后台地址配置

宿主启动cosmos参数时，需要填入Ip字段，通过Ip字段来配置后台地址，参考文档Cosmos大核SDK接入文档中"初始化环境"，支持多ip或域名填入

Ip填入规则：

单ip规则：ip:port; (ip和端口之间使用:分割，最后需要加上;) 示例：127.0.0.1:9999;

多ip规则：ip:port;ip1:port1;(ip和端口之间使用:分割, 多个地址之间使用;分割，需要加上;) 示例：127.0.0.1:9999;126.0.0.2:9999;

域名填入规则：与ip填入规则一致

支持域名和ip混合填写：

示例：127.0.0.1:9999;test.ip.com:9998;

接口设计

```
/**
 * 定义第三方查询参数的接口。
 * 该接口包含三个属性，用于描述第三方服务的查询请求。
 */
public interface IThirdRequestParameters
{
    /**
      * 获取或设置id。
      * 请求唯一标识
    */
    string uuid { get; set; }

    /**
     * 获取或设置操作名称。
     * 例如，"GetUserData" 或 "UpdateUser"。
     */
    string Action { get; set; }

    /**
     * 获取或设置查询参数。
     * 序列化串。
     */
    string Parameters { get; set; }

    /**
     * 获取或设置路由信息。
     * 用于指定请求的目标路由。
     * 为空默认发给upb_cosmos_plugin_req
     */
    string Server { get; set; }

    /// <summary>
    /// 订阅的topic
```

```
    /// 请求时该参数不填写，订阅时该字段必须填写
    /// </summary>
    string Topic { get; set; }
}

/// <summary>
/// 向第三方服务同步发送请求。
/// </summary>
/// <param name="parameters">请求参数</param>
/// <returns>应答结果</returns>
Task<IThirdResponse> ThirdModuleRequest(IThirdRequestParameters parameters);

/// <summary>
/// 向第三方服务异步发送请求。
/// </summary>
/// <param name="parameters">请求参数</param>
/// <returns>应答结果</returns>
Task ThirdModuleRequestAsync(IThirdRequestParameters parameters,
EventHandler<IThirdResponse> callback);

/// <summary>
/// 向第三方模块订阅数据
/// </summary>
/// <param name="subscriberParameters">订阅参数</param>
/// <param name="subscriber"></param>
/// <returns></returns>
Task<ITradeDataSubscription> SubscribThirdModule(ISubscribeParameters
subscriberParameters, EventHandler<IPushData> subscriber);

/// <summary>
/// 向第三方模块取消订阅数据
/// </summary>
/// <param name="unsubscriberParameters"></param>
/// <param name="subscriber"></param>
/// <returns></returns>
Task UnSubscribThirdModule(IUnSubscribeParameters unsubscriberParameters,
ITradeDataSubscription subscription);
```

请求路由

客户端如何将请求路由到不同的服务，根据IThirdQueryParameters类中的Router字段

该字段如何填写，需要询问服务提供方的服务名，根据后台定义填入对应Router，Action，Parameters参数进行请求、订阅等

组件如何调用

1.组件添加最新后台服务开发包:Cosmos.DataAccess.Trade.x.x.x.nupkg， 获取地址：组件开发包，找到最新版本开发包

2.组件开发者需要继承ICosmosTradeDataInteraction这个接口类

3.cosmos在创建组件过程中，会将后台服务的访问器赋值给组件的（ICosmosTradeDataInteraction.TradeAccessorsInjection）,通过该访问器，组件可直接访问后台服务

示例 参考示例

参考示例中和下面的请求订阅例子都是与后台pluginserver1服务做的数据交互，方法和参数都是根据pluginserver1中提供的填写的，如果测试的为别的服务，具体参数需要和后台服务提供方确认

请求

```
 private async void btn_req_higate_Click(object sender, RoutedEventArgs e)
 {
     /// 调用第三方接口
     IThirdRequestParameters requestParameters =
_tradeDataAccessor.DataProvider.CreateThirdRequestParameters();
     requestParameters.Action = "request.plugin.test";
     requestParameters.Server = "pluginserver1";
     requestParameters.uuid = Guid.NewGuid().ToString();
     var options = new JsonSerializerOptions
     {
         Encoder = JavaScriptEncoder.Create(
             allowedRanges: new[]
             {
                 UnicodeRanges.BasicLatin,      // ASCII 字符
                 UnicodeRanges.CjkUnifiedIdeographs // 中文字符（CJK 统一表意文字）
             }
         )
     };
     JsonObject context = new JsonObject();
     context["name"] = text_higate.Text;
     requestParameters.Parameters = JsonSerializer.Serialize(context, options);

     //同步等待应答
     {
         var result = await
_tradeDataAccessor.DataProvider.ThirdModuleRequest(requestParameters);
         _logger?.Log(CosmosLogLevel.Information, $"reqresult:{result.data} id :
{result.uuid} code:{result.code}  msg:{result.msg} ");
     }

     //异步不卡住界面
     {

        await
_tradeDataAccessor.DataProvider.ThirdModuleRequestAsync(requestParameters,
(object sender, IThirdResponse result) =>
        {
            _logger?.Log(CosmosLogLevel.Information, $"reqresult:{result.data}
id : {result.uuid} code:{result.code}  msg:{result.msg} ");
        });
     }

 }
```

订阅

```
private async void btn_sub_higate_Click(object sender, RoutedEventArgs e)
{
```

```
    /// 调用第三方订阅接口
    ISubscribeParameters subscribeParameters =
_tradeDataAccessor.DataProvider.CreateSubscribeParameters();
    subscribeParameters.Action = "subscribe.plugin.test";
    subscribeParameters.Server = "pluginserver1";
    subscribeParameters.Parameters = "date=today";
    subscribeParameters.Topic = "entrust";
    subscribeParameters.uuid = Guid.NewGuid().ToString();
    pushsubscribetion_ = await
_tradeDataAccessor.DataProvider.SubscribThirdModule(subscribeParameters, (object
sender, IPushData pushResult) =>
    {
        _logger?.Log(CosmosLogLevel.Information, $" RecvPushData :
{pushResult.data}");
    });

    if(pushsubscribetion_.code != 0)
    {
        WpfToast.Show($"订阅失败 msg:{pushsubscribetion_.msg}",
CosmosLogLevel.Error);
        _logger?.Log(CosmosLogLevel.Information, $"订阅失败 msg:
{pushsubscribetion_.msg}");
    }
    else
    {
        WpfToast.Show("订阅higate完成");
        _logger?.Log(CosmosLogLevel.Information, $"订阅higate成功 code:
{pushsubscribetion_.code}, msg:{pushsubscribetion_.msg}");
    }
}
```

取消订阅

```
private async void btn_unsub_higate_Click(object sender, RoutedEventArgs e)
{
    if (pushsubscribetion_!=null)
    {
        /// 调用第三方取消订阅接口
        IUnSubscribeParameters unsubscribeParameters =
_tradeDataAccessor.DataProvider.CreateUnSubscribeParameters();
        unsubscribeParameters.Action = "unsubscribe.plugin.test";
        unsubscribeParameters.Server = "pluginserver1";
        unsubscribeParameters.Parameters = "topic=entrust,date=today";
        unsubscribeParameters.Topic = "entrust";
        unsubscribeParameters.uuid = Guid.NewGuid().ToString();

        var result = await
_tradeDataAccessor.DataProvider.UnSubscribThirdModule(unsubscribeParameters,
pushsubscribetion_);
        if(result.code == 0)
        {
            WpfToast.Show("取消订阅higate成功");
            _logger?.Log(CosmosLogLevel.Information, "取消订阅higate成功");
        }
        else
        {
            WpfToast.Show($"取消订阅失败 msg:{result.msg}", CosmosLogLevel.Error);
```

```
            _logger?.Log(CosmosLogLevel.Information, $"取消订阅失败 msg:
{result.msg}");

        }
    }
}
```

订阅通道链接状态

```
 //订阅连接是否断开
_tradeDataAccessor.DataSessionController.TradeStatusChanged +=
DataSessionController_TradeStatusChanged;

 private void DataSessionController_TradeStatusChanged(object? sender,
TradeDataSessionStatus e)
 {
     //已连接
     if(e == TradeDataSessionStatus.Running)
     {
         //处理查询或者重新订阅数据
         _logger?.Log(CosmosLogLevel.Information, "已连接");
     }
     //断开连接
     else if(e == TradeDataSessionStatus.Stopped)
     {
         _logger?.Log(CosmosLogLevel.Information, "已断开");

     }
 }
```