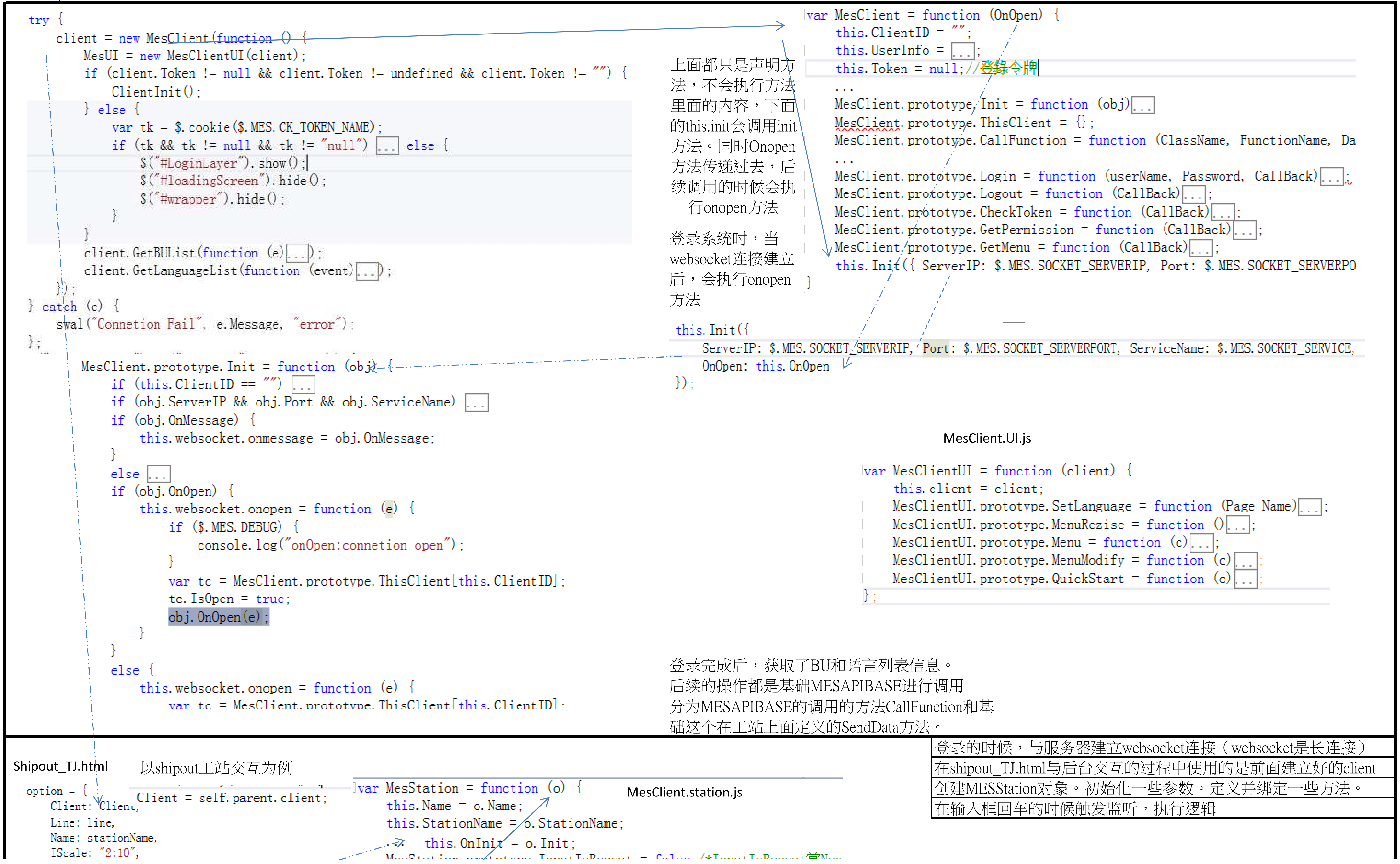
虚线不带箭头，代表展开后看到的细节
虚线带箭头，是值映射关系
实现带箭头是真正执行调用

连接的时候服务名和端口与后台一致



Mes.js

```
try {
    client = new MesClient(function () {
        MesUI = new MesClientUI(client);
        if (client.Token != null && client.Token != undefined && client.Token != "") {
            ClientInit();
        } else {
            var tk = $.cookie($.MES.CK_TOKEN_NAME);
            if (tk && tk != null && tk != "null") ... else {
                $("#LoginLayer").show();
                $("#loadingScreen").hide();
                $("#wrapper").hide();
            }
        }
        client.GetBUList(function (e)...);
        client.GetLanguageList(function (event)...);
    });
} catch (e) {
    swal("Connetion Fail", e.Message, "error");
};
```

```
var MesClient = function (OnOpen) {
    this.ClientID = "";
    this.UserInfo = ...;
    this.Token = null;//登錄令牌
    ...
    MesClient.prototype.Init = function (obj)...
    MesClient.prototype.ThisClient = {};
    MesClient.prototype.CallFunction = function (ClassName, FunctionName, Da
    ...
    MesClient.prototype.Login = function (userName, Password, CallBack)...;
    MesClient.prototype.Logout = function (CallBack)...;
    MesClient.prototype.CheckToken = function (CallBack)...;
    MesClient.prototype.GetPermission = function (CallBack)...;
    MesClient.prototype.GetMenu = function (CallBack)...;
    this.Init({ ServerIP: $.MES.SOCKET_SERVERIP, Port: $.MES.SOCKET_SERVERPO
}
```

上面都只是声明方法，不会执行方法里面的内容，下面的this.init会调用init方法。同时Onopen方法传递过去，后续调用的时候会执行onopen方法

登录系统时，当websocket连接建立后，会执行onopen方法

```
this.Init({
    ServerIP: $.MES.SOCKET_SERVERIP, Port: $.MES.SOCKET_SERVERPORT, ServiceName: $.MES.SOCKET_SERVICE,
    OnOpen: this.OnOpen
});
```

```
MesClient.prototype.Init = function (obj) {
    if (this.ClientID == "") ...
    if (obj.ServerIP && obj.Port && obj.ServiceName) ...
    if (obj.OnMessage) {
        this.websocket.onmessage = obj.OnMessage;
    }
    else ...
    if (obj.OnOpen) {
        this.websocket.onopen = function (e) {
            if ($.MES.DEBUG) {
                console.log("onOpen:connetion open");
            }
            var tc = MesClient.prototype.ThisClient[this.ClientID];
            tc.IsOpen = true;
            obj.OnOpen(e);
        }
    }
    else {
        this.websocket.onopen = function (e) {
            var tc = MesClient.prototype.ThisClient[this.ClientID];
```

MesClient.UI.js

```
var MesClientUI = function (client) {
    this.client = client;
    MesClientUI.prototype.SetLanguage = function (Page_Name)...;
    MesClientUI.prototype.MenuRezise = function ()...;
    MesClientUI.prototype.Menu = function (c)...;
    MesClientUI.prototype.MenuModify = function (c)...;
    MesClientUI.prototype.QuickStart = function (o)...;
};
```

登录完成后，获取了BU和语言列表信息。
后续的操作都是基础MESAPIBASE进行调用
分为MESAPIBASE的调用的方法CallFunction和基础这个在工站上面定义的SendData方法。

登录的时候，与服务器建立websocket连接（websocket是长连接）
在shipout_TJ.html与后台交互的过程中使用的是前面建立好的client
创建MESStation对象。初始化一些参数。定义并绑定一些方法。
在输入框回车的时候触发监听，执行逻辑

Shipout_TJ.html

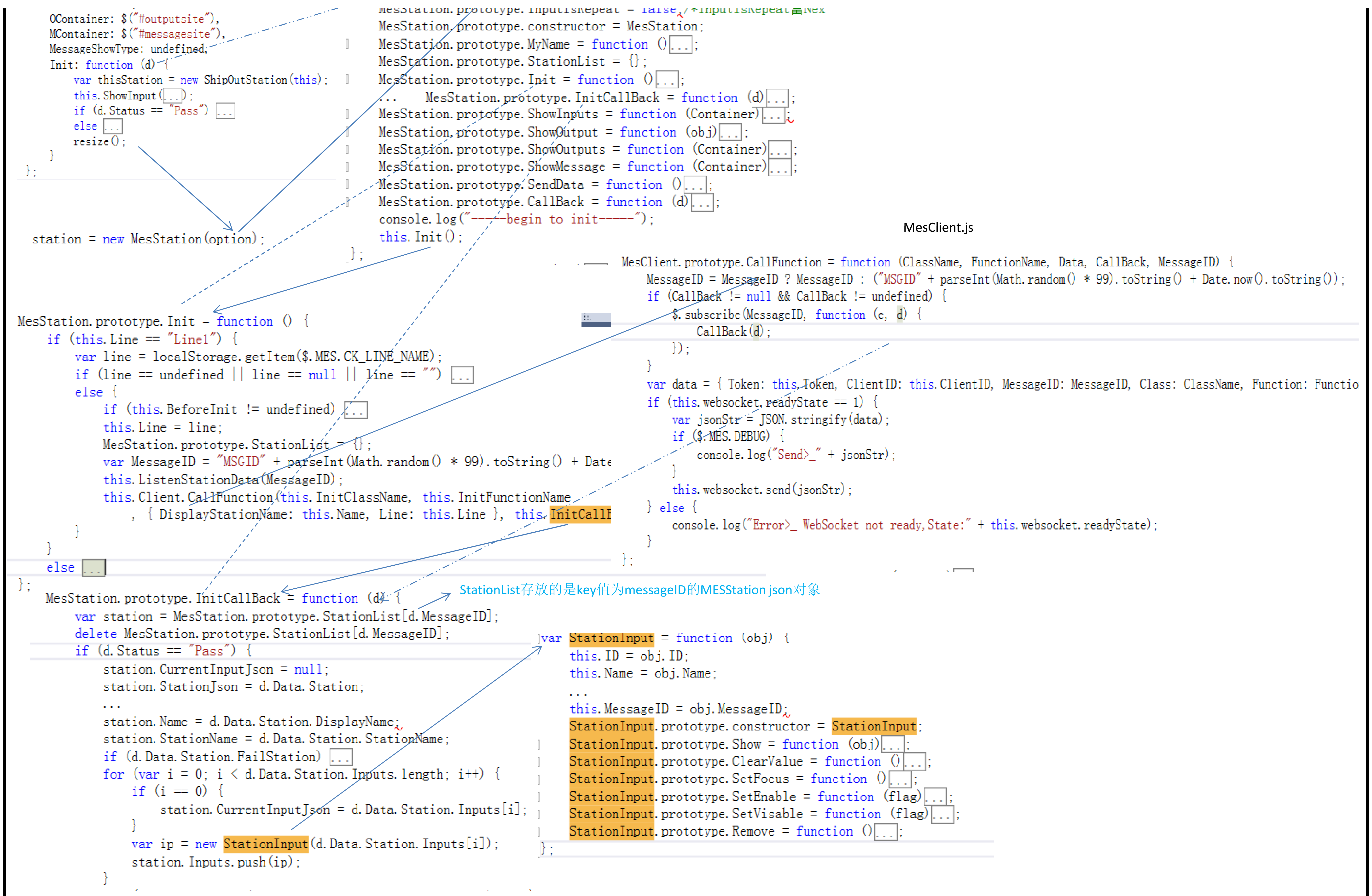
以shipout工站交互为例

```
option = {
    Client: Client,
    Line: line,
    Name: stationName,
    IScale: "2:10",
```

```
Client = self.parent.client;
```

MesClient.station.js

```
var MesStation = function (o) {
    this.Name = o.Name;
    this.StationName = o.StationName;
        this.OnInit = o.Init;
    MesStation.prototype.InputIsRepeat = false;/*InputIsRepeat
```



```
    OContainer: $("#outputsite"),
    MContainer: $("#messagesite"),
    MessageShowType: undefined,
    Init: function (d) {
        var thisStation = new ShipOutStation(this);
        this.ShowInput(...);
        if (d.Status == "Pass") ...
        else ...
        resize();
    }
};

station = new MesStation(option);
```

```
MesStation.prototype.InputIsRepeat = false;/*InputIsRepeat[illegible]Nex
MesStation.prototype.constructor = MesStation;
MesStation.prototype.MyName = function ()...;
MesStation.prototype.StationList = {};
MesStation.prototype.Init = function ()...;
...     MesStation.prototype.InitCallBack = function (d)...;
MesStation.prototype.ShowInputs = function (Container)...;
MesStation.prototype.ShowOutput = function (obj)...;
MesStation.prototype.ShowOutputs = function (Container)...;
MesStation.prototype.ShowMessage = function (Container)...;
MesStation.prototype.SendData = function ()...;
MesStation.prototype.CallBack = function (d)...;
console.log("-----begin to init-----");
this.Init();
};
```

MesClient.js

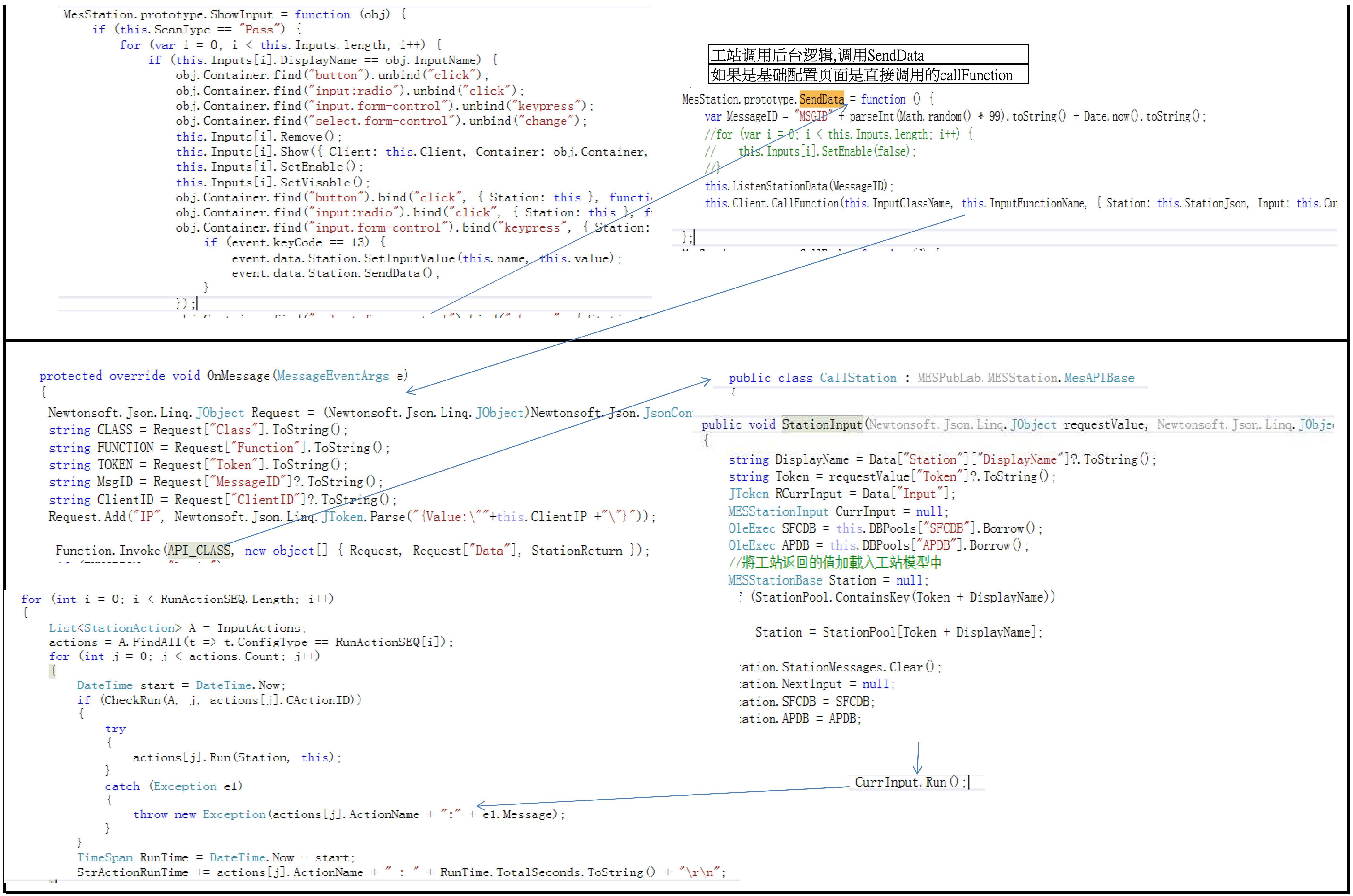
```
MesClient.prototype.CallFunction = function (ClassName, FunctionName, Data, CallBack, MessageID) {
    MessageID = MessageID ? MessageID : ("MSGID" + parseInt(Math.random() * 99).toString() + Date.now().toString());
    if (CallBack != null && CallBack != undefined) {
        $.subscribe(MessageID, function (e, d) {
            CallBack(d);
        });
    }
    var data = { Token: this.Token, ClientID: this.ClientID, MessageID: MessageID, Class: ClassName, Function: Functio
    if (this.websocket.readyState == 1) {
        var jsonStr = JSON.stringify(data);
        if ($.MES.DEBUG) {
            console.log("Send>_" + jsonStr);
        }
        this.websocket.send(jsonStr);
    } else {
        console.log("Error>_ WebSocket not ready,State:" + this.websocket.readyState);
    }
};
```

```
MesStation.prototype.Init = function () {
    if (this.Line == "Line1") {
        var line = localStorage.getItem($.MES.CK_LINE_NAME);
        if (line == undefined || line == null || line == "") ...
        else {
            if (this.BeforeInit != undefined) ...
            this.Line = line;
            MesStation.prototype.StationList = {};
            var MessageID = "MSGID" + parseInt(Math.random() * 99).toString() + Date
            this.ListenStationData(MessageID);
            this.Client.CallFunction(this.InitClassName, this.InitFunctionName
                , { DisplayStationName: this.Name, Line: this.Line }, this.InitCallB
        }
    }
    else ...
};
```

```
MesStation.prototype.InitCallBack = function (d) {
    var station = MesStation.prototype.StationList[d.MessageID];
    delete MesStation.prototype.StationList[d.MessageID];
    if (d.Status == "Pass") {
        station.CurrentInputJson = null;
        station.StationJson = d.Data.Station;
        ...
        station.Name = d.Data.Station.DisplayName;
        station.StationName = d.Data.Station.StationName;
        if (d.Data.Station.FailStation) ...
        for (var i = 0; i < d.Data.Station.Inputs.length; i++) {
            if (i == 0) {
                station.CurrentInputJson = d.Data.Station.Inputs[i];
            }
            var ip = new StationInput(d.Data.Station.Inputs[i]);
            station.Inputs.push(ip);
        }
```

StationList存放的是key值为messageID的MESStation json对象

```
var StationInput = function (obj) {
    this.ID = obj.ID;
    this.Name = obj.Name;
    ...
    this.MessageID = obj.MessageID;
StationInput.prototype.constructor = StationInput;
StationInput.prototype.Show = function (obj)...;
StationInput.prototype.ClearValue = function ()...;
StationInput.prototype.SetFocus = function ()...;
StationInput.prototype.SetEnable = function (flag)...;
StationInput.prototype.SetVisable = function (flag)...;
StationInput.prototype.Remove = function ()...;
};
```



```
MesStation.prototype.ShowInput = function (obj) {
    if (this.ScanType == "Pass") {
        for (var i = 0; i < this.Inputs.length; i++) {
            if (this.Inputs[i].DisplayName == obj.InputName) {
                obj.Container.find("button").unbind("click");
                obj.Container.find("input:radio").unbind("click");
                obj.Container.find("input.form-control").unbind("keypress");
                obj.Container.find("select.form-control").unbind("change");
                this.Inputs[i].Remove();
                this.Inputs[i].Show({ Client: this.Client, Container: obj.Container,
                this.Inputs[i].SetEnable();
                this.Inputs[i].SetVisable();
                obj.Container.find("button").bind("click", { Station: this }, functi
                obj.Container.find("input:radio").bind("click", { Station: this }, f
                obj.Container.find("input.form-control").bind("keypress", { Station:
                    if (event.keyCode == 13) {
                        event.data.Station.SetInputValue(this.name, this.value);
                        event.data.Station.SendData();
                    }
                });
```

工站调用后台逻辑,调用SendData
如果是基础配置页面是直接调用的callFunction

```
MesStation.prototype.SendData = function () {
    var MessageID = "MSGID" + parseInt(Math.random() * 99).toString() + Date.now().toString();
    //for (var i = 0; i < this.Inputs.length; i++) {
    //    this.Inputs[i].SetEnable(false);
    //}
    this.ListenStationData(MessageID);
    this.Client.CallFunction(this.InputClassName, this.InputFunctionName, { Station: this.StationJson, Input: this.Cu

};
```

```
protected override void OnMessage(MessageEventArgs e)
{
 Newtonsoft.Json.Linq.JObject Request = (Newtonsoft.Json.Linq.JObject)Newtonsoft.Json.JsonCon
 string CLASS = Request["Class"].ToString();
 string FUNCTION = Request["Function"].ToString();
 string TOKEN = Request["Token"].ToString();
 string MsgID = Request["MessageID"]?.ToString();
 string ClientID = Request["ClientID"]?.ToString();
 Request.Add("IP", Newtonsoft.Json.Linq.JToken.Parse("{Value:\""+this.ClientIP +"\"}"));

  Function.Invoke(API_CLASS, new object[] { Request, Request["Data"], StationReturn });
```

```
public class CallStation : MESPubLab.MESStation.MesAPIBase
{

public void StationInput(Newtonsoft.Json.Linq.JObject requestValue, Newtonsoft.Json.Linq.JObje
{
    string DisplayName = Data["Station"]["DisplayName"]?.ToString();
    string Token = requestValue["Token"]?.ToString();
    JToken RCurrInput = Data["Input"];
    MESStationInput CurrInput = null;
    OleExec SFCDB = this.DBPools["SFCDB"].Borrow();
    OleExec APDB = this.DBPools["APDB"].Borrow();
    //將工站返回的值加載入工站模型中
    MESStationBase Station = null;
     (StationPool.ContainsKey(Token + DisplayName))

       Station = StationPool[Token + DisplayName];

     ation.StationMessages.Clear();
     ation.NextInput = null;
     ation.SFCDB = SFCDB;
     ation.APDB = APDB;

            CurrInput.Run();
```

```
for (int i = 0; i < RunActionSEQ.Length; i++)
{
    List<StationAction> A = InputActions;
    actions = A.FindAll(t => t.ConfigType == RunActionSEQ[i]);
    for (int j = 0; j < actions.Count; j++)
    {
        DateTime start = DateTime.Now;
        if (CheckRun(A, j, actions[j].CActionID))
        {
            try
            {
                actions[j].Run(Station, this);
            }
            catch (Exception e1)
            {
                throw new Exception(actions[j].ActionName + ":" + e1.Message);
            }
        }
        TimeSpan RunTime = DateTime.Now - start;
        StrActionRunTime += actions[j].ActionName + " : " + RunTime.TotalSeconds.ToString() + "\r\n";
```