Artículo
Mathew Lambert · 7 jul, 2020 Lectura de 6 min

Ejecución remota de código para InterSystems Caché

Introducción
Si gestionas múltiples instancias de Caché en varios servidores, puede que quieras ejecutar código arbitrario de una instancia de Caché en otra. Los administradores de sistemas y especialistas de soporte técnico también podrían querer ejecutar un código arbitrario en servidores Caché remotos. Para satisfacer estas necesidades, he desarrollado una herramienta especial llamada RCE.

En este artículo, analizaremos cuáles son las formas más comunes de resolver tareas similares y cómo RCE (Remote Code Execution) puede ser útil.
¿Cuáles son los enfoques posibles?

Ejecutar comandos del SO localmente

Empecemos por lo más simple – Ejecutar comandos del SO localmente desde Caché. Para hacerlo, puedes ejecutar el comando $zf:

- $ZF(-1) invocará un programa o comando del sistema operativo. Se realiza una llamada desde un nuevo proceso, y el proceso padre espera hasta que el proceso hijo finaliza. Una vez ejecutado el proceso, $ZF(-1) devuelve el código resultante del proceso hijo: 0 si se ejecutó con éxito, 1 si se ejecutó con errores o -1 si el sistema no fue capaz de crear el proceso hijo. Se ve así:
  set status = $ZF(-1,"mkdir ""test folder""")
- $ZF(-2) es un comando similar, excepto que el proceso padre no espera a que el proceso hijo finaliza. El comando devuelve 0 si el proceso se creó con éxito o -1 si el sistema no logró crear el proceso hijo.

También hay métodos de la clase %Net.Remote.Utility (solo para uso de InterSystems) que ofrece "wrappers" prácticos para funciones estándar y muestra los resultados de procesos invocados de una forma más intuitiva:

- RunCommandViaCPIPE ejecuta un comando usando Command Pipe. Devuelve el dispositivo creado y la salida del proceso.
- RunCommandViaZF ejecuta un comando usando $ZF(-1). Escribe la salida del proceso en un archivo y devuelve la salida del proceso.

Una opción alternativa es usar el comando de terminal ! (o $, que es lo mismo) que abre el intérprete estándar del sistema operativo directamente dentro del terminal de Caché. Hay dos modos de funcionamiento disponibles:

- Modo de una línea – todo el comando se pasa con ! y el intérprete de comandos lo ejecuta inmediatamente, mientras que la salida se envía al dispositivo de Caché actual. El ejemplo anterior se ve así:

```
SAMPLES>! mkdir ""test folder""
```

- Modo multilínea – el sistema ejecuta ! primero y luego abre el intérprete, donde puedes introducir los comandos necesarios del sistema operativo. Para cerrar el intérprete, escribe "quit" o "exit", dependiendo del intérprete en el que estás trabajando:

```
SAMPLES>!
C:\InterSystems\Cache\mgr\samples\> mkdir "test folder"
```

```
C:\InterSystems\Cache\mgr\samples\> quit
SAMPLES>
```

Ejecución remota de código ObjectScript de Caché

La ejecución remota es posible a través de la clase %Net.RemoteConnection (discontinuada) que ofrece la siguiente funcionalidad:

- Abrir y modificar objetos almacenados;
- Ejecutar métodos y objetos de clase;
- Ejecutar consultas.

Código de prueba para demostrar estas funcionalidades

```
Set rc=##class(%Net.RemoteConnection).%New()
Set Status=rc.Connect("127.0.0.1","SAMPLES",1972,"_system","SYS") break:'Status
Set Status=rc.OpenObjectId("Sample.Person",1,.per) break:'Status
Set Status=rc.GetProperty(per,"Name",.value) break:'Status
Write value
Set Status=rc.ResetArguments() break:'Status
Set Status=rc.SetProperty(per,"Name","Jones,Tom "_$r(100),4) break:'Status
Set Status=rc.ResetArguments() break:'Status
Set Status=rc.GetProperty(per,"Name",.value) break:'Status
Write value
Set Status=rc.ResetArguments() break:'Status
Set Status=rc.AddArgument(150,0) break:'Status  // Addition 150+10
Set Status=rc.AddArgument(10,0) break:'Status  // Addition 150+10
Set Status=rc.InvokeInstanceMethod(per, "Addition", .AdditionValue, 1) break:'Status
Write AdditionValue
Set Status=rc.ResetArguments() break:'Status
Set Status=rc.InstantiateQuery(.rs,"Sample.Person","ByName")
```

Este código realiza varias acciones:

- Conecta con el servidor de Caché
- Abre la instancia de clase Sample.Person con ID=1
- Obtiene el valor del atributo
- Modifica el valor del atributo
- Define argumentos para el método
- Llama al método de la instancia
- Ejecuta la consulta Sample.Person:ByName

Para operar del lado del servidor, %Net.RemoteConnection necesita tener instalado C++ binding.

También merece la pena mencionar la tecnología ECP. Esta tecnología permite ejecutar los procesos JOB remotamente en un servidor de base de datos desde el servidor de tu aplicación.

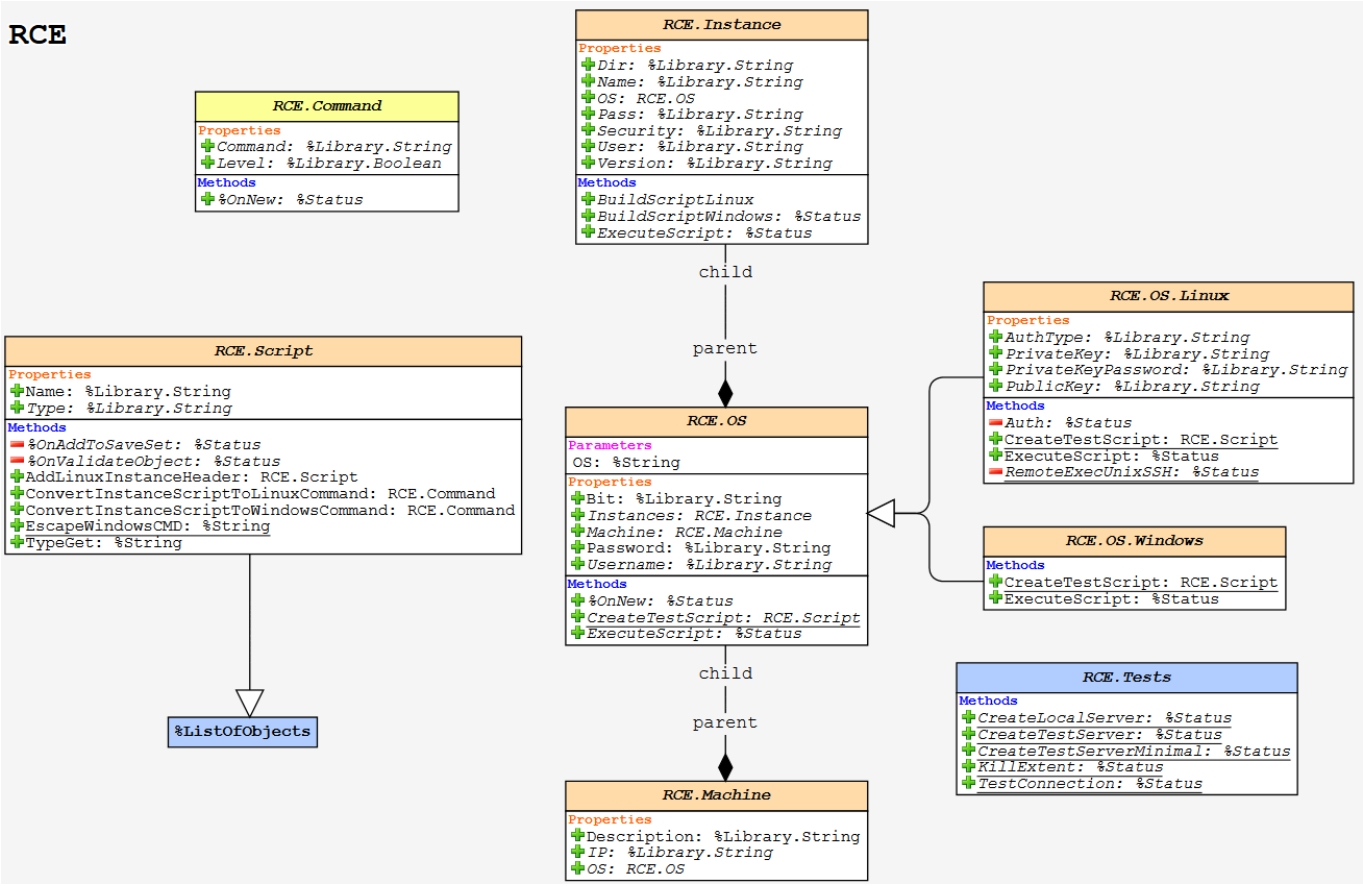
En general, la combinación de estos dos enfoques puede resolver nuestra tarea eficientemente, pero los usuarios aún necesitan un flujo de trabajo simple para crear scripts por lotes, ya que estos enfoques pueden ser bastante complejos de entender e implementar.

RCE

Por lo tanto, los objetivos de estos proyectos fueron los siguientes:

- Ejecución de scripts en servidores remotos desde Caché;
- No tener necesidad de configurar servidores remotos (del lado del cliente);
- Reducir al mínimo las configuraciones en servidores locales (del lado del servidor);
- Alternar de forma transparente entre comandos del sistema operativo y ObjectScript de Caché;
- Soportar clientes Windows y Linux.

La jerarquía de clases del proyecto se ve así:



La jerarquía "Máquina – SO – Instancia" almacena la información requerida para acceder a los servidores remotos. Todos los comandos se almacenan en la clase RCE.Script, que contiene la lista ordenada de objetos de clase RCE.Command, que sirven ya sea como comandos de SO o código ObjectScript de Caché.

Ejemplos de comandos:

```
Set ?ommand1 = ##class(RCE.Command).%New("cd 1", 0)
Set ?ommand2 = ##class(RCE.Command).%New("zn ""%SYS""", 1)
```

El primer argumento es el texto del comando, el segundo argumento es el nivel de ejecución: 0 – SO, 1 – Cache.

Ejemplo de creación de un nuevo script:

```
Set Script = ##class(RCE.Script).%New()
Do Script.Insert(##class(RCE.Command).%New("touch 123", 0))
Do Script.Insert(##class(RCE.Command).%New("set ^test=1", 1))
Do Script.Insert(##class(RCE.Command).%New("set ^test(1)=2", 1))
Do Script.Insert(##class(RCE.Command).%New("touch 1234", 0))
Do Script.%Save()
```

En este ejemplo, el sistema ejecutará el 1er y el 4º comando a nivel del SO y el 2º y el 3er comando a nivel de Caché. Alternar entre estos dos niveles es totalmente transparente para los usuarios.

Mecanismos de ejecución

Actualmente, se soportan las siguientes rutas de ejecución:

Servidor	Cliente
Linux	Linux, Windows (el servidor SSH de
Windows	Linux, Windows (deberías instalar u lado del servidor)

Si se soporta ssh del lado del cliente, el servidor generará el comando ssh y lo ejecutará del lado del cliente usando la clase estándar %Net.SSH.Session.
Si tanto el servidor como el cliente operan bajo Windows, el sistema generará un archivo BAT y lo ejecutará del lado del cliente usando psexec.

Agregar un servidor

Carga clases desde el repositorio hacia cualquier namespace. Si tu servidor funciona en Windows y quieres gestionar otros servidores Windows, entonces asigna al global ^settings("exec") una ruta hacia psexec. ¡Y eso es todo!

Añadir un cliente

Añadir un cliente es básicamente guardar todos los datos necesarios para hacer la autenticación.

Ejemplo del código de programa que crea una nueva jerarquía "PC – SO – Instancia"

```
Set Machine = ##class(RCE.Machine).%New()
Set Machine.IP = "IP or Host"
Set OS = ##class(RCE.OS).%New("O?") // Linux or Windows
Set OS.Username = "Operation system user"
Set OS.Password = "User password"
Set Instance = ##class(RCE.Instance).%New()
Set Instance.Name = "Caché instance name"
Set Instance.User = "Caché user name" // Unrequired on minimal security settings
Set Instance.Pass = "Caché user password" // Unrequired on minimal security settings
Set Instance.Dir = "Path to cterm"  // Required only on Windows clients, who don't ha
ve cterm in PATH
Set Instance.OS = OS
Set OS.Machine = Machine
Write $System.Status.GetErrorText(Machine.%Save())
```

Ejecución de scripts

Y, por último, ejecutamos nuestros scripts. Es muy fácil – solo necesitamos ejecutar el método ExecuteScript desde la clase RCE.Instance a la que se transfieren el objeto del script y el namespace (%SYS por defecto):

```
Set Status = Instance.ExecuteScript(Script, "USER")
```

Resumen

RCE ofrece un mecanismo práctico para ejecutar código de forma remota para InterSystems Caché. Como la herramienta solo usa scripts almacenados, deberás escribir cada uno de ellos solo una vez y luego ejecutarlos

donde quieras, sobre cualquier cantidad de clientes.

Referencias

Repositorio de RCE en GitHub
Archivo de clases del proyecto RCE

#Administración del sistema #Callout #Herramientas #Terminal #Caché