Scheduling Jobs in LANDesk Directly through SQL.

Software Deployment and other Scripts can be inserted into the Management Database directly using SQL statements and set to run in the future or from the command line on a console using CUSTJOB.EXE.

The Management Database and the Console

All data for the Console is contained in the Management Database, PDDB. The Web Console uses the DataMart, DMDB. In most cases it is easier to work with the DataMart than the Management Database, however, in this case the opposite is true. For this article I am using MS SQL Server 2000, some things may differ slightly if you are using Oracle or DB2.

Scheduling Tasks in the Management Database

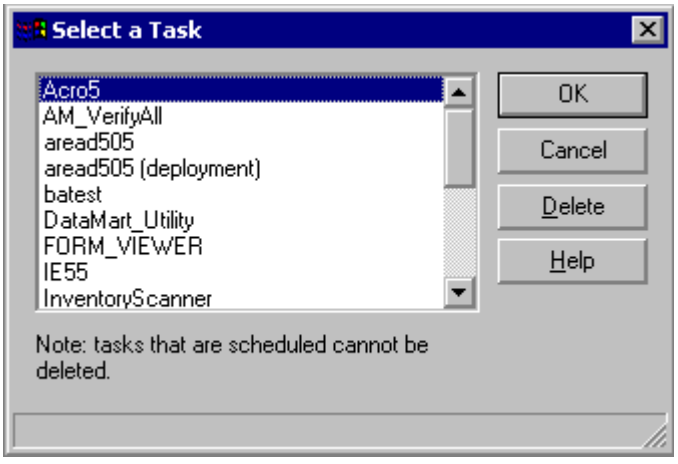
Information about scheduled jobs is stored in just four tables in PDDB, and most of the information is readily comprehensible to humans as well. The key tables are:

- LD_TASK
- LD_TASK_CONFIG
- LD_TASK_MACHINE
- LD_SCHEDULE_EXE

In scheduling your own jobs through SQL, you will not need to deal with LD_SCHEDULE_EXE which defines some operations for the console.

LD_TASK_CONFIG

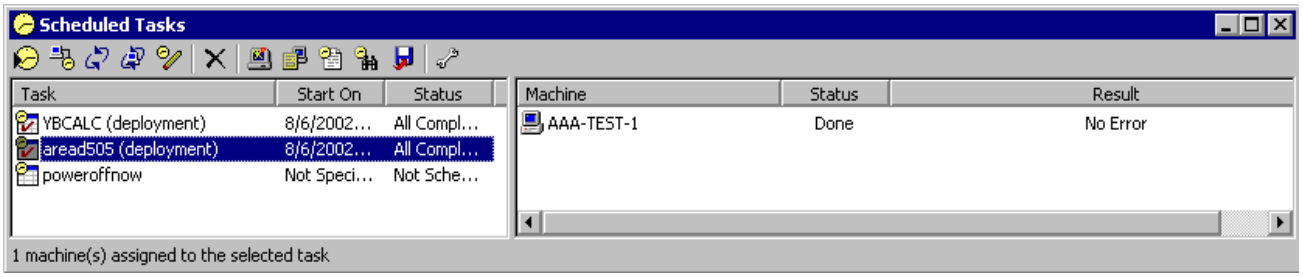
LD_TASK_CONFIG corresponds to the Select A Task window opened when DTM:Tools:"Create Distribution Package Script" is selected. The table has only four columns, the last of which can be ignored. EXE_IDN is always 2. CFG_NAME is the name of the INI file for the script. CFG_IDN is a number used to reference the script in LD_TASK



TASK_CFG_IDN	EXE_IDN	CFG_NAME	CFG_PARAMS
2	2	YBCALC (deployment)	<Binary>
3	2	aread505 (deployment)	<Binary>
4	2	removeYBCalc	<Binary>
8	2	winzip	<Binary>
9	2	FORM_VIEWER	<Binary>
10	2	DataMart_Utility	<Binary>
17	2	poweroffnow	<Binary>

LD_TASK

LD_TASK contains the data in the Scheduled Tasks Window. To fit the LD_TASK table on the page I've omitted columns you're not likely to need to manipulate.



TASK_IDN	TASK_NAME	TASK_CFG_IDN	TASK_STATUS	TASK_WAKE_MACHINES	NEXT_START	LAST_STATUS	MAX_RETRIES	TIMES_TRIED
1	YBCALC (deployment)	2	5	0	08/06/2002 14:11	5	0	2
2	aread505 (deployment)	3	5	0	08/06/2002 15:15	5	0	1
3	poweroffnow	17	9	0	12/31/1969 19:00	0	0	0

TASK_NAME is also the name of the script: `..\DTM\SCRIPTS\%SCRIPTNAME%.INI` and matches a column in LD_TASK_CONFIG. TASK_CFG_IDN comes directly from LD_TASK_CONFIG and is required to schedule the task. TASK_IDN is the TaskID parameter for CUSTJOB. The other fields are self-explanatory. The date fields often default to values long ago in the past, NEXT_START contains a datetime value which if set to a future time will run that job at that time.

The following values are valid for TASK_STATUS:

0	1	3	4	5	6
Waiting	Working	Failed	Service Stopped	All Complete	Partial Complete
7	8	9	10	11	12
None Complete	Failed	Not Scheduled	Waiting	Failed	Available for Pull

LD_TASK_MACHINE

This table associates distinct machines with a scheduled job. It corresponds to the right pane in the scheduled tasks window, except that one table supports all of the tasks. TASK_IDN is the foreign key for the LD_TASK table. MAC_OBJID is the value OBJECTROOT_ID taken from the LD_OBJECTROOT table, which can be queried " USE PDDB \GO \SELECT OBJECT_ROOT_IDN FROM LANDESK.LD_OBJECTROOT WHERE DEVICE_NAME = 'COMPUTER' ". A MAC_STATUS of 0 means the machine is waiting for the job, any other value means it has been run. The final two columns deal with the results of the last run.

TASK_IDN	MAC_OBJID	MAC_STATUS	MAC_RETCODE	MAC_WOKE_UP
1	517	2	229638144	0
2	911	2	229638144	0
3	517	4	1201	0
3	911	4	1201	0

SQL Stored Procedure to Schedule a LANDesk Job

```
-- Create a Stored Procedure to schedule a job in LANDesk
-- Requires @JOB = script name in DTM\scripts directory (no .ini extension.)
-- Optional @WAKE (default 0, 1 to wake computers),
-- @Retries (number of times to retry failed attempts),
-- @STARTTIME, the time the job should start, if not specified or Null,
-- the job must be manually kicked off by running CUSTJOB.EXE with the TASKID
-- returned at the end of the Procedure.
-- The job requires TEMP_MACHINES to contain the DEVICENAMES of the targets.

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

-- FOR SET_JOB to work only DEVICENAME is needed, the other fields are defined
-- to match the table this one will be imported from in my environment.
-- CREATE TABLE [TEMP_MACHINES] (
--     [DEVICENAME] [varchar] (30) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
--     [OS_SHORT] [smallint] NULL DEFAULT (0),
--     [SERVER] [smallint] NULL DEFAULT (0),
--     [LOCATION] [char] (12) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
--     [SHUT_PRIORITY] [smallint] NULL DEFAULT (0)
-- ) ON [PRIMARY]
-- GO

CREATE PROCEDURE SET_JOB

@JOB VARCHAR(50),
@WAKE_MACHINES SMALLINT = 0,
@MAX_RETRIES SMALLINT = 0,
@STARTTIME DATETIME = NULL

AS
DECLARE @TASK_IDN INT
DECLARE @TASK_CFG_IDN INT
DECLARE @EXE_IDN INT
DECLARE @EMPTY1 INT -- A placeholder
DECLARE @EMPTY2 INT -- A placeholder
DECLARE @MAC_OBJID INT
DECLARE @DEVICE_NAME varchar(30)

-- Inserting a new task, need a new IDN.

SET @TASK_IDN =
      1+
      (
      select max ( task_idn )
      from LD_TASK
      )

-- EXE_IDN TELLS TASK_CONFIG TO RUN CUSTJOB!
-- The number for EXE_IDN can vary between Core Server's as the numbers
-- seem to be assigned the first time each job type is run!
```

```
--      (there are only three types I've seen in 6.4 & 6,.5)
-- IF SCRIPT ALREADY DEFINED IN TASK_CONFIG, USE EXISTING TASK_CFG_IDN
-- IF NOT DEFINED, ADD IT, USING NEXT AVAILABLE TASK_CFG_IDN.

SET @EXE_IDN = ( SELECT EXE_IDN FROM LD_SCHEDULE_EXE
      WHERE EXE_UNC = 'CUSTJOB.EXE' )
SET @TASK_CFG_IDN = ( SELECT TASK_CFG_IDN FROM
      LD_TASK_CONFIG WHERE CFG_NAME = @JOB )
IF @TASK_CFG_IDN IS NULL
BEGIN
SET @TASK_CFG_IDN =
      1+
      (
      select max ( task_cfg_idn )
      from ld_task_config
      )
INSERT LD_TASK_CONFIG
      ( TASK_CFG_IDN, EXE_IDN, CFG_NAME )
      VALUES ( @TASK_CFG_IDN, @EXE_IDN, @JOB )
END

CREATE TABLE TEMP_OBJECTS
      (
      DEVICE_NAME  VARCHAR(30),
      MAC_OBJID    INT,
      TASK_CFG_IDN INT,
      TASK_IDN     INT,
      MAC_STATUS   INT,
      MAC_RETCODE  INT,
      MAC_WOKE_UP  INT
      )

INSERT TEMP_OBJECTS ( DEVICE_NAME )
      SELECT DEVICENAME FROM TEMP_MACHINES

DECLARE MY_OBJ CURSOR
      FOR SELECT DEVICE_NAME, MAC_OBJID, TASK_CFG_IDN, TASK_IDN
      FROM TEMP_OBJECTS

-- Get the identifier MAC_OBJID for the machine into TEMP_OBJECTS.
-- If the machine has no identifier in inventory delete it from the job.

OPEN MY_OBJ
FETCH NEXT FROM MY_OBJ
            INTO @DEVICE_NAME, @MAC_OBJID, @EMPTY1, @EMPTY2
WHILE @@FETCH_STATUS = 0
BEGIN
      SET @MAC_OBJID =
      (
      select OBJECT_ROOT_IDN
      FROM LD_OBJECTROOT
      WHERE DEVICE_NAME = @DEVICE_NAME
      )
      IF @MAC_OBJID > 0
      BEGIN
            UPDATE TEMP_OBJECTS
            SET MAC_OBJID = @MAC_OBJID,
                TASK_IDN = @TASK_IDN,
                MAC_STATUS = 0,
                MAC_RETCODE = 0,
                MAC_WOKE_UP = 0
            WHERE DEVICE_NAME = @DEVICE_NAME
      END
      ELSE
```

```
	BEGIN
		DELETE TEMP_OBJECTS
		WHERE DEVICE_NAME = @DEVICE_NAME
		PRINT 'DELETING ' + @DEVICE_NAME
	END
FETCH NEXT FROM MY_OBJ
	INTO @DEVICE_NAME, @MAC_OBJID, @EMPTY1, @EMPTY2
END

CLOSE MY_OBJ
DEALLOCATE MY_OBJ

DECLARE @LONGAGO DATETIME
SET @LONGAGO = '1/1/1970'
INSERT LD_TASK
	( TASK_IDN , TASK_NAME, TASK_CFG_IDN, TASK_STATUS, TASK_WAKE_MACHINES,
	  NEXT_START, LAST_MODIFIED, LAST_START, LAST_END, LAST_STATUS,
	  DAY_WEEK_FLAGS, DAY_MONTH_FLAGS, MAX_RETRIES, TIMES_TRIED )
	VALUES ( @TASK_IDN, @JOB, @TASK_CFG_IDN, 0, @WAKE_MACHINES, @STARTTIME,
	  @LONGAGO, @LONGAGO, @LONGAGO, 9, 0, 0, @MAX_RETRIES, 0 )

INSERT INTO LD_TASK_MACHINE
		( TASK_IDN, MAC_OBJID, MAC_STATUS, MAC_RETCODE, MAC_WOKE_UP )
	SELECT TASK_IDN, MAC_OBJID, MAC_STATUS, MAC_RETCODE, MAC_WOKE_UP
	FROM TEMP_OBJECTS

DROP TABLE TEMP_OBJECTS

SELECT @TASK_IDN AS TASK_IDN

GO
```

History

Researched August 2002, by John Karr

Original Author, John Karr.