**DBAs and the ITIL Framework**

**By**[**John F. Tamburo**](http://www.sqlservercentral.com/Authors/Articles/John_F._Tamburo/572434/)**, 2015/10/26**

ITIL: The *Information Technology Infrastructure Library*. Most DBA out there have heard of ITIL; many think that ITIL and DBA functions touch only tangentially or not at all. Considering that many USA companies are pushing their IT operations toward ITIL, it's time for DBA to start learning about ITIL and how it can affect us.

Once we understand ITIL, we can begin to see that implementing it will make our job easier. It may *seem* like a nightmarish pile of bureaucratic steps, but that impression is illusory.

**The DBA's Job within ITIL**

Consider some of the things with ITIL. I'll pick a few and give examples of how ITIL and DBA work intersects, and how DBA will benefit from following ITIL-aligned practices and using ITIL-aligned tool sets.

Let's consider ITIL aligned Service Portfolio Management, Change Management, Release/Deployment Management, and Event management.

**Service Portfolio Management**

Do DBAs offer Services to the end user community? Of course we do! If any database supports any application that any other company employee uses, we have services that we offer to our customers. Who is our "customer?" Everyone in the company. We may deal most often with IT development and operations professionals. However that fact does not change the identities of our ultimate customers.

Here's an example of an ITIL-aligned Service Catalog category for DBA:

***Database Services***

* **I Need a New Database**  
  Help our company's data sprawl! Use this form to request a new database.
* **I Need to Retire a Database**  
  Database, you're outta here! Use this form to archive and delete an existing database that's no longer in use and no longer needs to be online.
* **I Need to Refresh a Lower Environment**  
  Are you developing or testing an application and need a production copy in which to do this? Use this form to request a refresh.

Each menu option, in this case, opens up a specific form template within a servicing tool. Examples of servicing tools that offer ITIL aligned service catalogs include the freeware [OTRS](http://www.otrs.org/)and ManageEngine's [ServiceDesk Plus](https://www.manageengine.com/products/service-desk/).

The form allows you to ask your questions up front. You design a form that asks for sizes, transaction velocities, RPO/RTO goals and the like. Also, the form can also have approvals and the appropriate approvers or groups installed into it, and you don't get hit with the request to create the database **until and unless it's approved**.

And then there are **work flows**. Define your every task in the workflow. You now have a checklist built into the software, and the ability to log your work against the workflow tasks. This helps to ensure that steps do not fall through the cracks. That in turn increases DBA's quality of work, and that in turn helps fund your new [Tesla Model S](http://www.teslamotors.com/models).

**Change Management**

Some people think that [Grant Fritchey](https://www.simple-talk.com/author/grant-fritchey/) is overboard with his emphasis on Change Management, and his steering RedGate toward building one heck of a robust SQL source control tool. Nothing could be further from the truth. Grant is performing a most valuable service to DBA by stressing this as he does. DISCLAIMER: Nothing in this paragraph shall be construed as my endorsement of [wearing kilts at SQLPass Summit](http://www.sqlpass.org/summit/2014/Connect/AfterHours.aspx). But I digress.

But source code management is but one aspect of change management. We DBA have to deal with: (a) changes in the underlying machine(s) running SQL Server; and (b) P-to-V or V-to-P; and (c) SQL Configuration changes; and (d) SQL Job and or Job Schedule changes; and (e) SQL Logins and Users; and (f) Database configuration changes (e.g., Simple to Full recovery and Log backup changes). **All of these changes need to be well-managed, and ITIL makes this happen.**

An ITIL-aligned tool will have an Asset/CMDB, with all servers, workstations, VMs and clusters and all other capital assets included, along with software configurations for each of them. It will also have a Change Request management system that links a change to the asset and or its installed software.

Again, as with a service catalog (which requests a service from outside the group), a Change request submits, approves and implements work flows to accomplish changes. Here's an example:

You have a production SQL Server running SQL 2014. You want to install SP1. You open a CR, creating a set of steps to install the update in a lower environment, complete the requisite testing, arrange production downtime, install the information, and update the CMDB with the changes (usually via an automatic scan).

The CR is then presented for approval. The appropriate stakeholders approve the change. The steps are executed as planned and work logged. Work is completed, then presented to the reviewer. The CR is closed. If you look into the CMDB at the server(s) affected, you will see the change request annotated to the server's history.

What does this buy the DBA? How about a complete change history with dates and notes!  If a problem develops later on, such histories can be instrumental in detecting the problem's root cause.

Here's another example: You have a group of users who need access to the production database.  The server has SOX-implicated data resident on it.  The business stakeholder creates a request to add users.  The appropriate approvals are automatically sought and obtained before the work is scheduled.  Your AD admins create a group with the requisite users.  When you do the work of adding the new AD group to the SQL Server, the asset (server) is tied to the work, and the change(s) are stored in the server's history.

When the auditor is checking out the SOX-implicated server and sees that the logins were changed from last audit, the completed approved request stops the auditor's "ding" on the company's report dead in its tracks. The DBA is now the hero of the beleaguered IT executive in charge of handling the audit. One more step toward that new Tesla.

**Release and Deployment Management**

Similar to the above, software-level releases can be tracked and coordinated with source control, and release activities can be done according to a standard workflow that is an approved ITGC (Information Technology General Control). DBA (or whoever does the deployments - preferably *not* DBA) follows the process, and the record is kept inside the Change Management system and the CMDB reflects the changes.

Another benefit is that ITIL tends to force Development staff to move to a release model and preferably a *scheduled* release model. It also tends to reduce screw-ups, since there is a detailed workflow for each deployment. Scheduled releases mean that you don't get a note like this:  "I need a new DB Server for production with Windows Server 2012R2, SQL 2014 SP1 Enterprise (I don't think we have spare licenses), 24 cores, 512GB of RAM and 10TB of SSD - do you think you can have that up this afternoon?" Every DBA has received such a request at one time or another. By forcing developers to plan ahead, the DBA's job becomes easier.

**Event Management**

Finally, there is the huge benefit of Event Management. If you tie your event management to [your monitoring tool](http://sqlblimp.com/2015/03/06/more-benefits-of-a-good-monitoring-tool/), you can track problems more rapidly, see patterns, and proactively repair issues.

A good monitoring tool should issue an SNMP trap for issues that exceed a certain level of criticality, which in turn should be able to open an *Incident* in your issue resolution system.

Let's look at the different types of events that ITIL expects us to manage:

**Events:** Something has happened. It may or may not require a professional to intervene to change things. However, it needs to be reviewed. Usually a monitoring tool will change a color on its heat map from green to gray or yellow. An example of such an event would be [Page Life Expectancy](http://sqlblimp.com/2015/03/27/page-life-expectancy-a-metric-on-the-wane/) falling below a DBA-defined threshold.

**Incidents:**Here is the official ITILv2 definition of an incident: "*An event which is not part of the standard operation of a service and which causes or may cause disruption to or a reduction in the quality of services and customer productivity.*" To me, that description sounds just like an event that needs to be evaluated. So most people define an Incident as an event which requires intervention by a technician in order to prevent or remediate degraded service. An example of this would be that a database has no remaining growths before filling up its files in either a data file group or the transaction log. If a DBA does not open up another file or clear off unneeded data before the remaining growth fills up, downtime will ensue.

This should be the kind of thing for which the monitoring tool opens up an incident ticket and causes the DBA to be alerted - rapidly. The incident contains logs of the work done, and on what assets the work has been done, thus keeping a record of any changes made in the CMDB.

*Incidents and SLAs* : Ahh, the dreaded Service Level Agreement. The bane of every production DBA, especially as so many shops demand an arbitrary SLA for entertaining an incident and for resolving that incident. Most CIOs are not receptive to the notion that a database-oriented incident SLA needs to be different than that for a floor employee whose workstation is blue-screening. Some are extremely reasonable and understand that different task sets require different SLA. I'm blessed to have a CIO of the latter category. ITIL-aligned tools live and die by the SLA. So it's good for you to get to know the tool. If you can *honestly* stop the SLA clock for an incident (e.g., if you can't do anything until new LUNs are allocated from the SAN by the admins and presented to your server), or reassign the incident to the technician that's slowing down the work, *do it*.

You *want* incidents to be tracked, and you *want* to show how you perform against the SLA. It will drive you and (if you have them) your DBA team to excellence.

**Problems:** Here is the tough nut. Say that you consistently see a replication distribution agent on a SQL Server fail, and generating incident after incident. Your response to each incident was to restart the distribution agent, which worked every time.

These incidents should be grouped into a Problem ticket. The Problem ticket can be worked to find the root cause of a repeated set of incidents and permanently remediate the issue. As an example, let's stick with the scenario of repeated distribution agent failures. So you see the incidents and you begin research. You look at the server's Windows Event log and see that the NIC on the server failed about 3-8 seconds prior to each distribution agent's failure, and that the NIC in question came back online after about 4-5 minutes. Now, you have eliminated SQL Server as the cause of the problem, and you assign the problem to the Hardware team to diagnose and replace the bad NIC.

**Summary**

The ITIL framework is a DBA's friend, once you embrace it and begin to think of your job in an ITIL-aligned manner. After a fairly short learning curve, you will be able to see that your job is made easier by doing it under the framework. You will spend more time doing those things you love about SQL Server.

----

**John F. Tamburo** is the Chief Database Administrator for Landauer, Inc., the world's leading authority on radiation measurement, physics and education. John can be found at @SQLBlimp on Twitter.  John also blogs at [www.sqlblimp.com](http://www.sqlblimp.com/).