# Logic Monitor设计

1. 整体设计思路
   1. 只存储comm100字段与salesforce字段的映射关系。
   2. Salesforce字段和对象即用即取。
2. 数据库设计：general库，新增t\_LiveChat\_SalesforceIntegration表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Null** | **Default** | **Description** |
| SiteId | Int | Not null |  |  |
| Salesforce\_AccessToken | Nvarchar(1024) | Not null | ‘’ | 需要将原site 库t\_livechat\_integration表中Salesforce\_AccessToken导到这个表中 |
| Salesforce\_InstanceUrl | Nvarchar(1024) | Not null | ‘’ | 需要将原site 库t\_livechat\_integration表中Salesforce\_InstanceUrl导到这个表中 |
| EnumSalesforcePurposeMessage | Smallint | Not null | 0 | 根据message创建什么  0: contact & case  1: lead  2: case  3:do nothing  对老数据处理：  需要将原site 库t\_livechat\_integration表中EnumSalesforcePurpose导到这个表中 |
| EnumSalesforceUpdateContactMessage | Smallint | Nut null | 0 | 更新contact的类型  0：不更新  1：更新全部字段  2：更新部分字段  需要将原site 库t\_livechat\_integration表中EnumSalesforceUpdateContact导到这个表中 |
| IfAutomaticallyCreateFromChat | bit | Not null | 0 | 是否根据chat自动创建  0：手动创建  1：自动创建 |
| EnumSalesforcePurposeChat | Smallint | Not null | 0 | 根据chat创建什么  0: contact & case  1: lead  2: case  3:do nothing |
| EnumSalesforceUpdateContactChat | Smallint | Not null | 0 | 更新contact的类型  0：不更新  1：更新全部字段  2：更新部分字段 |
| IfDisplayAccountInfoInVM | bit | not null | 0 | 控制VM中是否显示account信息  0：不显示  1：显示 |
| IfDisplayContactInfoInVM | bit | not null | 0 | 控制VM中是否显示contact信息  0：不显示  1：显示 |
| SalesforceAccountFieldsDisplayInVM | nvarchar(max) | not null | ‘’ | json格式，字符串数组:  [“field1”, “field2”]，存的是system name，显示时顺序按存的这个顺序显示 |
| SalesforceContactFieldsDisplayInVM | nvarchar(max) | not null | ‘’ | json格式，字符串数组:  [“field1”, “field2”] ，存的是system name，显示时顺序按存的这个顺序显示 |
| AccountIdentitySalesforceField | nvarchar(256) | not null | ‘’ | Salesforce account字段的系统名（account第一个mapping） |
| AccountIdentityComm100Field | nvarchar(256) | not null | ‘’ | Comm100 字段的系统名（account第一个mapping） |
| SecondAccountIdentitySalesforceField | nvarchar(256) | not null | ‘’ | Salesforce account字段的系统名（account第二个mapping） |
| SecondAccountIdentityComm100Field | nvarchar(256) | not null | ‘’ | Comm100 字段的系统名（account第二个mapping） |
| ContactIdentitySalesforceField | nvarchar(256) | not null | ‘’ | Salesforce contact字段的系统名（contact第一个mapping） |
| ContactIdentityComm100Field | nvarchar(256) | not null | ‘’ | Comm100 字段的系统名（contact第一个mapping） |
| SecondContactIdentitySalesforceField | nvarchar(256) | not null | ‘’ | Salesforce contact字段的系统名（contact第二个mapping） |
| SecondContactIdentityComm100Field | nvarchar(256) | not null | ‘’ | Comm100 字段的系统名（contact第二个mapping） |

1. EnumSalesforcePurpose增加：
   1. CreateCase = 2
   2. DoNothing = 3
2. 增加对象：
   1. 增加StruIntegrationObject
      1. IfHasContact bool
      2. ContactFields List<StruIntegrationFieldAndValue>
      3. IfHasAccount bool
      4. AccountFields List<StruIntegrationFieldAndValue>
      5. IfHasLead bool
      6. LeadFileds List<StruIntegrationFieldAndValue>
      7. Error StruError
   2. 增加StruIntegrationFieldAndValue
      1. Label string ---for display
      2. Name string ----system name
      3. Value string ----field value
      4. fieldType int ----field type: 0(string), 1(bool), 2(double), 3(date)
      5. SalesforcefieldType // 需要知道salesforce是什么类型
3. Salesforce接口，
   1. chatserver中展示Account/Contact信息在原来的集成接口CRMIntegrationService.asmx增加方法：// 为什么要知道所有字段
      1. GetSalesforceFields()，用于integration展示字段、routing rule设置条件
         1. 参数：int siteId
         2. 返回：StruIntegrationObject, 包含ContactFields和AccountFields
      2. GetSFContactAndRelatedAccount() // 方法是要改名字的,实际应该是判断account 和 contact , 然后再判断 是否有Lead
         1. 参数：int siteId, string contactIdendityField1, string contactValue1, string contactIdendityField2, string contactValue2, string accountIdendityField1, string accountValue1, string accountIdendityField2, string accountValue2
         2. 返回：StruIntegrationObject，包含ContactFields和AccountFields，若匹配不到对应的contact，则ContactFields为null，同理若匹配不到account，AccountFields为null, 匹配不到Lead, LeadField为空
   2. chatserver调用ChatEnded方法时（生成salesforce 对象），如果在聊天过程中有手动创建，点了按钮后就创建，聊天结束后再更新；如果自动创建，跟原来的逻辑一样），原来的接口增加参数：
      1. IfAutomaticallyCreateFromChat
      2. EnumSalesforcePurposeChat
      3. EnumSalesforceUpdateContactChat
      4. String id（如果在聊天过程中已经创建过对应的object，则为该obj的id，如果没有，则为空） //现在会有2个对象。
   3. 手动创建时调用的接口CreateSFObject
      1. 参数：int siteId

string name

string email

string company

string phone

List<int> operatorIds

EnumSalesforcePurposeChat

EnumSalesforceUpdateContactChat

* + 1. 返回：StruStringValue，如果选择创建lead，则返回对应的lead id；如果创建case，则对应case Id，如果是attach case to contact，则返回对应的case id，用于聊天结束时更新聊天内容

1. Portal
   1. 增加SalesforceIntegration类
      1. 属性

|  |  |
| --- | --- |
| \_conn | SqlConnection |
| \_transaction | SqlTransaction |
| SiteId | Int |
| Salesforce\_AccessToken | string |
| Salesforce\_InstanceUrl | string |
| EnumSalesforcePurposeMessage | EnumSalesforcePurpose |
|
|
|
|
|
|
| EnumSalesforceUpdateContactMessage | EnumSalesforceUpdateContact |
|
|
|
|
| IfAutomaticallyCreateFromChat | bool |
|
|
| EnumSalesforcePurposeChat | EnumSalesforcePurpose |
|
|
|
|
| EnumSalesforceUpdateContactChat | EnumSalesforceUpdateContact |
|
|
|
| IfDisplayAccountInfoInVM | bool |
|
|
| IfDisplayContactInfoInVM | bool |
|
|
| SalesforceAccountFieldsDisplayInVM | string |
|
| SalesforceContactFieldsDisplayInVM | string |
|
| AccountIdentitySalesforceField | string |
| AccountIdentityComm100Field | string |
| SecondAccountIdentitySalesforceField | string |
| SecondAccountIdentityComm100Field | string |
| ContactIdentitySalesforceField | string |
| ContactIdentityComm100Field | string |
| SecondContactIdentitySalesforceField | string |
| SecondContactIdentityComm100Field | string |

* + 1. SalesforceIntegration(SqlConnection conn, SqlTransaction transaction, int siteId)
    2. Save(siteId，salesforce\_AccessToken，salesforce\_InstanceUrl，enumSalesforcePurposeMessage，enumSalesforceUpdateContactMessage，ifAutomaticallyCreateFromChat，enumSalesforcePurposeChat，enumSalesforceUpdateContactChat，ifDisplayAccountInfoInVM，ifDisplayContactInfoInVM，salesforceAccountFieldsDisplayInVM，SsalesforceContactFieldsDisplayInVM，accountIdentitySalesforceField，accountIdentityComm100Field，secondAccountIdentitySalesforceField，secondAccountIdentityComm100Field，contactIdentitySalesforceField，contactIdentityComm100Field，secondContactIdentitySalesforceField，secondContactIdentityComm100Field)
       1. 调用access层时，需要判断是否存在这个站点的记录，如果不存在就insert一条记录，如果已存在就update
       2. 用于界面上connect salesforce和保存salesforce集成设置
    3. Disconnect (siteId)，删除这个站点的记录，用于disconnect salesforce集成
  1. 对应的增加SalesforceIntegrationProcess.cs
     1. SalesforceIntegration GetSalesforceIntegration(siteId)
     2. Save(conn, transaction, siteId，salesforce\_AccessToken，salesforce\_InstanceUrl，enumSalesforcePurposeMessage，enumSalesforceUpdateContactMessage，ifAutomaticallyCreateFromChat，enumSalesforcePurposeChat，enumSalesforceUpdateContactChat，ifDisplayAccountInfoInVM，ifDisplayContactInfoInVM，salesforceAccountFieldsDisplayInVM，SsalesforceContactFieldsDisplayInVM，accountIdentitySalesforceField，accountIdentityComm100Field，secondAccountIdentitySalesforceField，secondAccountIdentityComm100Field，contactIdentitySalesforceField，contactIdentityComm100Field，secondContactIdentitySalesforceField，secondContactIdentityComm100Field)
     3. Disconnect
  2. 对应增加SalesforceIntegrationAccess.cs
     1. GetSalesforceIntegration(SqlConnection conn, SqlTransaction transaction, int siteId)
     2. Save(conn, transaction, siteId，salesforce\_AccessToken，salesforce\_InstanceUrl，enumSalesforcePurposeMessage，enumSalesforceUpdateContactMessage，ifAutomaticallyCreateFromChat，enumSalesforcePurposeChat，enumSalesforceUpdateContactChat，ifDisplayAccountInfoInVM，ifDisplayContactInfoInVM，salesforceAccountFieldsDisplayInVM，SsalesforceContactFieldsDisplayInVM，accountIdentitySalesforceField，accountIdentityComm100Field，secondAccountIdentitySalesforceField，secondAccountIdentityComm100Field，contactIdentitySalesforceField，contactIdentityComm100Field，secondContactIdentitySalesforceField，secondContactIdentityComm100Field)
     3. Disconnect(SqlConnection conn, SqlTransaction transaction, int siteId)
  3. 原来Integration类去掉salesforce相关字段：accessToken, instanceUrl, enumsalesforcePurpose, enumSalesforceUpdatecontact
  4. Salesforce integration页面，
     1. 如果原来已经connect，则在页面加载时通过接口GetSalesforceFields()同步获取salesforce字段
     2. 如果原来disconnect，则在connect完成后通过接口GetSalesforceFields()同步获取salesforce字段（取的是Create 页面的所有字段, 不是所有字段）
     3. 显示时用字段的label，保存到数据时用name
     4. Comm100 field：
        1. Visitor：列出Name，email，IP，country/Region，state/Province，city
        2. Pre-chat：列出所有code plan的pre-chat字段，如果重名（display text一样）则只显示一次
        3. Custom variable：列出所有自定义变量
        4. 显示使用字段的display text，保存时
           1. Visitor的字段直接使用system name
           2. Pre-chat系统字段使用system name，自定义字段按格式\_\_PreChat\_Custom\_DisplayText保存
           3. Custom variable字段使用按格式\_\_CustomVariable\_name保存
     5. 选中的要显示在VM中的salesforce字段有顺序关系，保存时需按顺序保存
     6. 编辑时，如果原来设置的salesforce字段没有在这次取到的字段列表中（根据保存的字段name与获取到的salesforce字段name匹配），则认为这个字段已经被删除，就不再显示出来
  5. Routing rule页面
     1. 加载routing rule页面时，如果salesforce integration enable，则通过接口GetSalesforceFields()同步获取salesforce字段
     2. salesforce字段存到数据库中ConditionVariable
        1. Account：\_\_Salesforce\_Account\_fieldName格式
        2. Contact：\_\_Salesforce\_Contact\_fieldName格式
        3. FieldName为这个salesforce字段的系统字段
     3. 显示用salesforce字段的label

1. Chat server
   1. 增加SalesforceIntegrationCache，用来存储Salesforce配置信息的缓存，和CacheSettings等实现方式类似；
   2. Identify Contact
      1. 时机：和原来相同，在ChatEngineWrap的RequestChat，ManualInvitationAccepte。根据用户配置逻辑来定位SalesforceContact，此动作仍然按原来的方式异步进行；
         1. 即在用户提交PreChat后，服务器起线程去调用Salesforce的服务；
         2. 此时将Chat状态置为EnumSystemProcessing，访客状态为EnumSystemProcessing; 同时增加一个15s的超时时钟；
         3. 在此状态时，访客仍然停留在PreChat提交界面，客服端展示为这个状态展示一个新图标，但是此访客不会展示My Chats；
         4. Salesforce调用成功返回时，继续RequestChat的动作，进行路由分配，更改状态为WaitingforChat。
         5. 如果超时时间到，则忽略Salesforce的数据返回，继续RequestChat的动作，进行路由分配，更改状态为WaitingforChat。
      2. IntegrationThreadHelper增加函数：IdentitySalesforceContact。根据用户配置的Fieldmapping规则，调用SalesforceIntegration的接口，获取ContactUrl，Contact和Account字段信息，设置到CurrentVisitor对象中；
      3. 默认FieldMapping情况下，根据Email查找Contact，同时根据此Contact关联的Account得到Account信息。在设置自定义规则的情况下，根据自定义的规则定位Contact，然后再根据自定义规则定位Account。
      4. 和原来逻辑不通的是不再对Lead进行自动识别。
   3. Contact和Account字段存储与展示
      1. Current Visitor增加字段 bool IfHasContact, List<StruIntegrationFieldAndValue> SalesforceContactFields和bool IfHasAccount, List<StruIntegrationFieldAndValue> SalesforceAccountFields。在对Identity Contact赋值，会将两个对象所有的值设置到相应的值列表中。
      2. Visitor，VisitorWithMoreDiff同时增加如下两个字段，存储VM中展示的信息。同时增加差量计算相关逻辑

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Abbrev** |
| SalesforceContactFields | List<StruIntegrationFieldAndValue> | cb |
| SalesforceAccountFields | List<StruIntegrationFieldAndValue> | cc |
| IfHasContact | bool | cd |
| IfHasAccount | bool | ce |

* 1. Routing rule计算
     1. 在原先Routing Rule的计算逻辑中的请求聊天之前的时间点进行
     2. 路由计算函数中增加Salesforce相关字段比较逻辑；，加入Salesforce字段判断逻辑。匹配条件里名字以\_\_Salesforce开头的字段，和自定义字段类似。
  2. 创建Salesforce相关对象Lead, Case, Contact&Case
     1. 增加聊天action，EnumCreateSalesforceObject。参数包括

|  |  |
| --- | --- |
| **Enum** |  |
| EnumSalesforceCreateLead |  |
| EnumSalesforceCreateCase |  |
| EnumSalesforceCreateContactAndAttachCase | 分3种情况，  message字段为枚举表示：  1：DoNotUpdateContact  2：UpdateContact  3：OnlyAddNewInformation |

* + 1. 访客监控请求新建Salesforce对象时，直接异步调用SalesforceIntegration的接口新建Salesforce对象。如果此时处于聊天状态，则当聊天结束时，再将聊天结束时的信息对新建对象进行更新。
    2. 聊天结束和留言的自动生成Salesforce对象的逻辑和原来相同。