TRS Web Application Server

TRS WAS V5.0 与 IDS 集成手册

北京拓尔思信息技术股份有限公司

Beijing TRS Information Technology Co., Ltd.

版权说明

关 于 本 手 册

本手册详细介绍了 TRS WAS 5.0 与 IDS 的集成过程。在按照本文档进行集成之前，假设 TRS WAS 5.0 和 IDS 均已安装成功。

本手册适用于 TRS WAS 5.0 与 TRS IDS 的集成。

读 者 对 象

本手册的读者为企业身份管理服务的使用者和企业身份管理集成开发人员。

用 户 反 馈

TRS 公司感谢您使用 TRS 产品。如果您发现本手册中有错误或者产品运行不正确，或者您对本手册有任何意见和建议，请及时与 TRS 公司联系。您的意见将是我们做版本修订时的重要依据。

目 录

第1章 集成步骤

1.1 在 IDS 上注册协作应用

出于安全的考虑，只有在 TRS IDS 上注册了的协作应用才能够使用 TRS IDS 的服务。

在 IDS 管理台中点击“应用”标签中的“新建”打开新建页面，如下图所示：

在添加新协作应用时根地址/请求 URL/主页设置为：

http://WAS5.0 所在 ip:WAS5.0 端口/was5

应用内部授权 URL 设置为：

http://WAS5.0 所在 ip:WAS5.0 端口/was5/service/Authorization

在以上添加页面中填入相应的信息即可。如果对填写的项有疑问，请查阅《TRSIDS3.5用户手册》添加协作应用。

1.2 配置 trsids-agent.properties 文件

该文件位于 WAS 安装目录/Tomcat/webapps/was5/WEB-INF/classes 目录下。

下面给出 WAS5.0 上部署的 Agent 的配置文件作为示例，标红的地方需要根据实际 IDS 信息来进行设置：

```
# TRS IDS Servlet-Agent Application Config
# TRS身份服务器Servlet应用代理配置文件
#
# Copyright(c): www.trs.com.cn
# Last Modified: 2009-06-11

## 注意事项：
#  1、配置项格式：名称=值
#  2、#开始的行表示注释
##

############################## 基础配置项信息开始
########################################

## 连接属性
#所要连接IDS的IP(或主机). 必填.
idm.server.host = 192.9.200.76
#所要连接IDS的后台SSLServer端口.必填.如果不填默认为2005.
idm.server.port = 2005
#创建socket数
idm.sockets.amount = 5

#socket超时
#soTimeout = 30000
# 从空闲Socket队列中取出一个Socket的最大间隔(毫秒, ms), 有效值3000-30000之间, 默认为15000
#socket.dispatchWaitMillis=15000

#使用IBM JDK时在JSSE可能会遇到问题, 此时可采用如下配置(如果使用"socketType=plain"则必须在IDS端也配置"socketType=plain")
socketType=plain
#socketType=dummySSL

## 协作应用属性
#协作应用名. 必填, 并且必须和在所要连接IDS上注册的协作应用名保持一致.
agent.name = WAS5
```

```
#是否允许匿名访问, y为允许, 否则为不允许.
allow.anonymous = n

#[名称] coAppActor.className
#[作用] 协作应用实现的IDS回调接口的类名称(包括包名的类名),
#[要求] 必填，根据实际情况填写。
#[示例] coAppActor.className = com.trs.idm.client.DemoActor
#[特例] 如果第三方协作应用采用提供page页面的集成方式，则actor为:
#         coAppActor.className = com.trs.idm.client.DemoActor
#[参考] TRS BBS实现的IDS回调接口的类名称:
#         coAppActor.className = com.trs.idm.coappsupport.bbs.BBSActor
#       TRS CIS实现的IDS回调接口的类名称:
#         coAppActor.className = com.trs.idm.coappsupport.cis.CISActor
#       TRS CDS(Demo应用)实现的IDS回调接口的类名称
#         coAppActor.className = com.trs.cds.ids.app.CDS4IDSWebCoAppActor
#       TRS CDS(管理台)实现的IDS回调接口的类名称
#         coAppActor.className =
com.trs.cds.ids.admin.CDSAdmin4IDSWebCoAppActor
#       TRS WCM v5.1实现的IDS回调接口的类名称
#         coAppActor.className = com.trs.idm.coappsupport.wcm.WCM51Actor
#       TRS WCM v5.2实现的IDS回调接口的类名称
#         coAppActor.className = com.trs.presentation.ids.WCM52Actor
coAppActor.className = com.trs.idm.client.actor.WASActor


#########################################################################
# 该应用web.xml中TRSIdSSSOFilter截获的URL中, 对哪些URL进行SSO控制.
# 有两种配置方式, 对应两个配置项: ignoreUrl.prefix和processUrl.prefix.
#方式1: 不进行SSO控制的链接.
#ignoreUrl.prefix =
#方式2: 进行SSO控制的链接.
#processUrl.prefix =
#
# 相关说明:
#1. 两个配置项均采用前缀匹配方式(即"/admin"相当于"/admin*", 匹配所有以"/admin"
开始的URL).
#2. 两个配置项均可以配置多个值, 多个值间用","来分隔. 比如"/blogger/,/admin/"
#3. 两个配置项如果都不配置, 则对所有被TRSIdSSSOFilter截获的URL都进行SSO.
#4. 建议两项中只配置一项, 避免把问题复杂化. 除非有非常特殊的需要必须同时配置两项.
#5. 如果两个配置项都配置了, 则对于重合部分的URL匹配, processUrl.prefix的优先级更
高.
#6. 更多疑问请参考产品文档的说明.
#
# 一些配置示例:
#适用于TRS WCM v5.1:
#ignoreUrl.prefix =
/appManager.jsp,/doc/soap,/doc/batch_unzip.jsp,/wcmregister.jsp
```

```
#适用于TRS CIS v2.5:
#ignoreUrl.prefix = /admin/
#适用于TRS Blog(BSP):
#processUrl.prefix = /blogger/,/admin/

#########################################################################

#[名称] loginAction.uri
#[作用] 协作应用原有登录页面上的form action链接.
#[要求] 必填。
#       具体配置规范如下：
#       1. 从应用上下文开始（不包括上下文）,采用后缀匹配方式进行判断.
#       2. 该配置项支持多个配置，中间使用逗号","分割，如：loginAction.uri =
/do_login.jsp,/login.do,/do_login.jsp?mod=dologin.
#       3. 该配置支持携带参数的形式，如：loginAction.uri =
/do_login.jsp?mod=dologin.
#[示例] 比如用户登录POST的URI为/do_login.jsp,
#       如： <form action="/do_login.jsp" method="post"></form>
#       那么loginAction.uri参数需要配置为/do_login.jsp，表示从上下文开始（不包括
上下文），匹配所有以/do_login.jsp开头的URL.
#[参考] TRS CDS的自有登录页面的form action链接:   loginAction.uri = /login
#       TRS CIS的自有登录页面的form action链接:  loginAction.uri = /login.jsp
#       TRS BBS的自有登录页面的form action链接:  loginAction.uri = /UserLogin
#       TRS WCMv6的自有登录页面的form action链接:  loginAction.uri =
/login_dowith.jsp
loginAction.uri = /admin/dologin.jsp,/web/dologin.jsp

#[名称] loginAction.method
#[作用] 协作应用原有登录的提交方式，分为GET和POST(不区分大小写).
#     该参数主要是针对同一登录请求URL,根据提交方式判断是否为登录请求的情况。
#[要求] 可以不填
#[示例] loginAction.method = GET
#[参考] 假设原应用中，以POST方式访问index.jsp为登录请求，那么需要这样配置：
#      loginAction.uri = /index.jsp
#      loginAction.method = POST
#      此时，以GET方式访问index.jsp时，Filter将作为普通请求处理，不会作为登录的请
求处理。
#loginAction.method = GET

#[名称] logout.uri
#[作用] 协作应用原退出页面的的操作链接。用于统一注销,如果填写错误则会影响单点注销功能。
#[要求] 具体配置规范如下：
#     1. 采用后缀匹配方式进行判断，uri从应用上下文开始（不包括上下文，即不包括应用的
contextPath，如果应用调整了contextPath，这个配置项的值并不需要改变）。
#     2. 该配置项支持多个配置，中间使用逗号","分割，如：logout.uri =
/logout.jsp,logout.do
#     3. 该配置支持携带参数的形式，如：logout.uri = /logout.jsp?mod=dologout
```

```
#[示例] 假设原应用中，用户请求的注销URL为
http://192.9.200.11/demo1/user/logout.jsp
#       那么logout.uri参数必须配置为 /user/logout.jsp，表示从上下文开始（不包括上下文），匹配所有以/user/logout.jsp开头的URL.
#[参考] TRS WCM, TRS CIS, TRS CDS的注销操作链接均为：logout.uri = /logout.jsp
logout.uri = /admin/logout.jsp,/web/logout.jsp

#[名称] afterLoginOk.gotoUrl
#[作用] 在协作应用自有登录页面登录成功后跳转的页面
#[要求] 可以不填，默认登录后仍返回原页面。
#[示例] afterLoginOk.gotoUrl = main.jsp
#afterLoginOk.gotoUrl = main.jsp

#[名称] afterLoginFail.gotoUrl
#[作用] 在协作应用自有登录页面登录失败后跳转的页面
#[要求] 可以不填，默认登录后仍返回原页面。
#[示例] afterLoginFail.gotoUrl = error.jsp
#afterLoginFail.gotoUrl = error.jsp


########################################    用户在IDS不存在时的控制逻辑开始
#########################################################

#[名称] sso.ifUserNotExistOnIDS.useLocalLoginLogic
#[作用] 当用户在IDS上不存在时，是否尝试将用户名密码提交到应用自身执行登录的页面上进行登录
#[要求] 可以不填，默认是false。
#[示例] true 或者 false
sso.ifUserNotExistOnIDS.useLocalLoginLogic=true

#[名称] sso.ifUserNotExistOnIDS.selfLoginPage.action.url
#[作用] 当用户在IDS上不存在时，将用户名密码提交到应用自身执行登录的页面
#[要求] 可以不填，默认是获取loginAction.uri第一个配置项。
#[示例] 比如应用登录POST的URI为/do_login.jsp,
#       如：  <form action="/do_login.jsp" method="post"></form>
#       那么sso.ifUserNotExistOnIDS.selfLoginPage.action.url参数需要配置为/do_login.jsp，表示当用户在IDS上不存在时，将用户名密码提交到该页面进行登录。
sso.ifUserNotExistOnIDS.selfLoginPage.action.url = /login

#[名称] sso.selfLoginPage.userName.field
#[作用] 应用原来的登录页面上，用户名字段的名称。
#           如果希望用户在IDS上不存在时，仍能将用户名提交到应用自身执行登录的页面上，必须配置此项。
#[要求] 此配置项的值必须和应用登录表单上，用户名字段的值一致。
#[示例] 假设 应用原有登录页面上，有如下形式的表单：
#       <form action="do_login.jsp" method=post>
#               <input name="coAppUserName">
```

```
#              <input type=password name="coAppPassword" >
#       </form>
#       则此配置项应当被配置为      coAppUserName
sso.selfLoginPage.userName.field=username

#[名称] sso.selfLoginPage.password.field
#[作用] 应用原来的登录页面上，密码字段的名称。
#          如果希望用户在IDS上不存在时，仍能将用户名提交到应用自身执行登录的页面上，必须配置此项。
#[要求] 此配置项的值必须和应用登录表单上，密码字段的值一致。
#[示例] 假设 应用原有登录页面上，有如下形式的表单：
#       <form action="do_login.jsp" method=post>
#              <input name="coAppUserName">
#              <input type=password name="coAppPassword" >
#       </form>
#       则此配置项应当被配置为      coAppPassword
sso.selfLoginPage.password.field=password

########################################    用户在IDS不存在时的控制逻辑结束
#########################################################


#[名称] cookie.domain.level
#[作用] 应用匿名访问cookie标记的tld级别。
#[要求] 默认不需要做设置，在特殊情况才做设置。不填则cookie使用默认的domain。
#[示例] 常见场景为访问同一个应用，却可能存在不同的域名，比如，访问博客系统，却可能存在以下的访问url：
#          http://user1.blog.com, http://user2.blog.com
#       如果不做tld设置的话，会导致每次访问都需要去身份服务器做验证，而这个是不必要的。
cookie.domain.level=


#[名称] maxAllowAsyncSessionIdNumber
#[作用] 同步通知中允许的链表最大数量
#[要求] 可以不填。不填默认是100000。
maxAllowAsyncSessionIdNumber=100000

#############################  PageActor配置开始
##########################################
# 当协作应用无法提供Java开发的Actor类时，可以通过提供相应的Page页面完成与身份服务器的集成，
# 下面所有页面都需要重新定制一份，保持与现有系统逻辑页面独立，请不要使用系统自身已存在的页面
#########################################################################
###################

#[名称] pageActor.sessionLoginFlag
#[作用] 协作应用用户登录后的session标识
```

```
#[要求] 默认为"trs_coapp_loginUser"，当用户未登录时，则session中不存在该标识或该标识对应的value为空
pageActor.sessionLoginFlag =

#[名称] pageActor.handleLoginPage
#[作用] 接受用户登录请求并完成登录逻辑的页面
#[要求] 从应用的上下文（request.getContextPath()）开始，默认为"/pageactor/handleLoginPage.jsp"。
#       注意：不要与loginAction.uri重复!
pageActor.handleLoginPage =

#[名称] pageActor.handleLogoutPage
#[作用] 接受用户退出请求并完成退出逻辑的页面
#[要求] 从应用的上下文（request.getContextPath()）开始，默认为"/pageactor/handleLogoutPage.jsp"。
#       注意：不要与logout.uri重复!
pageActor.handleLogoutPage =

#[名称] pageActor.handleAddUserPage
#[作用] 完成用户添加逻辑的页面
#[要求] 从应用的上下文（request.getContextPath()）开始，默认为"/pageactor/handleAddUserPage.jsp"
pageActor.handleAddUserPage =

#[名称] pageActor.handleDelUserPage
#[作用] 完成用户删除逻辑的页面
#[要求] 从应用的上下文（request.getContextPath()）开始，默认为"/pageactor/handleDelUserPage.jsp"
pageActor.handleDelUserPage =

#[名称] pageActor.handleUpdateUserPage
#[作用] 完成用户更新逻辑的页面
#[要求] 从应用的上下文（request.getContextPath()）开始，默认为"/pageactor/handleUpdateUserPage.jsp"
pageActor.handleUpdateUserPage =

#[名称] sso.selfLoginPage.userName.field、sso.selfLoginPage.password.field
#[作用] 于匿名应用而言，如果使用使用自己的登录表单，则需要提供表单用户名和密码元素的名称
#[要求] 用户名表单元素名称默认为 "userName"，密码表单元素名称默认为"password"。
#       注意：不要重复配置这两项。
#sso.selfLoginPage.userName.field=userName
#sso.selfLoginPage.password.field=password

##############################  PageActor配置结束
##########################################
```

```
############################## 新的 SSO集成方式 配置开始
##########################################

#//////////////////////////  基本的配置项
//////////////////////////////

#[名称] sso.basic.type
#[作用] 要使用的SSO集成方式的类型。simpleToken或者saml2。
#          simpleToken：TRSIDS自己实现的集成方式
#          saml2：支持SAML2标准的集成方式
#[要求] 必填。填写simpleToken或者saml2，如果不填写或者填写错误，默认为simpleToken。
sso.basic.type = simpleToken

#[名称] sso.basic.ids.url
#[作用] IDS的地址
#[要求] 必填。
#       注意！！！
#       此项必需填写确实能访问到的IDS地址，否则集成完成以后，访问应用，会报404错误
#[示例] http://192.9.200.72:8080/ids
sso.basic.ids.url = http://192.9.200.76:8080/ids

#[名称] sso.basic.resource.protected
#[作用] 应用中受保护的资源列表，受保护资源列表中的uri需要登录才能访问。
#          “受保护资源”可以是一个文件夹，也可以是一个页面。
#[要求] 必填。
#       支持多项配置，多个值请以 分号 隔开。
#       如果想要将应用所有的页面都定义为受保护资源，声明为 "*.*"即可。（具体请参见示
例）
#[示例] 假设应用的结构如下：
#          应用根
#             |- admin文件夹
#             |- image文件夹
#             |- index.jsp页面
#             |- protectA.jsp页面
#             |- protectB.jsp页面
#       如果想要指定访问 /admin, protectA.jsp, protectB.jsp时，都需要先做登录，
配置项应该如下：
#
sso.basic.resource.protected=/admin;/protectA.jsp;/protectB.jsp
#       如果希望访问应用所有的地址都先做登录，则配置项如下：
#          sso.basic.resource.protected=*.*
sso.basic.resource.protected=*.*

#[名称] sso.basic.page.autoSubmit
#[作用] agent所使用的自动提交页面的地址。
#[要求] 选填。
```

```
#       自动提交页面在部署完成以后，默认是在 应用根的idsSPPages文件夹下。
#       如果没有移动过idsSPPages文件夹，那么本项参数不需要配置。
#       默认值为： /idsSPPages/sp.jsp
#[示例] sso.basic.page.autoSubmit=/idsSPPages/sp.jsp
sso.basic.page.autoSubmit=/idsSPPages/sp.jsp



#////////////////////////// SSO API集成时的配置项
//////////////////////////////

#[名称] sso.simpleToken.sercurityMode
#[作用] simpleToken集成方式中，是否使用安全模式进行集成。
#       "安全模式"可以最大限度的避免token被伪造，但同时也会增加通讯过程的负担，对SSO
的效率有一定的影响。
#       一般情况下不建议采用安全模式。
#[要求] 选填。
#       true 或者 false。不填默认为false。
sso.simpleToken.sercurityMode=false

#[引用] 使用自有登录页时，执行登录的页面
#           具体说明参见  loginAction.uri配置项

#[引用] 使用自有登录页时的用户名字段名
#           具体说明参见sso.selfLoginPage.userName.field配置项

#[引用] 使用自有登录页时的密码字段
#           具体说明参见sso.selfLoginPage.password.field配置项

#[引用] 使用自有登录页面时，登录成功以后跳转到的页面地址。可以不填，如果不填写，直接跳
转到执行登录页面的前一个地址。
#           具体说明参见afterLoginOk.gotoUrl配置项

#[引用] 使用自有登录页面时，登录失败后跳转到的页面，可以不填写，如果不填写，直接跳转到
执行登录页面的前一个地址。
#           具体说明参见afterLoginFail.gotoUrl配置项

#使用binding的方式,可以不填，不填或者填错默认是HTTPPOST
#可填写的值有httpPost，redirect
sp.simple.sso.bindings.type=httpPost

#////////////////////////// 需要支持SAML2时的配置项
//////////////////////////////

#[名称] sso.saml2.config.issuer
#[作用] SAML2标准中，关于提交者的信息。
#[要求] 选填。一般填写应用的实际地址。
```

```
#[示例] sso.saml2.config.issuer=http://www.xinhuanet.com/forum
sso.saml2.config.issuer =

#[名称] sso.saml2.config.bindings.type
#[作用] SAML2标准中，使用binding的方式。
#       供选择的值有：
#           HTTPPost：Post的binding方式，断言通过Post的方式进行传递。
#           HTTPRedirect：断言已Http URL参数的方式传递，可能会导致URL超长，不建议
使用。
#           SOAP11：断言通过SOAP 1.1标准的方式进行传递
#           HTTPClient：非SAML2标准的binding方式，仅供IDS内部集成应用时使用，如果
有第三方SAML应用，不建议使用。
#[要求] 选填。可以不填，如果不填写或者填写错误，默认为HTTPPOST。
#[示例] sso.saml2.config.bindings.type = HTTPPost
sso.saml2.config.bindings.type =



############################## 新的 SSO集成方式 配置结束
##########################################


#END:配置文件结束
```

其中各配置项的详细说明请参见《TRS IDS3.5 协作应用集成手册》.

1.3 配置 web.xml 文件

备份 WAS 安装目录/Tomcat/webapps/was5/WEB-INF/web.xml➔was_web.xml

重命名同级目录下的 ids_web.xml➔web.xml

其中 ids_web.xml 中已经设置 IDS 集成所需的参数。只需要重命名为 web.xml 后重启 WAS 应用就可以成功集成。

1.4 Weblogic/Websphere 集成

在这两个应用服务器下集成 IDS 也只需要参考 1.3 节配置将应用部署到应用服务器后相应路径下的 web.xml 文件即可。

- **Weblogic**

  将部署到 Weblogic 的应用目录下的 web.xml 根据 1。3 节的方法进行配置，weblogic 的部署目录为安装目录/WASData/deploy/weblogic

- **Websphere**

  将安装目录/WASData/deploy/websphere 的 ear 包部署到 websphere 后，配置下面参考路径下的 web.xml：

  /opt/IBM/WebSphere7.0/AppServer/profiles/AppSrv01/config/cells/localhostNode01Cell/applications/was5.ear/deployments/was5/was5.war/WEB-INF