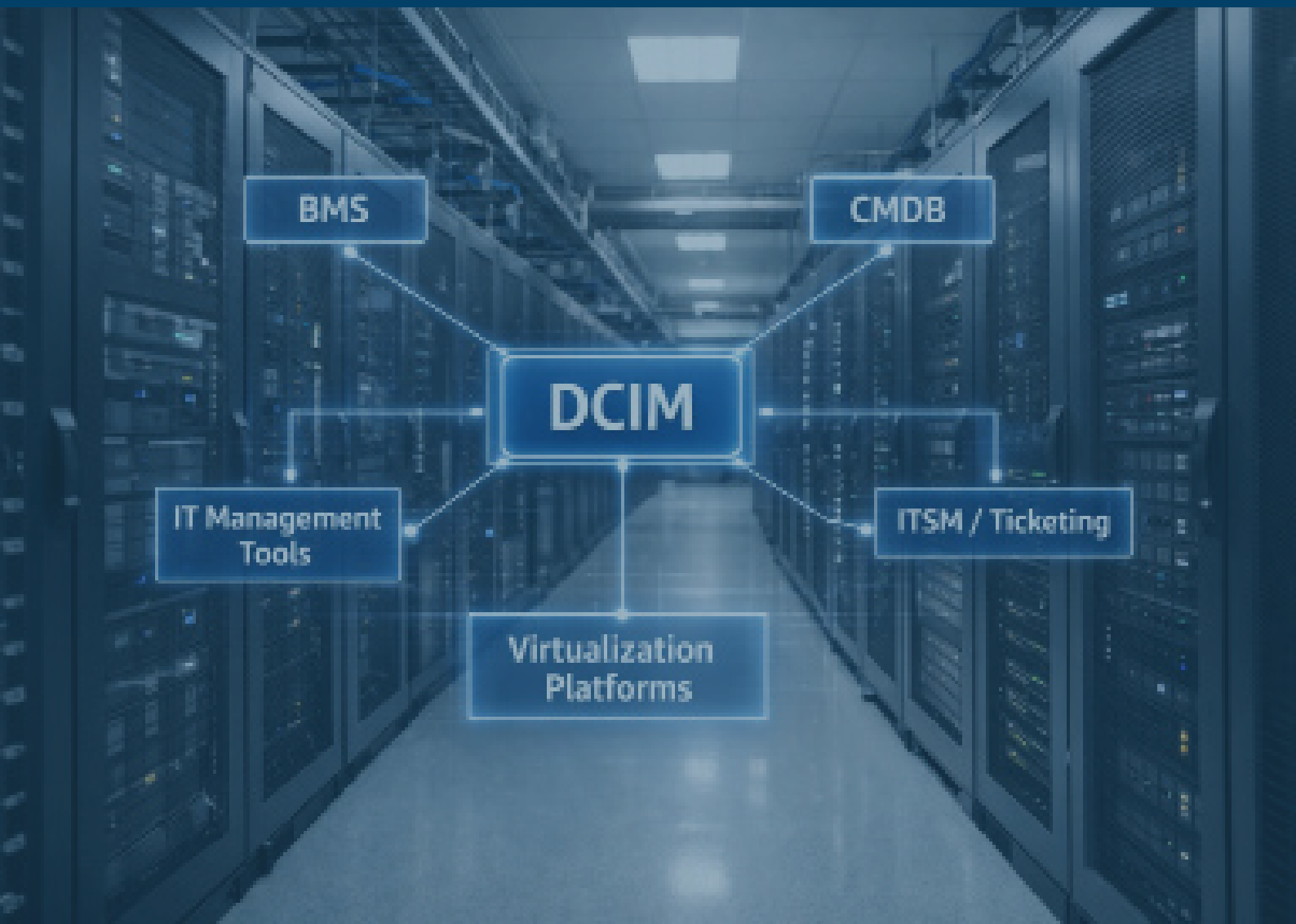
Bjumper

Guía práctia de integraciones en DCiM

Cómo conectar tu DCiM con el ecosistema de tu Data Center sin morir en el intento



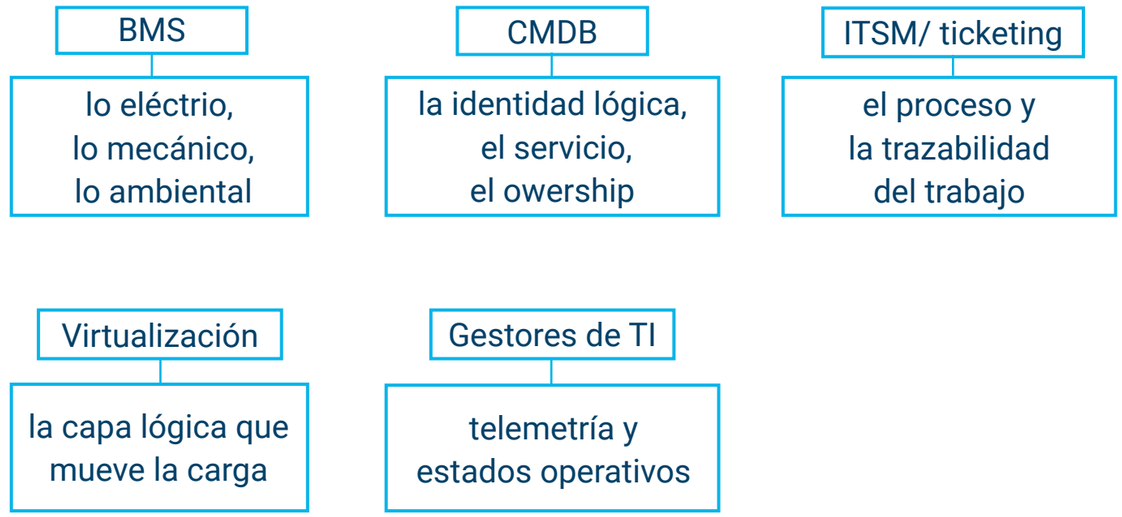
Introducción

Un DCiM “solo” ya aporta valor, pero un DCiM conectado es cuando empieza a convertirse en una herramienta de automatización real para tu Data Center: pasa a ser una pieza viva que **recibe señales, las pone en contexto y las transforma en acción.**

La integración en DCiM no va de “traer datos porque sí”. Va de responder preguntas operativas:

- ¿Qué está pasando de verdad en energía y clima, y cómo impacta en mis racks?
- ¿Dónde está la fuente de verdad del inventario y quién manda en cada campo?
- ¿Cómo reduzco doble carga de trabajo y errores humanos?
- ¿Cómo cruzo el mundo lógico (VMs/servicios) con el mundo físico (racks/hosts)?.
- ¿Donde tengo mi información del servicio y como la pongo a disposición de lo que sucede en la infraestructura?

El Data Center real no vive en un único sistema. Vive repartido entre:



Y todos ellos responden a un mismo objetivo, entregar el servicio en tiempo y eficacia para la continuidad del negocio.

Después de muchas reuniones e integraciones en los clientes, en Bjumper hemos decidido hacer esta guía que te ayude, y nos ayude, a diseñar y ejecutar integraciones de forma práctica, con un enfoque pensado en lo que aporta, en el outcome (el impacto real).

Por eso es importante el qué integrar, por qué, cómo hacerlo bien y qué evitar. Porque el problema no es que existan muchos sistemas donde se encuentran los datos. El problema es cuando no se hablan, ahí duplicas tareas, se rompe la coherencia, aparecen “verdades paralelas” y la operación se vuelve reactiva.

De hecho, esta guía a nivel conceptual podría valer para cualquier integración entre sistemas, no involucrando al DCiM o incluso de sistemas fuera del entorno del Data Center.

Los objetivos

Aunque suene obvio, definir los objetivos que buscan las integraciones entre sistemas es primordial, alinea expectativas entre departamentos y suaviza fricciones, por no decir que te obliga a hacerte las preguntas adecuadas.

Objetivo 1: Integridad del dato (lo primero y este no se puede saltar)

Sin identificadores consistentes, reglas claras y calidad mínima, cualquier integración se convierte en una fábrica de inconsistencias. **El DCiM no "arregla" el dato: lo amplifica.**

Por tanto, la elección de los identificadores debe ser correcta (tienen que ser únicos e inmutables). Si no lo son, cualquier integración acaba derivando en duplicidades, colisiones o dependencias frágiles difíciles de mantener.

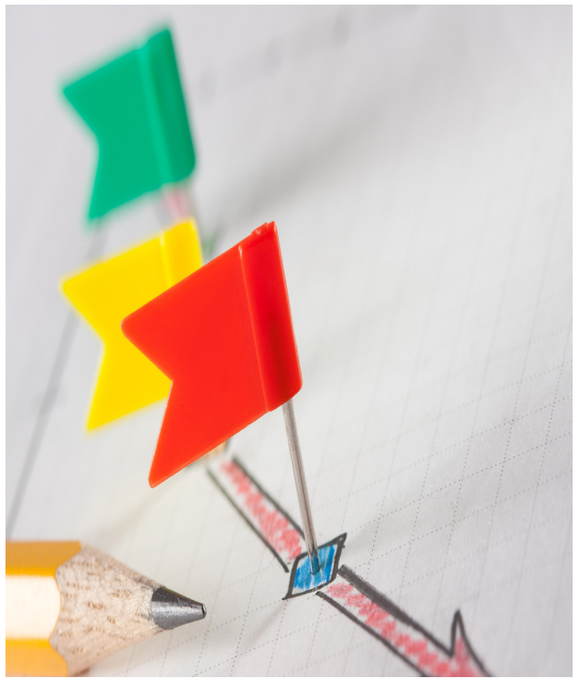
El tipo de integración determina la integridad del dato. No es lo mismo una integración unidireccional que una bidireccional. La primera es más sencilla de gobernar; la segunda introduce mucha más complejidad y obliga a definir reglas claras de sincronización y resolución de conflictos.

Si traes información de otros sistemas al DCiM, es fundamental entender que la gobernanza de ese dato no es del DCiM, sino del origen. Si el origen cambia, debería cambiar en el DCiM, no al revés. En integraciones bidireccionales esto es especialmente crítico: hay que definir ownership por dato y qué sistema tiene prioridad cuando ambos modifican la misma información al mismo tiempo, es decir la fuente de la verdad.

Objetivo 2: Eficiencia operativa

Pensar de forma conjunta y conforme a vuestros procesos, ¿ésta integración como respondería a cómo nos hace más eficientes en la operativa?.

- ¿Tardaremos menos en alguna tarea?
- ¿No tendremos doble carga de trabajo al incluir el mismo dato en 2 sitios?
- ¿Nos permitirá tener menos errores?
- ¿Confiaré más en el dato que me está dando el sistema y no haré un doble chequeo onsite o con otro sistema?
- ¿Podré medir la mejora?



Antes de activar cualquier integración, define cómo vas a saber si ha funcionado. Puede ser algo tan sencillo como medir cuánto tiempo dedicabas a una tarea antes y cuánto dedicas después, o cuántos errores de inventario detectabas al mes.

Sin una referencia medible, la eficiencia es una percepción, no un hecho, y cuando te pregunten si la integración ha valido la pena, necesitas responder con datos, no con sensaciones.

Objetivo 3: Contexto (lo que diferencia a DCiM)

Si vamos a conectar un sistema con el DCiM hay que definir muy bien qué va a aportar al DCiM para darle mayor contexto y buscar la automatización y qué aportaría el DCiM al sistema para enriquecer sus datos.

El DCiM no compite con sistemas tan válidos y propias del entorno del Data Center como el BMS, CMDB, ITSM, Virtualización, Sistemas AIMs, de networking etc, pero no los confundamos tampoco con el concepto propio del DCiM.

Por tanto, la pregunta mágica es ¿Qué contextualiza el DCiM?

Por ejemplo:

- **Del BMS**, la energía y clima enriquecerá al conocimiento por sala, fila, rack y será la base para reportes de consumo por la línea de negocio, consecución de incidentes y cómo adelantarse, etc.

- **De la CMDB** el propietario de un equipo, para comunicar el estado del activo a la persona adecuada, incluso en momentos de mantenimiento preventivo al igual que el DCiM le puede aportar confiabilidad en los datos relacionados con la capa física.

Igualmente no nos podemos olvidar que el DCiM ya dispone de mucho valor que también va a compartir con otros sistemas como:

DCiM → CMDB

El DCiM devuelve a la CMDB:
la realidad física del activo. Ubicación exacta (site, sala, fila, rack, U), estado real de ocupación, conectividad física (puertos, cableado), potencia nominal y consumida. Cuando la CMDB recibe esta información del DCiM, deja de depender de que alguien actualice manualmente un campo de ubicación que en la mayoría de casos está desactualizado.

DCiM → ITSM

El DCIM aporta al ITSM:
Contexto para la ejecución y evidencia para el cierre. Cuando un técnico recibe un ticket de instalación, el DCiM le dice exactamente dónde hay espacio, potencia disponible y conectividad libre.

Al cierre, el DCiM puede confirmar que el trabajo se ha ejecutado (el equipo está registrado, ubicado y conectado), lo que convierte el cierre del ticket en algo basado en evidencia, no en la palabra de alguien.

Además, el DCiM puede ser el que dispare tickets de forma automática: un mantenimiento preventivo programado, una alarma de capacidad al límite o un umbral ambiental superado pueden generar un ticket en el ITSM sin intervención humana.

DCiM → Virtualización

El DCiM aporta a la plataforma de virtualización:
algo que esta no tiene visibilidad, el contexto físico del host. ¿En qué rack está? ¿Qué alimentación tiene? ¿Comparte circuito eléctrico con otros hosts del mismo cluster?

Objetivo 4: Base para automatización (a medio plazo)

Para optimizar los recursos del Data Center, minimizar riesgos La automatización no llega por “meter IA”. Llega cuando:

- el dato es fiable,
- los procesos están claros, tal vez el punto más crítico para la operación de un Data Center.
- la tecnología/integración sostiene el proceso sin fricción.

Conceptos clave: “Monitorizar” vs “Integrar”

Queremos hacer hincapié en la diferencia entre monitorizar versus integrar, para ello vamos a definirlas:

Monitorizar

Monitorizar en un DCiM es capturar datos operativos directamente desde el hardware o sistemas OT (por ejemplo CRACS/UPS/PDUs/sensores), normalmente mediante protocolos estándar y con una frecuencia alta (minutos o incluso más frecuente), para reflejar “lo que está pasando ahora” en la infraestructura.

¿Qué trae?
medidas y estados (kW/kWh, V/A, temperatura, humedad, estado UPS/generador, alarmas básicas…).

Flujo típico:
el DCiM recoge los datos directamente desde el HW, normalmente, salvo casos muy específicos, será unidireccional, ya que el DCiM no va a controlar el HW, no es un sistema de control.

¿Para qué sirve?
operación y control en tiempo casi real: capacidad, riesgo, y poner alarmas con contexto, que estas sí pueden ir al DCiM en tiempo real, pero tampoco es un NOC.

Integrar

Integrar en un DCiM es conectar el DCiM con otros sistemas de gestión (CMDB, ITSM/ticketing, virtualización, gestores IT, AIM, BMS…) mediante API contra API (u otros mecanismos de integración), para sincronizar y complementar información y, sobre todo, conectar procesos entre equipos.

¿Qué trae?
identidad y atributos de activos (CI, servicio, owner, estado lógico, hostname/IP), procesos (tickets/órdenes/estados), relaciones lógicas (VM ⟷ host), etc

Flujo típico:
puede ser unidireccional o bidireccional, según gobierno del dato.

¿Para qué sirve?
reducir doble carga, mantener coherencia entre sistemas y habilitar trazabilidad end-to-end para comenzar con la automatización.

Es decir, la monitorización es una capacidad que tiene el DCiM donde busca sacar datos de equipos electromecánicos o de IT que son capaces mediante protocolos de comunicación de ofrecer telemetría, y la integración es ir sistema contra sistema y puede tener varias metodologías según capacidades de los sistemas o a veces incluso protocolos internos de las compañías: un simple intercambio de ficheros, llamadas concretas a la base de datos con querys o ir API contra API.
Desde nuestro punto de vista, acceder directamente a la base de datos de otro sistema es la opción más frágil y peligrosa. En la medida de lo posible siempre es mejor hacerlo vía API.

Principios de oro (antes de hablar de tecnología)

1. “Fuente de la verdad” por campo

Para cada dato importante define quién manda, esto que a veces puede parecer lógico, se convierte en momento de discusión en la mayoría de las empresas, porque tras los datos están las personas, las cuales tendemos a apropiarnos de su autoría.

La fuente de la verdad es que el dato que manda, cuando tenemos diferentes plataformas, es el que se encuentre más cerca del origen o del activo.

Lo primero si hay datos en plataformas que recogen de forma automática, veáse en cableado inteligente si un puerto está ocupado o no, no hay discusión, pero el tema viene cuando lo que existen son 2 sistemas donde se introducen datos manuales, la pregunta en este caso es en cuál de ellos confías más para ese dato.

Para combatir esto, lo mejor es hacer pruebas, pequeñas auditorías de activos donde se pueda determinar dónde está realmente el dato de calidad.

Estos casos son los minoritarios, hay muchos ejemplos donde sí está claro como:

Ubicación física (rack / U / Sala)	Manda el DCiM
CI / Servicio / Owner / Estado lógico	Manda CMDB o ITSM
Métrica eléctricas y ambientales	Mandra BMS o sistema de monitoreo
Relación VM ⟷ Host	Manda virtualización

Lo importante es proteger la calidad del dato, porque acordaros **El DCiM no “arregla” el dato: lo amplifica.**

2. Identidad inequívoca del activo

Toda integración necesita un “match” robusto, y para ello entre los diferentes sistemas siempre tiene que existir un campo de los activos a integrar que sea único, inequívoco y consensuado.

La regla práctica nos dice que los más usados son:

1.- Serial Number (ideal)
2.- ID único (CI / tag / Asset ID)
3.- Nombre único gobernado (con norma estricta), el menos recomendado.

Si un activo no puede identificarse de forma inequívoca, no lo integres “a medias”. Lo único que vas a crear es ruido, crea la regla de identificación única, pon en marcha el proceso y cuando esto esté al 100% entonces integramos.

3. Integrar “solo datos de valor”

Con las integraciones entre sistemas a veces puede pasar como cuando abres 20 pestañas en el navegador para tenerlo a mano por si acaso, y cuando vas a buscar algo, solo crea confusión e incluso a veces errores.

Integrar todos los datos que estén disponibles es:

- caro
- lento
- frágil
- difícil de mantener

Antes de decidir qué datos vamos a integrar, vuelve a leer los objetivos, te servirá para descartar.

Además, como todo en la vida, o casi todo, podrás integrar más datos en el futuro. Esto no es estático, es dinámico conforme a las necesidades de cada momento.

Igualmente hay que destacar que la frecuencia en la integracion es tan importante como los datos.

El tiempo de muestreo en las integraciones no debe ser igual para todos. Hay datos donde se requiere ver cambios de forma rápida casi instantánea y otros que no son necesarias.

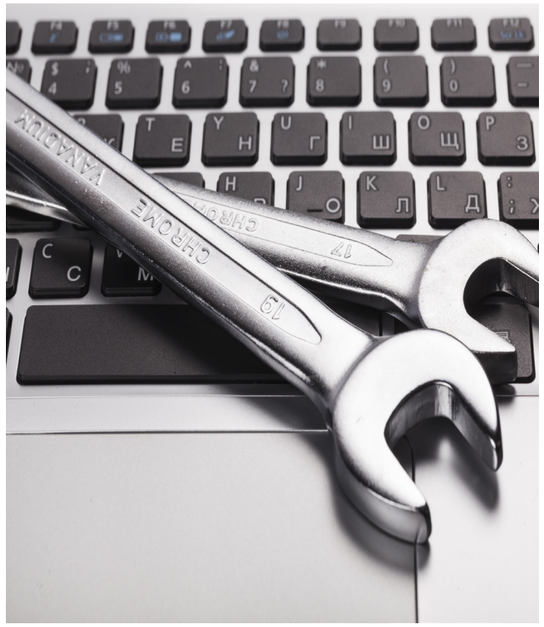
P.ej. la integración de VMware donde la virtualización es muy volátil requiere de mayor nivel de consulta o muestreo que una CMDB.

4. Log y excepciones como parte del diseño

La integración nunca está “perfecta” el día 1. Nosotros aconsejamos diseñar un circuito de mejora:

- lo no mapeado ⟶ log
- Monitorizar la salud de la integración igual que lo hacemos con un servicio.
- revisión periódica ⟶ corrección de datos / reglas
- objetivo: log vacío o controlado

5. El mantenimiento



Los sistemas migran de on-premise a cloud, nuevas versiones, cambias como las usas, es por ello que no podemos obviar que las integraciones se deben mantener en el tiempo, adaptarlas e incluso en ocasiones repersarlas y re-hacerlas.

En estos casos, no se trata de decir que se hicieron mal en el origen, simplemente no podemos pensar que son eternas y que no requieren de nuestra atención y soporte.

Qué hacer y qué no hacer

Vamos con un punto práctico, esto es una guía!

Qué hacer

- Definir casos de uso medibles.
- Definir fuente de verdad por campo.
- Definir identificador único y reglas de matching.
- Empezar por un piloto (ámbito controlado donde confies en tus datos)
- Mapear pocos campos, pero bien.
- Establecer frecuencia de mapeo sensata (según tipo de dato).
- Diseñar logs, alertas y un “proceso de limpieza”.
- Documentar mapeos (aunque sea en una tabla simple), así podremos hacer seguimiento.
- Asegurar seguridad: credenciales, permisos mínimos, TLS, o trazabilidad.
- Preparar una “política de cambios”: cuando cambie el API/versión, ¿qué pasa?

Qué no hacer

- Integrar sin gobierno del dato.
- Integrar sin plan de rollback.
- Usar nombres no únicos o inventar IDs a posteriori.
- Pisar datos sin reglas claras de prioridad.
- Hacer bidireccional “porque suena bien”.
- Integrar con frecuencias que no aportan nada al modelo (minuto a minuto donde no aplica)
- Depender de una persona que “sabe cómo va” sin documentación
- Creer que “si hay integración, ya hay automatización”.

Bjumper

La tabla: Fuente de verdad por campo

A continuacion presentamos una tabla, que no tiene la verdad absoluta, como decíamos antes, cada compañía tiene su organización, procesos y políticas, pero que puede servir de base para una discusión.

Campo/dato	Sistema “dueño” (source of truth)	Sistema(s) que lo consumen	Regla de actualización	Frecuencia	¿Qué pasa en cado de conficto?	Validaciones mínimas
Identificador único (serial/CI/ Tag)	CMDB/ DCiM (según política)	Todos	Nunca se sobrescribe sin control	N/A	Identificar claramente por caso	Único, no nulo, formato estándar
Ubicación física (site/ sala/rack/U)	DCiM	CMDB/ ITSM	DCiM manda	Diaria/ evento	Identificar claramente por caso	No duplicados, consistencia rack/U
Servicio/ aplicación	CMDB	DCiM/ ITSM	CMBD manda	Diaria	Identificar claramente por caso	Necesario en activos críticos
Owner/ Responsable	CMDB/ITSM	DCiM	CMBD/ITSM manda	Diaria	Identificar claramente por caso	Campo normalizado
Estado operativo físico	DCiM/BMS	ITSM	Se reporta no se “inventa”	5-15 min	Identificar claramente por caso	Valores válidos/ umbrales
Energía (kW/ kWh)	BMS/ Medición	DCiM	BMS manda	5-15 min	Identificar claramente por caso	Unidades rangos
Temperatura/ Humedad	BMS/OT	DCiM	BMS manda	5-15 min	Identificar claramente por caso	Sensor mapeado a ubicación
Ticket/ Workorder	ITSM	DCiM	ITSM crea, DCiM ejecuta/ actualiza	Por evento	Identificar claramente por caso	Estados alineados
VM ↔ Host Rack	Virtualización +DCiM	DCiM/ CMDB	Cada uno manda en su capa	1 h	Identificar claramente por caso	Host físico mapeado

Integraciones más comunes

DCiM ⟷ CMDB (inventario físcio-lógico con reglas)

Para qué sirvie

- Evitar doble carga de datos.
- Unificar trazabilidad del activo: unir la capa física con la lógica.
- Mejorar auditoría y compliance a la que se enfrentan las empresas que incluyen el Data Center.
- Preparar el terreno para automatización de procesos.

Diseño recomentado

- Bidireccional solo si hay claridad de fuente de verdad por campo.
- Si no, empezar unidireccional y madurar.

Qué suele vivir en cada lado

- En DCiM: ubicación física, potencia nominal, conectividad física, rack/U, sala.
- En CMDB: CI, servicio, owner, criticidad, estado lógico, hostname, IP.

Reglas prácticas de matching

- Serial o ID único primero.
- Si hay colisiones, se bloquea el update y se manda a log.

Errores típicos

- “La CMDB no está limpia, pero integremos igual”.
- No acotar ámbito (entran entornos no confiables y se envenena el dato).
- Hacer bidireccional sin reglas y que ambos sistemas re-escriban.

DCiM + ITSM/Ticketing (proceso y trazabilidad)

Para qué sirvie

- Conectar solicitud con la ejecución y el cierre del proyecto.
- Incorporar a los peticionarios a los flujos del Data Center.
- Reducir tiempos de coordinación IT/Ops/Facilities.

Patrones habituales

- Ticket crea proyecto/orden de trabajo en DCiM.
- DCiM devuelve estado y cierre al ticket.
- Disparadores por evento (aprobado, en ejecución, finalizado).

Consejos prácticos

- Definir “tipo de ticket” que dispara trabajo (cambio / request / incidente).
- Definir campos mínimos obligatorios en formularios, si no, se devuelve ticket.
- Definir cuándo un trabajo se considera “cerrado” (evidencia)

Errores típicos

- Integrar sin diseño de proceso (solo “sincronizar estados”)
- No alinear taxonomía: estados del ITSM ≠ estados del DCiM
- No definir roles (quién aprueba, quién ejecuta, quién valida)

DCiM + Virtualización (del lógico al físico)

Para qué sirvie

- Visibilidad de hosts, clusters, VMs, recursos.
- Relación servicios/VMs con activos físicos.
- Mejor planificación de capacidad (más allá del rack).

Frecuencia recomendada

- cada 1 hora suele ser suficiente en la mayoría de casos.
- Si lo quieres “near real-time”, necesitas justificar el uso (y asumir coste).

Buenas prácticas

- Empezar con inventario + relaciones (host/cluster/VM).
- Añadir consumo/recursos después.
- Validar reconciliación: una VM “vive” en un host que debe existir físicamente.

Errores típicos

- Querer “mapear todo” (etiquetas, políticas, redes complejas) desde el día 1.
- No tener mapeo sólido host ↔ servidor físico.
- No acordar qué se considera “fuente de verdad” del estado (encendido/apagado).

DCiM + Gestores IT (métricas IT operativas)

Para qué sirvie

- Traer telemetría IT relevante al contexto físico.
- Detectar riesgos operativos (temperatura, consumo, fallos).
- Apoyar capacity planning con señales reales.

Frecuencia recomendada

- diaria para inventario/estado base.
- más frecuente solo si hay caso claro (alerting operativo).

Buenas prácticas

- Definir mínimo viable: CPU/mem/consumo/temp/firmware e identificación.
- Gestionar excepciones (IPs duplicadas, cambios de hostname, etc.).
- Correlacionar con DCiM por serial o ID robusto.

Errores típicos

- Integrar sin identificador fiable (IP cambia, hostname cambia).
- Convertir el DCiM en un NMS (no es el objetivo).
- Traer métricas excesivas sin uso real.

La despedida (la pregunta que lo resume todo)

¿Cómo saber si lo hiciste bien?

Si mañana te apagan la integración, ¿la operación empeora de forma medible?

Si la respuesta es “sí”, lo hiciste bien: integraste valor.
Si la respuesta es “más o menos…”, probablemente integraste “datos”.