[](https://www.sqlserverscience.com/)

Technical Articles for the DBA / Developer

Allowing users to start and stop SQL Server Agent Jobs

FROM: <https://www.sqlserverscience.com/tools/allowing-users-to-start-and-stop-sql-server-agent-jobs/>

2016-04-28 · by [Max Vernon](https://www.sqlserverscience.com/author/admin/) · in [tools](https://www.sqlserverscience.com/tools/)

I recently received a request to allow certain users to start and stop SQL Server Agent jobs, without giving them the ability to use the SQL Server Agent tools in SQL Server Management Studio.

I created a Visual Studio .Net application that provides that ability through stored procedures that use the WITH EXECUTE AS OWNER directive.

The T-SQL code for the SQL Server prerequisites for this app are as follows:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433 | /\*  Script to create objects required for the Job Control App  By: Max Vernon  Date: 2016-04-27  Notes: Compatible with SQL Server 2005+ (not tested on 2014+)  \*/  USE msdb;  GO  IF OBJECT\_ID('dbo.sysjobs\_log') IS NOT NULL  DROP TABLE dbo.sysjobs\_log;  GO  IF OBJECT\_ID('dbo.sysjobs\_action\_types') IS NOT NULL  DROP TABLE dbo.sysjobs\_action\_types;  GO  CREATE TABLE dbo.sysjobs\_action\_types  (  ActionTypeID INT NOT NULL  CONSTRAINT PK\_sysjobs\_action\_types  PRIMARY KEY  CLUSTERED  , ActionDescription VARCHAR(100) NOT NULL  ) ON [PRIMARY];  INSERT INTO dbo.sysjobs\_action\_types (ActionTypeID, ActionDescription)  VALUES (1, 'Started Job')  INSERT INTO dbo.sysjobs\_action\_types (ActionTypeID, ActionDescription)  VALUES (2, 'Stopped Job');  GO  CREATE TABLE dbo.sysjobs\_log  (  sysjobs\_log\_ID INT NOT NULL  CONSTRAINT PK\_sysjobs\_log  PRIMARY KEY  CLUSTERED  IDENTITY(1,1)  , job\_id UNIQUEIDENTIFIER NOT NULL  CONSTRAINT FK\_sysjobs\_log\_\_job\_id  FOREIGN KEY  REFERENCES dbo.sysjobs(job\_id)  ON DELETE CASCADE  , JobActionDateTime DATETIME NOT NULL  CONSTRAINT DF\_sysjobs\_log\_JobActionDateTime  DEFAULT (GETDATE())  , ActionBy SYSNAME NOT NULL  CONSTRAINT DF\_sysjobs\_log\_StartedBy  DEFAULT (ORIGINAL\_LOGIN())  , ActionAs SYSNAME NOT NULL  CONSTRAINT DF\_sysjobs\_log\_StartedAs  DEFAULT (SUSER\_SNAME())  , ActionTypeID INT NOT NULL  CONSTRAINT FK\_sysjobs\_log\_\_ActionType  FOREIGN KEY  REFERENCES dbo.sysjobs\_action\_types(ActionTypeID)  ) ON [PRIMARY];  GO  IF OBJECT\_ID('dbo.sysjobs\_filter') IS NOT NULL  DROP TABLE dbo.sysjobs\_filter;  GO  CREATE TABLE dbo.sysjobs\_filter  (  job\_id UNIQUEIDENTIFIER NOT NULL  CONSTRAINT FK\_sysjobs\_filter\_\_job\_id  FOREIGN KEY  REFERENCES dbo.sysjobs(job\_id)  ON DELETE CASCADE  ) ON [PRIMARY];  CREATE INDEX PK\_sysjobs\_filter  ON dbo.sysjobs\_filter(job\_id);  GO  IF OBJECT\_ID('dbo.cp\_s\_job\_list') IS NOT NULL  DROP PROCEDURE dbo.cp\_s\_job\_list;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Shows a list of jobs that are available for non-DBAs to run  using the dbo.cp\_s\_start\_job interface    By: Max Vernon  Date: 2016-04-25    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_s\_job\_list  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  SELECT JobID = sj.job\_id  , JobName = sj.name  , JobDescription = sj.description  , IsEnabled = CASE WHEN sj.enabled = 1 THEN CONVERT(BIT, 1) ELSE CONVERT(BIT, 0) END  , LastRunDate = (SELECT TOP(1) msdb.dbo.agent\_datetime(sjh.run\_date, sjh.run\_time)  FROM msdb.dbo.sysjobhistory sjh  WHERE sjh.job\_id = sj.job\_id  ORDER BY sjh.run\_date DESC, sjh.run\_time DESC)  FROM msdb.dbo.sysjobs sj  INNER JOIN msdb.dbo.sysjobs\_filter sjf ON sj.job\_id = sjf.job\_id  ORDER BY sj.name;  END  GO  IF OBJECT\_ID('dbo.cp\_s\_job\_status') IS NOT NULL  DROP PROCEDURE dbo.cp\_s\_job\_status;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Gets the status of the specified Job    By: Max Vernon  Date: 2016-04-25    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_s\_job\_status  (  @JobID UNIQUEIDENTIFIER  )  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  IF EXISTS (SELECT 1 FROM msdb.dbo.sysjobs\_filter sjf WHERE sjf.job\_id = @JobID)  BEGIN    DECLARE @can\_see\_all\_running\_jobs INT  DECLARE @job\_owner   sysname    DECLARE @job\_execution\_state TABLE  (    job\_id                  UNIQUEIDENTIFIER NOT NULL  , date\_started            INT              NOT NULL  , time\_started            INT              NOT NULL  , execution\_job\_status    INT              NOT NULL  , execution\_step\_id       INT              NULL  , execution\_step\_name     sysname          COLLATE database\_default NULL  , execution\_retry\_attempt INT              NOT NULL  , next\_run\_date           INT              NOT NULL  , next\_run\_time           INT              NOT NULL  , next\_run\_schedule\_id    INT              NOT NULL  );    DECLARE @filtered\_jobs TABLE  (    job\_id                   UNIQUEIDENTIFIER NOT NULL  , date\_created             DATETIME         NOT NULL  , date\_last\_modified       DATETIME         NOT NULL  , current\_execution\_status INT              NULL  , current\_execution\_step   INT         NULL  , current\_execution\_step\_name sysname       COLLATE database\_default NULL  , current\_retry\_attempt    INT              NULL  , last\_run\_date            INT              NOT NULL  , last\_run\_time            INT              NOT NULL  , last\_run\_outcome         INT              NOT NULL  , next\_run\_date            INT              NULL  , next\_run\_time            INT              NULL  , next\_run\_schedule\_id     INT              NULL  , type                     INT              NOT NULL  );    DECLARE @xp\_results TABLE  (    job\_id                UNIQUEIDENTIFIER NOT NULL  , last\_run\_date         INT              NOT NULL  , last\_run\_time         INT              NOT NULL  , next\_run\_date         INT              NOT NULL  , next\_run\_time         INT              NOT NULL  , next\_run\_schedule\_id  INT              NOT NULL  , requested\_to\_run      INT              NOT NULL -- BOOL  , request\_source        INT              NOT NULL  , request\_source\_id     sysname          COLLATE database\_default NULL  , running               INT              NOT NULL -- BOOL  , current\_step          INT              NOT NULL  , current\_retry\_attempt INT              NOT NULL  , job\_state             INT              NOT NULL  );    SET @can\_see\_all\_running\_jobs = 1;  SELECT @job\_owner = SUSER\_SNAME();    INSERT INTO @xp\_results  EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @can\_see\_all\_running\_jobs, @job\_owner, @JobID;    INSERT INTO @job\_execution\_state  SELECT xpr.job\_id  , xpr.last\_run\_date  , xpr.last\_run\_time  , xpr.job\_state  , sjs.step\_id  , sjs.step\_name  , xpr.current\_retry\_attempt  , xpr.next\_run\_date  , xpr.next\_run\_time  , xpr.next\_run\_schedule\_id  FROM @xp\_results xpr  LEFT OUTER JOIN msdb.dbo.sysjobsteps sjs ON ((xpr.job\_id = sjs.job\_id) AND (xpr.current\_step = sjs.step\_id)),  msdb.dbo.sysjobs\_view sjv  WHERE (sjv.job\_id = xpr.job\_id);    INSERT INTO @filtered\_jobs  SELECT sjv.job\_id  , sjv.date\_created  , sjv.date\_modified  , ISNULL(jes.execution\_job\_status, 4) -- Will be NULL if the job is non-local or is not in @job\_execution\_state (NOTE: 4 = STATE\_IDLE)  , current\_execution\_step = jes.execution\_step\_id  , current\_execution\_step\_name = jes.execution\_step\_name  , jes.execution\_retry\_attempt         -- Will be NULL if the job is non-local or is not in @job\_execution\_state  , 0  -- last\_run\_date placeholder  , 0  -- last\_run\_time placeholder  , 5  -- last\_run\_outcome placeholder  , jes.next\_run\_date                   -- Will be NULL if the job is non-local or is not in @job\_execution\_state  , jes.next\_run\_time                   -- Will be NULL if the job is non-local or is not in @job\_execution\_state  , jes.next\_run\_schedule\_id            -- Will be NULL if the job is non-local or is not in @job\_execution\_state  , 0   -- type placeholder  FROM msdb.dbo.sysjobs\_view sjv  LEFT OUTER JOIN @job\_execution\_state jes ON (sjv.job\_id = jes.job\_id)  WHERE sjv.job\_id = @JobID;    UPDATE @filtered\_jobs  SET current\_execution\_status = NULL  WHERE (current\_execution\_status = 4)  AND (job\_id IN (  SELECT job\_id  FROM msdb.dbo.sysjobservers  WHERE (server\_id <> 0))  );    UPDATE @filtered\_jobs  SET last\_run\_date = sjs.last\_run\_date,  last\_run\_time = sjs.last\_run\_time,  last\_run\_outcome = sjs.last\_run\_outcome  FROM @filtered\_jobs         fj,  msdb.dbo.sysjobservers sjs  WHERE (fj.job\_id = sjs.job\_id);    UPDATE @filtered\_jobs  SET type = 1 -- LOCAL  FROM @filtered\_jobs fj,  msdb.dbo.sysjobservers sjs  WHERE (fj.job\_id = sjs.job\_id)  AND (server\_id = 0);  UPDATE @filtered\_jobs  SET type = 2 -- MULTI-SERVER  FROM @filtered\_jobs fj,  msdb.dbo.sysjobservers sjs  WHERE (fj.job\_id = sjs.job\_id)  AND (server\_id <> 0);    SELECT JobID = sjv.job\_id  , JobName = sjv.name  , JobEnabled = sjv.enabled  , JobDescription = sjv.description  , StartStepID = sjv.start\_step\_id  , last\_run\_date\_time = CASE WHEN fj.last\_run\_date > 0 THEN msdb.dbo.agent\_datetime(fj.last\_run\_date, fj.last\_run\_time) ELSE NULL END  , fj.last\_run\_outcome  , current\_execution\_status = COALESCE(fj.current\_execution\_status, 0)  , current\_execution\_step = COALESCE(fj.current\_execution\_step, 0)  , current\_execution\_step\_name = fj.current\_execution\_step\_name  , current\_retry\_attempt = COALESCE(fj.current\_retry\_attempt, 0)  FROM @filtered\_jobs fj  LEFT OUTER JOIN msdb.dbo.sysjobs\_view sjv ON (fj.job\_id = sjv.job\_id)  ORDER BY sjv.job\_id;    END  ELSE  BEGIN  DECLARE @msg NVARCHAR(1000);  SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';  RAISERROR (@msg, 11, 1);  END  END  GO  IF OBJECT\_ID('dbo.cp\_e\_start\_job') IS NOT NULL  DROP PROCEDURE dbo.cp\_e\_start\_job;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Starts the specified Job    By: Max Vernon  Date: 2016-04-25    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_e\_start\_job  (  @JobID UNIQUEIDENTIFIER  )  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  DECLARE @msg NVARCHAR(1000);  IF EXISTS (  SELECT 1  FROM dbo.sysjobs\_filter sjmf  INNER JOIN dbo.sysjobs sj ON sjmf.job\_id = sj.job\_id  WHERE sjmf.job\_id = @JobID  )  BEGIN  INSERT INTO dbo.sysjobs\_log (job\_id, ActionTypeID)  VALUES (@JobID, 1); --1 is "Started Job"  EXEC dbo.sp\_start\_job @job\_id = @JobID;  END  ELSE  BEGIN  SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';  RAISERROR (@msg, 11, 1);  END  END  GO  IF OBJECT\_ID('dbo.cp\_e\_stop\_job') IS NOT NULL  DROP PROCEDURE dbo.cp\_e\_stop\_job;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Stops the specified Job    By: Max Vernon  Date: 2016-04-25    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_e\_stop\_job  (  @JobID UNIQUEIDENTIFIER  )  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  DECLARE @msg NVARCHAR(1000);  IF EXISTS (  SELECT 1  FROM dbo.sysjobs\_filter sjmf  INNER JOIN dbo.sysjobs sj ON sjmf.job\_id = sj.job\_id  WHERE sjmf.job\_id = @JobID  )  BEGIN  INSERT INTO dbo.sysjobs\_log (job\_id, ActionTypeID)  VALUES (@JobID, 2); --1 is "Started Job"  EXEC dbo.sp\_stop\_job @job\_id = @JobID;  END  ELSE  BEGIN  SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';  RAISERROR (@msg, 11, 1);  END  END  GO  IF OBJECT\_ID('dbo.cp\_s\_job\_control\_perms') IS NOT NULL  DROP PROCEDURE dbo.cp\_s\_job\_control\_perms;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Confirms the objects exist and the determines the callers  security level for the SQL Agent Job Control system.    By: Max Vernon  Date: 2016-04-27    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_s\_job\_control\_perms  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  DECLARE @OutputList TABLE  (  ObjectName SYSNAME NOT NULL  , ObjectVal INT NOT NULL  );  INSERT INTO @OutputList (ObjectName, ObjectVal)  SELECT ObjectName = 'dbo.cp\_s\_job\_list', ObjectID = COALESCE(OBJECT\_ID('dbo.cp\_s\_job\_list'), 0)  UNION ALL  SELECT 'dbo.cp\_s\_job\_status', COALESCE(OBJECT\_ID('dbo.cp\_s\_job\_status'), 0)  UNION ALL  SELECT 'dbo.cp\_e\_stop\_job', COALESCE(OBJECT\_ID('dbo.cp\_e\_stop\_job'), 0)  UNION ALL  SELECT 'dbo.cp\_e\_start\_job', COALESCE(OBJECT\_ID('dbo.cp\_e\_start\_job'), 0)  EXECUTE AS CALLER;  INSERT INTO @OutputList (ObjectName, ObjectVal)  SELECT 'ControlPermissions', (  SELECT COUNT(1)  FROM sys.fn\_my\_permissions('dbo.sysjobs', 'OBJECT') p  WHERE p.permission\_name = 'CONTROL'  );  REVERT;    SELECT \*  FROM @OutputList;  END  GO  IF OBJECT\_ID('dbo.cp\_s\_get\_job\_outcome') IS NOT NULL  DROP PROCEDURE dbo.cp\_s\_get\_job\_outcome;  GO  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Gets the most recent history item for the specified job      By: Max Vernon  Date: 2016-04-27    Version: 1.0 Initial Version  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  CREATE PROCEDURE dbo.cp\_s\_get\_job\_outcome  (  @JobID UNIQUEIDENTIFIER  )  WITH EXECUTE AS OWNER  AS  BEGIN  SET NOCOUNT ON;  SELECT TOP(1) sj.job\_id    , [Job Name] = sj.name    , [Step Name] = sjh.step\_name    , sjh.message  FROM msdb.dbo.sysjobs sj    INNER JOIN msdb.dbo.sysjobhistory sjh ON sj.job\_id = sjh.job\_id  WHERE sj.job\_id = @JobID  ORDER BY msdb.dbo.agent\_datetime(sjh.run\_date, sjh.run\_time) DESC  , sjh.step\_id;  END  GO    IF NOT EXISTS (  SELECT 1  FROM sys.database\_principals dp  WHERE dp.name = 'agent\_job\_control'  )  CREATE ROLE agent\_job\_control AUTHORIZATION [dbo];  GRANT EXECUTE ON dbo.cp\_s\_job\_list TO agent\_job\_control;  GRANT EXECUTE ON dbo.cp\_s\_job\_status TO agent\_job\_control;  GRANT EXECUTE ON dbo.cp\_e\_start\_job TO agent\_job\_control;  GRANT EXECUTE ON dbo.cp\_e\_stop\_job TO agent\_job\_control;  GRANT EXECUTE ON dbo.cp\_s\_job\_control\_perms TO agent\_job\_control;  GRANT EXECUTE ON dbo.cp\_s\_get\_job\_outcome TO agent\_job\_control;  GO |

You’ll need to create user principals in the msdb object, and add them to the agent\_job\_control database role. Once that is completed those users can start and stop jobs, and obtain the status and recent history of jobs, by specifying the job\_id uniqueidentifier.

Project Code is here: [SQLServerJobControl](https://www.sqlserverscience.com/wp-content/uploads/2016/04/SQLServerJobControl.zip)

/\*

Script to create objects required for the Job Control App

By: Max Vernon

Date: 2016-04-27

Notes: Compatible with SQL Server 2005+ (not tested on 2014+)

\*/

USE msdb;

GO

IF OBJECT\_ID('dbo.sysjobs\_log') IS NOT NULL

DROP TABLE dbo.sysjobs\_log;

GO

IF OBJECT\_ID('dbo.sysjobs\_action\_types') IS NOT NULL

DROP TABLE dbo.sysjobs\_action\_types;

GO

CREATE TABLE dbo.sysjobs\_action\_types

(

ActionTypeID INT NOT NULL

CONSTRAINT PK\_sysjobs\_action\_types

PRIMARY KEY

CLUSTERED

, ActionDescription VARCHAR(100) NOT NULL

) ON [PRIMARY];

INSERT INTO dbo.sysjobs\_action\_types (ActionTypeID, ActionDescription)

VALUES (1, 'Started Job')

INSERT INTO dbo.sysjobs\_action\_types (ActionTypeID, ActionDescription)

VALUES (2, 'Stopped Job');

GO

CREATE TABLE dbo.sysjobs\_log

(

sysjobs\_log\_ID INT NOT NULL

CONSTRAINT PK\_sysjobs\_log

PRIMARY KEY

CLUSTERED

IDENTITY(1,1)

, job\_id UNIQUEIDENTIFIER NOT NULL

CONSTRAINT FK\_sysjobs\_log\_\_job\_id

FOREIGN KEY

REFERENCES dbo.sysjobs(job\_id)

ON DELETE CASCADE

, JobActionDateTime DATETIME NOT NULL

CONSTRAINT DF\_sysjobs\_log\_JobActionDateTime

DEFAULT (GETDATE())

, ActionBy SYSNAME NOT NULL

CONSTRAINT DF\_sysjobs\_log\_StartedBy

DEFAULT (ORIGINAL\_LOGIN())

, ActionAs SYSNAME NOT NULL

CONSTRAINT DF\_sysjobs\_log\_StartedAs

DEFAULT (SUSER\_SNAME())

, ActionTypeID INT NOT NULL

CONSTRAINT FK\_sysjobs\_log\_\_ActionType

FOREIGN KEY

REFERENCES dbo.sysjobs\_action\_types(ActionTypeID)

) ON [PRIMARY];

GO

IF OBJECT\_ID('dbo.sysjobs\_filter') IS NOT NULL

DROP TABLE dbo.sysjobs\_filter;

GO

CREATE TABLE dbo.sysjobs\_filter

(

job\_id UNIQUEIDENTIFIER NOT NULL

CONSTRAINT FK\_sysjobs\_filter\_\_job\_id

FOREIGN KEY

REFERENCES dbo.sysjobs(job\_id)

ON DELETE CASCADE

) ON [PRIMARY];

CREATE INDEX PK\_sysjobs\_filter

ON dbo.sysjobs\_filter(job\_id);

GO

IF OBJECT\_ID('dbo.cp\_s\_job\_list') IS NOT NULL

DROP PROCEDURE dbo.cp\_s\_job\_list;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Shows a list of jobs that are available for non-DBAs to run

using the dbo.cp\_s\_start\_job interface

By: Max Vernon

Date: 2016-04-25

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_s\_job\_list

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

SELECT JobID = sj.job\_id

, JobName = sj.name

, JobDescription = sj.description

, IsEnabled = CASE WHEN sj.enabled = 1 THEN CONVERT(BIT, 1) ELSE CONVERT(BIT, 0) END

, LastRunDate = (SELECT TOP(1) msdb.dbo.agent\_datetime(sjh.run\_date, sjh.run\_time)

FROM msdb.dbo.sysjobhistory sjh

WHERE sjh.job\_id = sj.job\_id

ORDER BY sjh.run\_date DESC, sjh.run\_time DESC)

FROM msdb.dbo.sysjobs sj

INNER JOIN msdb.dbo.sysjobs\_filter sjf ON sj.job\_id = sjf.job\_id

ORDER BY sj.name;

END

GO

IF OBJECT\_ID('dbo.cp\_s\_job\_status') IS NOT NULL

DROP PROCEDURE dbo.cp\_s\_job\_status;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Gets the status of the specified Job

By: Max Vernon

Date: 2016-04-25

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_s\_job\_status

(

@JobID UNIQUEIDENTIFIER

)

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS (SELECT 1 FROM msdb.dbo.sysjobs\_filter sjf WHERE sjf.job\_id = @JobID)

BEGIN

DECLARE @can\_see\_all\_running\_jobs INT

DECLARE @job\_owner sysname

DECLARE @job\_execution\_state TABLE

(

job\_id UNIQUEIDENTIFIER NOT NULL

, date\_started INT NOT NULL

, time\_started INT NOT NULL

, execution\_job\_status INT NOT NULL

, execution\_step\_id INT NULL

, execution\_step\_name sysname COLLATE database\_default NULL

, execution\_retry\_attempt INT NOT NULL

, next\_run\_date INT NOT NULL

, next\_run\_time INT NOT NULL

, next\_run\_schedule\_id INT NOT NULL

);

DECLARE @filtered\_jobs TABLE

(

job\_id UNIQUEIDENTIFIER NOT NULL

, date\_created DATETIME NOT NULL

, date\_last\_modified DATETIME NOT NULL

, current\_execution\_status INT NULL

, current\_execution\_step INT NULL

, current\_execution\_step\_name sysname COLLATE database\_default NULL

, current\_retry\_attempt INT NULL

, last\_run\_date INT NOT NULL

, last\_run\_time INT NOT NULL

, last\_run\_outcome INT NOT NULL

, next\_run\_date INT NULL

, next\_run\_time INT NULL

, next\_run\_schedule\_id INT NULL

, type INT NOT NULL

);

DECLARE @xp\_results TABLE

(

job\_id UNIQUEIDENTIFIER NOT NULL

, last\_run\_date INT NOT NULL

, last\_run\_time INT NOT NULL

, next\_run\_date INT NOT NULL

, next\_run\_time INT NOT NULL

, next\_run\_schedule\_id INT NOT NULL

, requested\_to\_run INT NOT NULL -- BOOL

, request\_source INT NOT NULL

, request\_source\_id sysname COLLATE database\_default NULL

, running INT NOT NULL -- BOOL

, current\_step INT NOT NULL

, current\_retry\_attempt INT NOT NULL

, job\_state INT NOT NULL

);

SET @can\_see\_all\_running\_jobs = 1;

SELECT @job\_owner = SUSER\_SNAME();

INSERT INTO @xp\_results

EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs @can\_see\_all\_running\_jobs, @job\_owner, @JobID;

INSERT INTO @job\_execution\_state

SELECT xpr.job\_id

, xpr.last\_run\_date

, xpr.last\_run\_time

, xpr.job\_state

, sjs.step\_id

, sjs.step\_name

, xpr.current\_retry\_attempt

, xpr.next\_run\_date

, xpr.next\_run\_time

, xpr.next\_run\_schedule\_id

FROM @xp\_results xpr

LEFT OUTER JOIN msdb.dbo.sysjobsteps sjs ON ((xpr.job\_id = sjs.job\_id) AND (xpr.current\_step = sjs.step\_id)),

msdb.dbo.sysjobs\_view sjv

WHERE (sjv.job\_id = xpr.job\_id);

INSERT INTO @filtered\_jobs

SELECT sjv.job\_id

, sjv.date\_created

, sjv.date\_modified

, ISNULL(jes.execution\_job\_status, 4) -- Will be NULL if the job is non-local or is not in @job\_execution\_state (NOTE: 4 = STATE\_IDLE)

, current\_execution\_step = jes.execution\_step\_id

, current\_execution\_step\_name = jes.execution\_step\_name

, jes.execution\_retry\_attempt -- Will be NULL if the job is non-local or is not in @job\_execution\_state

, 0 -- last\_run\_date placeholder

, 0 -- last\_run\_time placeholder

, 5 -- last\_run\_outcome placeholder

, jes.next\_run\_date -- Will be NULL if the job is non-local or is not in @job\_execution\_state

, jes.next\_run\_time -- Will be NULL if the job is non-local or is not in @job\_execution\_state

, jes.next\_run\_schedule\_id -- Will be NULL if the job is non-local or is not in @job\_execution\_state

, 0 -- type placeholder

FROM msdb.dbo.sysjobs\_view sjv

LEFT OUTER JOIN @job\_execution\_state jes ON (sjv.job\_id = jes.job\_id)

WHERE sjv.job\_id = @JobID;

UPDATE @filtered\_jobs

SET current\_execution\_status = NULL

WHERE (current\_execution\_status = 4)

AND (job\_id IN (

SELECT job\_id

FROM msdb.dbo.sysjobservers

WHERE (server\_id <> 0))

);

UPDATE @filtered\_jobs

SET last\_run\_date = sjs.last\_run\_date,

last\_run\_time = sjs.last\_run\_time,

last\_run\_outcome = sjs.last\_run\_outcome

FROM @filtered\_jobs fj,

msdb.dbo.sysjobservers sjs

WHERE (fj.job\_id = sjs.job\_id);

UPDATE @filtered\_jobs

SET type = 1 -- LOCAL

FROM @filtered\_jobs fj,

msdb.dbo.sysjobservers sjs

WHERE (fj.job\_id = sjs.job\_id)

AND (server\_id = 0);

UPDATE @filtered\_jobs

SET type = 2 -- MULTI-SERVER

FROM @filtered\_jobs fj,

msdb.dbo.sysjobservers sjs

WHERE (fj.job\_id = sjs.job\_id)

AND (server\_id <> 0);

SELECT JobID = sjv.job\_id

, JobName = sjv.name

, JobEnabled = sjv.enabled

, JobDescription = sjv.description

, StartStepID = sjv.start\_step\_id

, last\_run\_date\_time = CASE WHEN fj.last\_run\_date > 0 THEN msdb.dbo.agent\_datetime(fj.last\_run\_date, fj.last\_run\_time) ELSE NULL END

, fj.last\_run\_outcome

, current\_execution\_status = COALESCE(fj.current\_execution\_status, 0)

, current\_execution\_step = COALESCE(fj.current\_execution\_step, 0)

, current\_execution\_step\_name = fj.current\_execution\_step\_name

, current\_retry\_attempt = COALESCE(fj.current\_retry\_attempt, 0)

FROM @filtered\_jobs fj

LEFT OUTER JOIN msdb.dbo.sysjobs\_view sjv ON (fj.job\_id = sjv.job\_id)

ORDER BY sjv.job\_id;

END

ELSE

BEGIN

DECLARE @msg NVARCHAR(1000);

SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';

RAISERROR (@msg, 11, 1);

END

END

GO

IF OBJECT\_ID('dbo.cp\_e\_start\_job') IS NOT NULL

DROP PROCEDURE dbo.cp\_e\_start\_job;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Starts the specified Job

By: Max Vernon

Date: 2016-04-25

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_e\_start\_job

(

@JobID UNIQUEIDENTIFIER

)

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

DECLARE @msg NVARCHAR(1000);

IF EXISTS (

SELECT 1

FROM dbo.sysjobs\_filter sjmf

INNER JOIN dbo.sysjobs sj ON sjmf.job\_id = sj.job\_id

WHERE sjmf.job\_id = @JobID

)

BEGIN

INSERT INTO dbo.sysjobs\_log (job\_id, ActionTypeID)

VALUES (@JobID, 1); --1 is "Started Job"

EXEC dbo.sp\_start\_job @job\_id = @JobID;

END

ELSE

BEGIN

SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';

RAISERROR (@msg, 11, 1);

END

END

GO

IF OBJECT\_ID('dbo.cp\_e\_stop\_job') IS NOT NULL

DROP PROCEDURE dbo.cp\_e\_stop\_job;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Stops the specified Job

By: Max Vernon

Date: 2016-04-25

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_e\_stop\_job

(

@JobID UNIQUEIDENTIFIER

)

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

DECLARE @msg NVARCHAR(1000);

IF EXISTS (

SELECT 1

FROM dbo.sysjobs\_filter sjmf

INNER JOIN dbo.sysjobs sj ON sjmf.job\_id = sj.job\_id

WHERE sjmf.job\_id = @JobID

)

BEGIN

INSERT INTO dbo.sysjobs\_log (job\_id, ActionTypeID)

VALUES (@JobID, 2); --1 is "Started Job"

EXEC dbo.sp\_stop\_job @job\_id = @JobID;

END

ELSE

BEGIN

SET @msg = CONVERT(NVARCHAR(100), @JobID) + ' is not a valid job identifier.';

RAISERROR (@msg, 11, 1);

END

END

GO

IF OBJECT\_ID('dbo.cp\_s\_job\_control\_perms') IS NOT NULL

DROP PROCEDURE dbo.cp\_s\_job\_control\_perms;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Confirms the objects exist and the determines the callers

security level for the SQL Agent Job Control system.

By: Max Vernon

Date: 2016-04-27

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_s\_job\_control\_perms

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

DECLARE @OutputList TABLE

(

ObjectName SYSNAME NOT NULL

, ObjectVal INT NOT NULL

);

INSERT INTO @OutputList (ObjectName, ObjectVal)

SELECT ObjectName = 'dbo.cp\_s\_job\_list', ObjectID = COALESCE(OBJECT\_ID('dbo.cp\_s\_job\_list'), 0)

UNION ALL

SELECT 'dbo.cp\_s\_job\_status', COALESCE(OBJECT\_ID('dbo.cp\_s\_job\_status'), 0)

UNION ALL

SELECT 'dbo.cp\_e\_stop\_job', COALESCE(OBJECT\_ID('dbo.cp\_e\_stop\_job'), 0)

UNION ALL

SELECT 'dbo.cp\_e\_start\_job', COALESCE(OBJECT\_ID('dbo.cp\_e\_start\_job'), 0)

EXECUTE AS CALLER;

INSERT INTO @OutputList (ObjectName, ObjectVal)

SELECT 'ControlPermissions', (

SELECT COUNT(1)

FROM sys.fn\_my\_permissions('dbo.sysjobs', 'OBJECT') p

WHERE p.permission\_name = 'CONTROL'

);

REVERT;

SELECT \*

FROM @OutputList;

END

GO

IF OBJECT\_ID('dbo.cp\_s\_get\_job\_outcome') IS NOT NULL

DROP PROCEDURE dbo.cp\_s\_get\_job\_outcome;

GO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Gets the most recent history item for the specified job

By: Max Vernon

Date: 2016-04-27

Version: 1.0 Initial Version

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE PROCEDURE dbo.cp\_s\_get\_job\_outcome

(

@JobID UNIQUEIDENTIFIER

)

WITH EXECUTE AS OWNER

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP(1) sj.job\_id

, [Job Name] = sj.name

, [Step Name] = sjh.step\_name

, sjh.message

FROM msdb.dbo.sysjobs sj

INNER JOIN msdb.dbo.sysjobhistory sjh ON sj.job\_id = sjh.job\_id

WHERE sj.job\_id = @JobID

ORDER BY msdb.dbo.agent\_datetime(sjh.run\_date, sjh.run\_time) DESC

, sjh.step\_id;

END

GO

IF NOT EXISTS (

SELECT 1

FROM sys.database\_principals dp

WHERE dp.name = 'agent\_job\_control'

)

CREATE ROLE agent\_job\_control AUTHORIZATION [dbo];

GRANT EXECUTE ON dbo.cp\_s\_job\_list TO agent\_job\_control;

GRANT EXECUTE ON dbo.cp\_s\_job\_status TO agent\_job\_control;

GRANT EXECUTE ON dbo.cp\_e\_start\_job TO agent\_job\_control;

GRANT EXECUTE ON dbo.cp\_e\_stop\_job TO agent\_job\_control;

GRANT EXECUTE ON dbo.cp\_s\_job\_control\_perms TO agent\_job\_control;

GRANT EXECUTE ON dbo.cp\_s\_get\_job\_outcome TO agent\_job\_control;

GO