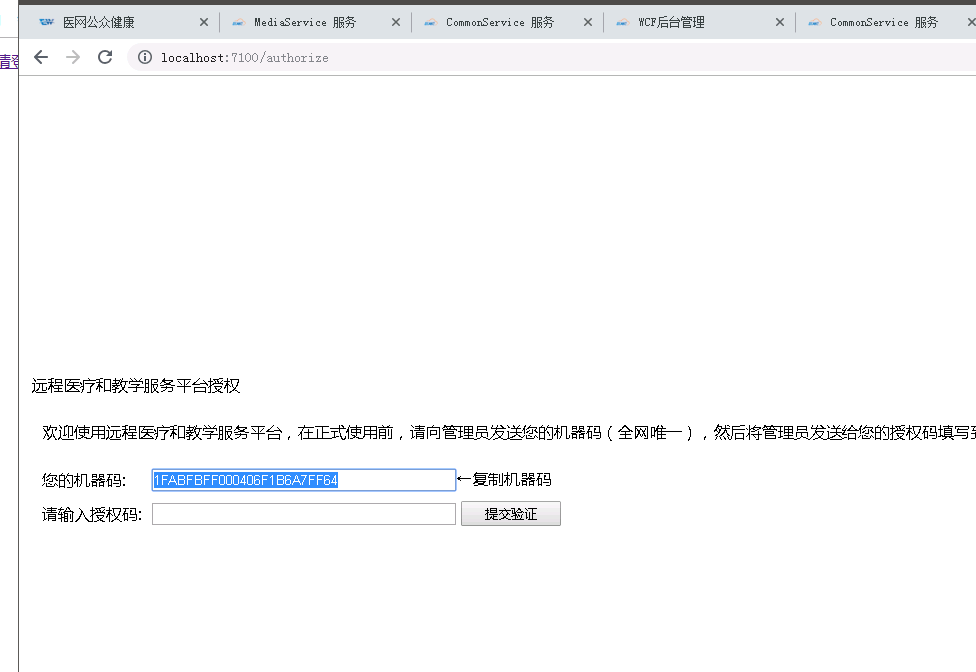


报错：

原因是appData这个没有放进去：

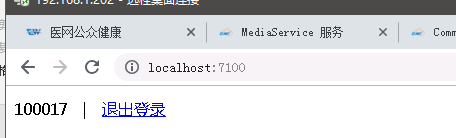


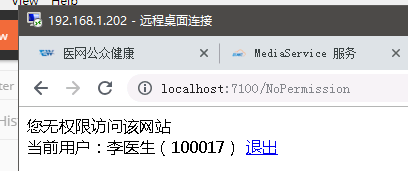
202的机器码：



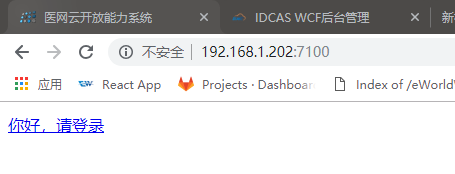


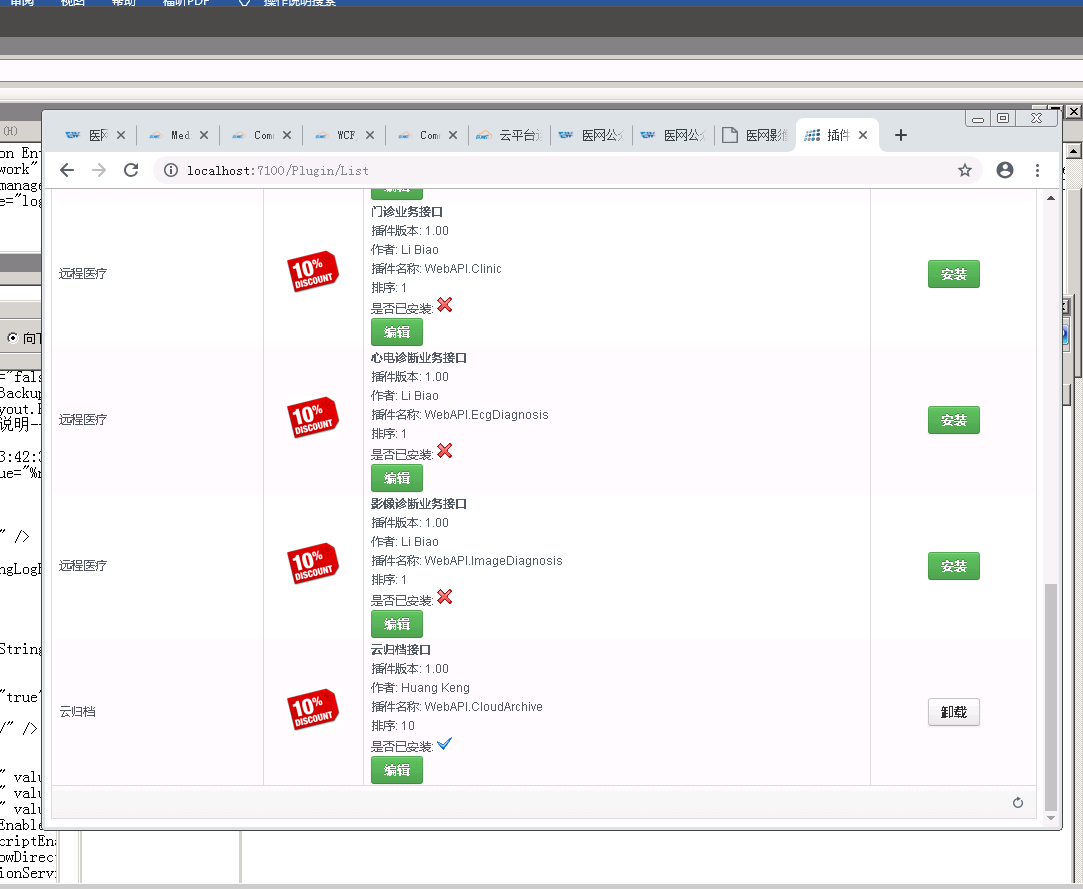
好了



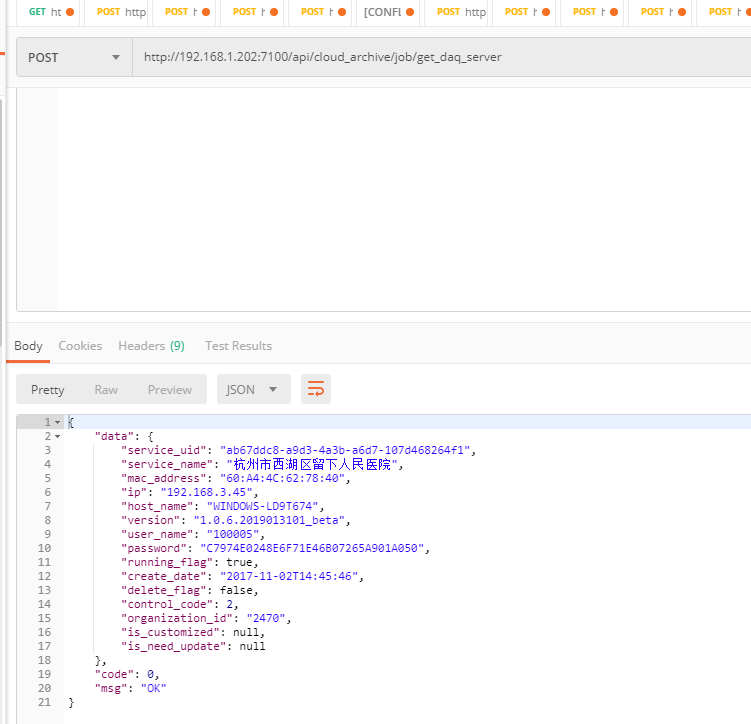


可以远程访问了：





！！！跑起来了！！！



14.

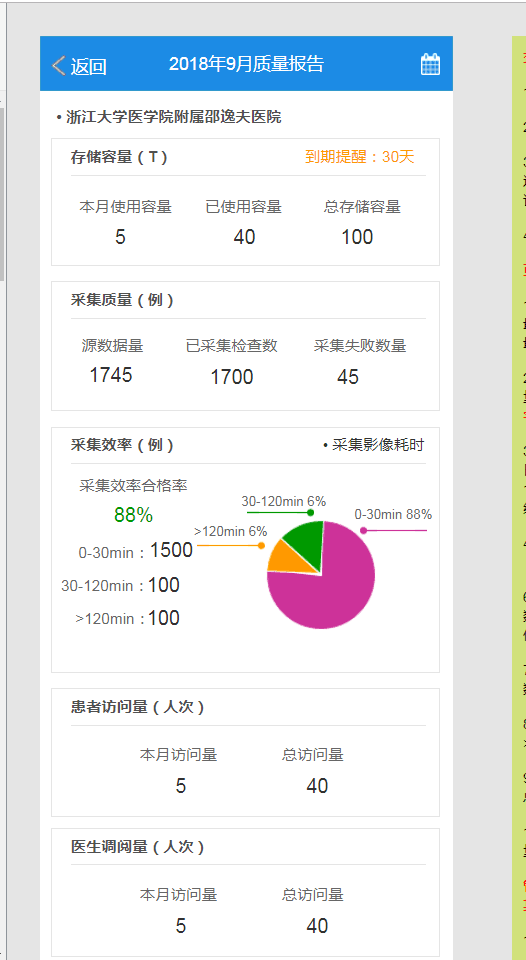
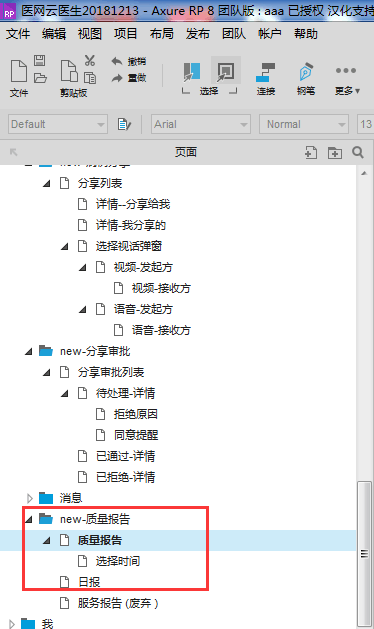
svn提交，修改采集服务映射忽略，修改采集服务版本返回错误，修正查询集成任务列表的入参

**！！！这里分支版本，把获取配置的代码改成不从debug获取了。要改回来，到时候。**

**把能力开放平台也变成了77的测试服务器，主干换成了77的数据库。**

15.

新的任务

袁凤娇 2019/2/27 16:20:35  
  
铿哥，app这边质量报告你需要提供接口  
  
袁凤娇 2019/2/27 16:21:01  
  
  
  
袁凤娇 2019/2/27 16:21:32  
  
  
  
袁凤娇 2019/2/27 16:21:44  
  
\\192.168.1.165\htdocs\yfj\医网云医生

据说是两个接口

16.

报错了

An exception was thrown while invoking the constructor 'Void .ctor(eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.PatientIndex], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.Visit], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationRequest], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationResult], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.DataSourceSetup])' on type 'PatientIndexService'. ---> 未将对象引用设置到对象的实例。 (See inner exception for details.)

Server stack trace:

在 System.ServiceModel.Channels.ServiceChannel.ThrowIfFaultUnderstood(Message reply, MessageFault fault, String action, MessageVersion version, FaultConverter faultConverter)

在 System.ServiceModel.Channels.ServiceChannel.HandleReply(ProxyOperationRuntime operation, ProxyRpc& rpc)

在 System.ServiceModel.Channels.ServiceChannel.Call(String action, Boolean oneway, ProxyOperationRuntime operation, Object[] ins, Object[] outs, TimeSpan timeout)

在 System.ServiceModel.Channels.ServiceChannelProxy.InvokeService(IMethodCallMessage methodCall, ProxyOperationRuntime operation)

在 System.ServiceModel.Channels.ServiceChannelProxy.Invoke(IMessage message)

Exception rethrown at [0]:

在 System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage reqMsg, IMessage retMsg)

在 System.Runtime.Remoting.Proxies.RealProxy.PrivateInvoke(MessageData& msgData, Int32 type)

在 IImageArchiveService.GetMediaList(MediaQuery query)

在 ImageArchiveServiceClient.GetMediaList(MediaQuery query)

An exception was thrown while invoking the constructor 'Void .ctor(eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.PatientIndex], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.Visit], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationRequest], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationResult], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.DataSourceSetup])' on type 'PatientIndexService'. ---> 未将对象引用设置到对象的实例。 (See inner exception for details.)

Server stack trace:

在 System.ServiceModel.Channels.ServiceChannel.ThrowIfFaultUnderstood(Message reply, MessageFault fault, String action, MessageVersion version, FaultConverter faultConverter)

在 System.ServiceModel.Channels.ServiceChannel.HandleReply(ProxyOperationRuntime operation, ProxyRpc& rpc)

在 System.ServiceModel.Channels.ServiceChannel.Call(String action, Boolean oneway, ProxyOperationRuntime operation, Object[] ins, Object[] outs, TimeSpan timeout)

在 System.ServiceModel.Channels.ServiceChannelProxy.InvokeService(IMethodCallMessage methodCall, ProxyOperationRuntime operation)

在 System.ServiceModel.Channels.ServiceChannelProxy.Invoke(IMessage message)

Exception rethrown at [0]:

在 System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage reqMsg, IMessage retMsg)

在 System.Runtime.Remoting.Proxies.RealProxy.PrivateInvoke(MessageData& msgData, Int32 type)

在 IImageArchiveService.GetServiceConfigList(ServiceConfigQuery query)

在 ImageArchiveServiceClient.GetServiceConfigList(ServiceConfigQuery query)

An exception was thrown while invoking the constructor 'Void .ctor(eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.PatientIndex], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.Visit], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationRequest], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.ObservationResult], eWorld.Core.Data.IImageArchiveRepository`1[eWorld.Entity.Domain.ImageArchive.DataSourceSetup])' on type 'PatientIndexService'. ---> 未将对象引用设置到对象的实例。 (See inner exception for details.)

"server=192.168.10.77;port=3306;database=ImageArchive;uid=sa;password=tomtaw;"

outstr = value.Replace(value.Substring(1, 1), hideflag);

这个匿名的方式，如果是黄黄，花花，这种，会全部变星号，比方黄娜娜，它是把“娜”字替换成\*，就全部变星号了。

17.

（1）采集流程，处理问题

（2）加密

（3）area

（4）处理云到云，一是校验，二是效率。

18.采集服务缺失接口：

（1）查询参数配置信息（Common数据库Setting表IDCAS系统参数）：

[Route("Job/GetSetting")]

var saveExamRes = \_imageArchiveClient.Invoke(t => t.GetSetting());

就是数据库Setting表中的云存储共享IDCAS有关的配置：

var setting = EngineContext.Current.Resolve<IDCASSettings>();

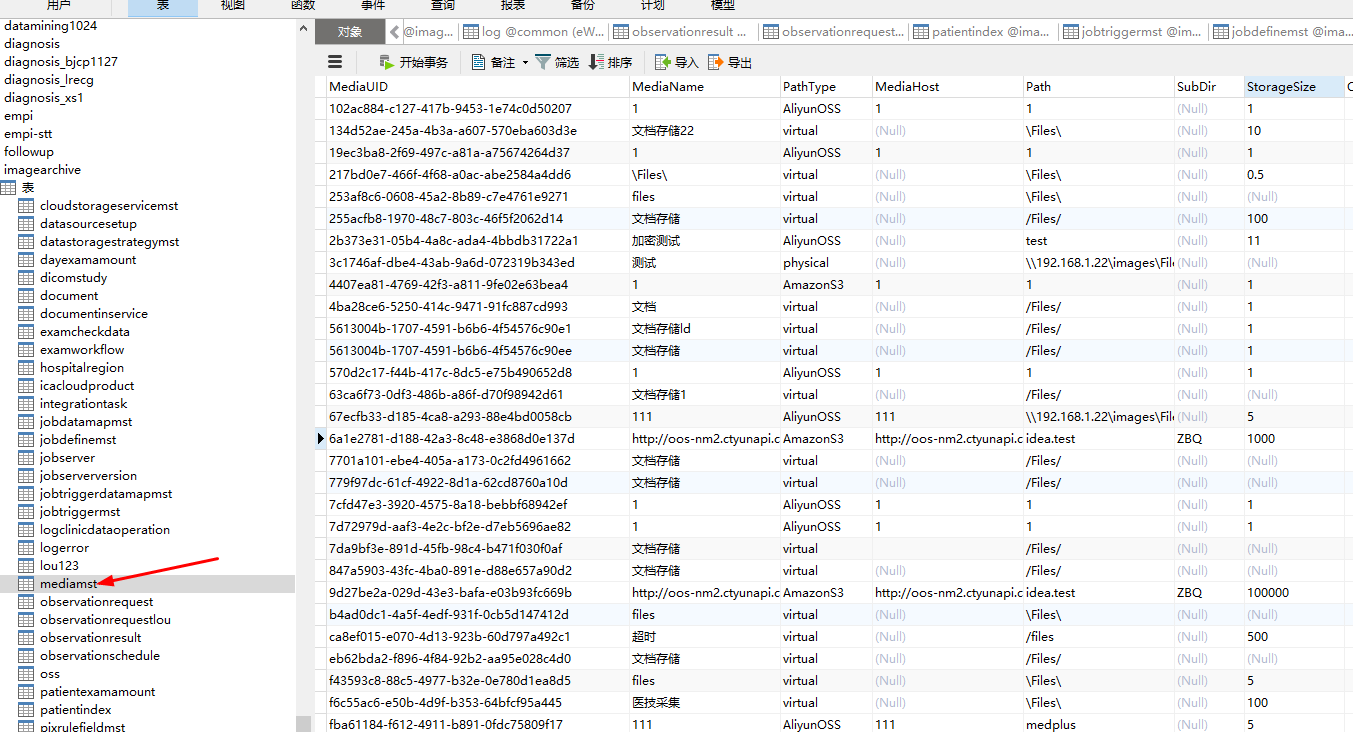
（2）获取存储媒质列表

string webApiUrl = AppConfig.WebApiUrl() + "/Job/GetMediaList";

var saveExamRes = \_imageArchiveClient.Invoke(t => t.GetMediaList(query));

return saveExamRes;

这个表里面就有容量大小（StorageSize）和已用容量（UsedSize），



（3）根据条件查询服务配置列表信息

string webApiUrl = AppConfig.WebApiUrl() + "/Job/GetServiceConfigList";

job控制器:

var saveExamRes = \_imageArchiveClient.Invoke(t => t.GetServiceConfigList(query));

（4）获取组织机构列表

string webApiUrl = AppConfig.WebApiUrl() + "/Job/GetInstitutionList";

job控制器:

这个估计要改调用军哥那边CRM的接口了：

var saveExamRes = \_crmClient.Invoke(t => t.GetInstitutionList(query));

（5）根据条件获取检查类型、就诊类型、检查科室

string webApiUrl = AppConfig.WebApiUrl() + "/Job/GetDataSourceSetUpList";

Job控制器，这里走的是Common的服务：

var result = \_imageArchiveClient.Invoke(t => t.GetDataSourceSetupList(query));

（6）检查信息校验结果上传（但是开会的时候说不需要校验了）（而且没有接受接口返回结果，似乎有没有实际用途）

string webApiUrl = AppConfig.WebApiUrl() + "/Verify/ExamDataCheck";

VerifyController：

result = \_imageArchiveClient.Invoke(t => t.SaveVerifyExamInfo(param.ToQuery()));

（7）保存文档存储信息，

string webApiUrl = AppConfig.WebApiUrl() + "/Job/SaveDocumentStorageList";

Jog控制器：

var saveRes = \_imageArchiveClient.Invoke(t => t.SaveDocumentStorageList(info));

（8）更新检查存储空间

string webApiUrl = AppConfig.WebApiUrl() + "/Job/UpdateExamDiskSpace";

Job控制器：

（9）编辑数据校验（获取云到云采集列表）

string webApiUrl = this.webApiUrl + "/Verify/ExamDataCheck";

（10）

获取能力开放的地址。

19.

这边验证走彪哥这边的，

到时候web还是走单点登陆。

20.

加了Base授权，ok了。

而且可以获取当前用户的Guid，UserId，ClientId了。

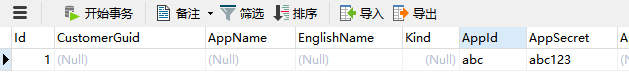
var guid = CurrentCustomerGuid;

var userId = CurrentUserId;

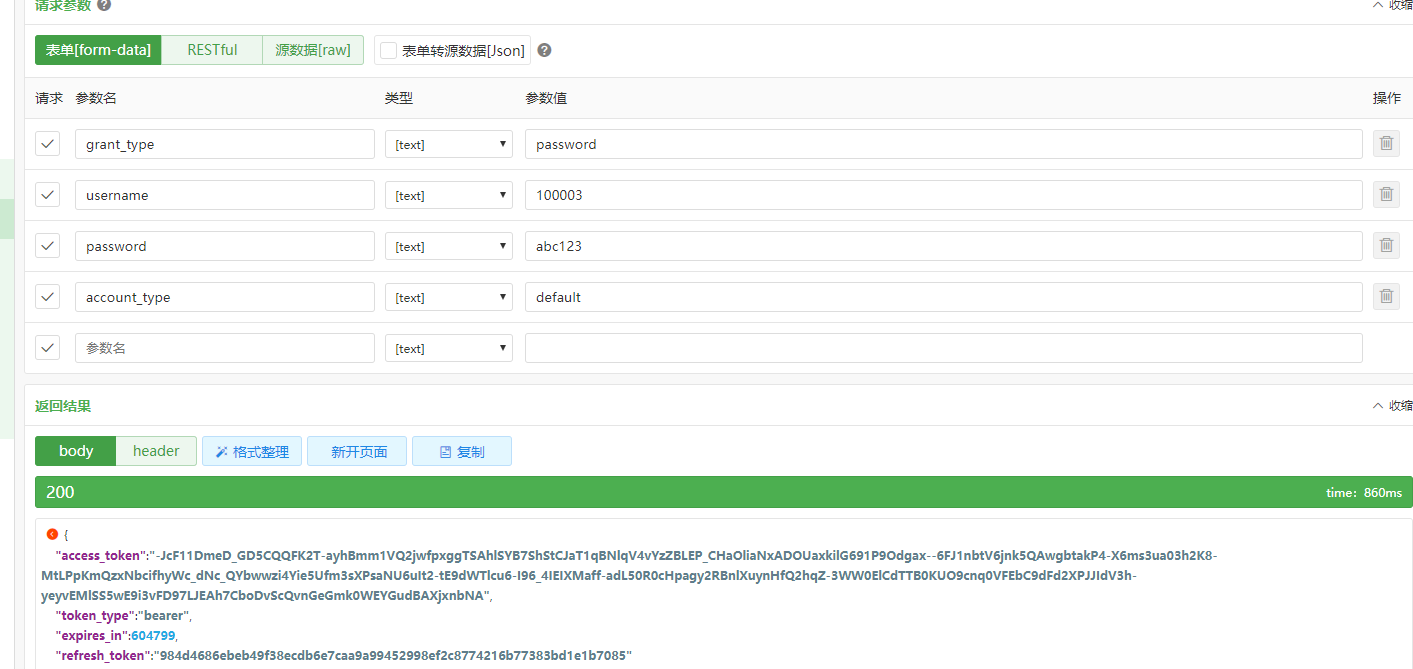
var clientId = ClientId;



数据库中：



直接用彪哥已经测试成功的token去访问：



var guid = CurrentCustomerGuid;

var userId = CurrentUserId;

var clientId = ClientId;

获取了数据：



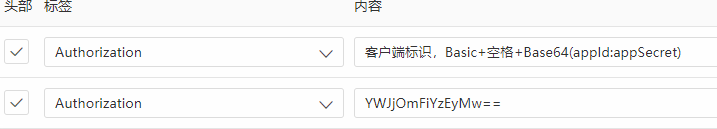
{ae9b17b9-89c5-433c-8688-6d868063741d}





这个用户难道就是上面的用户？

用户相同是因为Authorization头使用的是同一个appId:appSecret加密后的结果。



21.

云到云的问题：

（1）string的拼接，不是StringBuild。

（2）

22.

彪哥说插件，现在这样。越来越大，到时候一个G了。发布的接口的包，比别人的客户端还大。

让查查资料，想想看。

特别以后如果接口有V1，V2，每增加一个插件就要多几十G。

23.

云到云的校验。出方案。

设想，调用我的接口，把ObservationRequest的数据发过来，我这边做校验。

返回结果，如果是1，说明一致。如果是2。那服务这边接受到数据，将其置为1000，重新采集。

check\_mark，check\_state 校验标识，类似push\_state

校验结果，1：成功。2：失败。



校验的时候，与组织有关的数据可能会不同，因为可能两个平台所使用的，同一家医院的，机构ID是不一样的。

这里是观察申请信息的赋值：

examInfo.ObservationRequest = Mapper.Map<PatientObservationRequestDTO, ObservationRequestInfo>(obs);

examInfo.ObservationRequest.SourceOrganizationID = organizationID;

examInfo.ObservationRequest.SourceOrganizationName = organizationName;

examInfo.ObservationRequest.OrganizationID = model.OrganizationID;

examInfo.ObservationRequest.OrganizationName = model.OrganizationName;

24.

{SELECT

`Project1`.`ObservationUID`,

`Project1`.`PatientID`,

`Project1`.`PIDAssigningAuthority`,

`Project1`.`PatientMasterID`,

`Project1`.`PatientClass`,

`Project1`.`VisitUID`,

`Project1`.`OrderUID`,

`Project1`.`OrganizationID`,

`Project1`.`OrganizationName`,

`Project1`.`SourceOrganizationID`,

`Project1`.`SourceOrganizationName`,

`Project1`.`ClinicInfoType`,

`Project1`.`PlacerOrderNO`,

`Project1`.`PlacerAssigningAuthority`,

`Project1`.`PlacerOrderDetailNO`,

`Project1`.`FillerOrderNO`,

`Project1`.`FillerAssigningAuthority`,

`Project1`.`FillerPatientID`,

`Project1`.`AccessionNumber`,

`Project1`.`ServiceID`,

`Project1`.`ServiceCodeScheme`,

`Project1`.`ServiceText`,

`Project1`.`ServiceIDInsur`,

`Project1`.`ServiceSectID`,

`Project1`.`ProcedureID`,

`Project1`.`ProcedureName`,

`Project1`.`ProviderID`,

`Project1`.`ProviderName`,

`Project1`.`ProviderPhone`,

`Project1`.`RequestDeptID`,

`Project1`.`RequestDeptName`,

`Project1`.`RequestOrgID`,

`Project1`.`RequestOrgName`,

`Project1`.`RequestedDate`,

`Project1`.`Reason`,

`Project1`.`Attention`,

`Project1`.`Symptom`,

`Project1`.`AdverseReaction`,

`Project1`.`ClinicDiagnosis`,

`Project1`.`RelevantClinicalInfo`,

`Project1`.`FastingFlag`,

`Project1`.`TransportationMode`,

`Project1`.`SpecimenID`,

`Project1`.`SpecimenName`,

`Project1`.`SpecimenType`,

`Project1`.`SpecimenSource`,

`Project1`.`CollectorID`,

`Project1`.`CollectorName`,

`Project1`.`CollectionDept`,

`Project1`.`CollectionMethod`,

`Project1`.`CollectionVolume`,

`Project1`.`SpecimenReceivedDate`,

`Project1`.`ScheduledDate`,

`Project1`.`ScheduledDateEnd`,

`Project1`.`RegTime`,

`Project1`.`RegisterID`,

`Project1`.`RegisterName`,

`Project1`.`ObservationDeptID`,

`Project1`.`ObservationDeptName`,

`Project1`.`ObservationDate`,

`Project1`.`ObservationEndDate`,

`Project1`.`ObservationLocation`,

`Project1`.`ObservationEquipmentID`,

`Project1`.`ObservationEquipment`,

`Project1`.`EquipmentModel`,

`Project1`.`ObservationMethod`,

`Project1`.`StudyInstanceUID`,

`Project1`.`TechnicianID`,

`Project1`.`TechnicianName`,

`Project1`.`ResultAssistantID`,

`Project1`.`ResultAssistantName`,

`Project1`.`ResultAssistantPhone`,

`Project1`.`ResultPrincipalID`,

`Project1`.`ResultPrincipalName`,

`Project1`.`ResultReviseID`,

`Project1`.`ResultReviseName`,

`Project1`.`PreliminaryDate`,

`Project1`.`AuditDate`,

`Project1`.`ReviseDate`,

`Project1`.`ResultDate`,

`Project1`.`ResultOrganizationID`,

`Project1`.`ResultServiceCenterUID`,

`Project1`.`ResultStatus`,

`Project1`.`ResultStatusCode`,

`Project1`.`ResultPrintCount`,

`Project1`.`ResultPrintTime`,

`Project1`.`AbnormalFlags`,

`Project1`.`CriticalValue`,

`Project1`.`InfectionName`,

`Project1`.`PrivacyLevel`,

`Project1`.`PrivacyLevelExt`,

`Project1`.`Charges`,

`Project1`.`Payments`,

`Project1`.`PaymentsFlag`,

`Project1`.`FilmCount`,

`Project1`.`FilmNeed`,

`Project1`.`HasImage`,

`Project1`.`ImageLocation`,

`Project1`.`ConsultStatus`,

`Project1`.`ConsultRequestDate`,

`Project1`.`ConsultEndDate`,

`Project1`.`ArchiveDate`,

`Project1`.`LastUpdateDate`,

`Project1`.`DataSource`,

`Project1`.`LockFlag`,

`Project1`.`LockUserUID`,

`Project1`.`LockReason`,

`Project1`.`InWritingUserUID`,

`Project1`.`MessageCount`,

`Project1`.`UnProcessWorkflowCount`,

`Project1`.`PushState`,

`Project1`.`DeleteFlag`,

`Project1`.`RowTimestamp`,

`Project1`.`DiskSpace`,

`Project1`.`IsSummary`,

`Project1`.`IsRepeatExamSum`,

`Project1`.`IsDiseaseAnalysisSum`,

`Project1`.`CollectionMark`

FROM `ObservationRequest` AS `Project1`

WHERE ((0 = `Project1`.`DeleteFlag`) AND (0 = (((`Project1`.`CollectionMark`) / 100) % 10))) AND (`Project1`.`OrganizationID` = @p\_\_linq\_\_0)

ORDER BY

`Project1`.`ObservationDate` DESC LIMIT 20}

25.

苗哥这里的加解密还没有处理好，比方报告任务中，获取检查详情这里，机构id和名字我是没有加密的，而身份证我是加了密的。

if (result.data != null)

{

result.data.Name = DESHelper.Decrypt(result.data.Name);

result.data.Sex = DESHelper.Decrypt(result.data.Sex);

result.data.OrganizationName = DESHelper.Decrypt(result.data.OrganizationName);

result.data.OrganizationID = DESHelper.Decrypt(result.data.OrganizationID);

result.data.AccessionNumber = DESHelper.Decrypt(result.data.AccessionNumber);

}

26.

27.

28.

29.